MODELING IN APPLIED MATHEMATICS

Bengt Fornberg and Ben Herbst

Department of Applied Mathematics
University of Colorado
Boulder, CO 80309
USA
Email: bengt.fornberg@colorado.edu

Applied Mathematics
University of Stellenbosch
Stellenbosch 7601
South Africa
Email: herbst@sun.ac.za

Contents

NOTE. When this manuscript grows up it wants to be a book. In the mean time you will have to live with its growing pains. We do not take any responsibility for any injury, real or imaginary, that may result from using it. If you like it, please let us know what you like about it. If you don't like it, please tell us why, and what we can change to make it better. And if you know of good examples that might be useful, tell us about it.

This manuscript is somewhat unusual in the sense that it might not be possible to read it front-to-back. When we discuss the Applications in the first part for example, we sometimes refer to material that is covered in detail in later parts of the manuscript. It means that the reader may find it necessary to return to topics for a full understanding, after excursions to other parts of the manuscript. We don't really want to apologize for this. It is how things work in practice, at least in our experience. When first presented with an interesting problem, most of the time we have only a vague idea (or none at all) of how to solve it. It is only after repeatedly returning to the problem after numerous excursions, that we sometimes come up with something useful.

In earlier versions we had a section with Matlab code. It has been taken out of this version but it still exists. If you want to play with the code (highly recommended), we plan to make it available on some or other website, probably close to where you found this manuscript. Otherwise, please contact one of the authors for details.

Part 1

APPLICATIONS

CHAPTER 1

TOMOGRAPHIC IMAGE RECONSTRUCTION

1.1. Introduction.

In medicine as well as in many other situations, it is invaluable to be able to look inside objects without actually needing to slice them open, or to do something else that is grossly invasive (we regard here taking an X-ray image as 'non-invasive'). The problem with conventional X-ray imaging is that all objects along the path of the X-rays appear to be superposed on top of each other (a bit like taking many exposures without winding the film in a camera). A 3-D object has been *projected* to 2-D (with a big loss of information content, especially since one is often interested in localized and subtle changes in soft tissues that are near-transparent to X-rays). The methods we will discuss in this chapter allow full spatial reconstruction throughout a 3-D object or throughout a 2-D *slice* (in Greek $\tau\omega\mu\omega\sigma$—hence the name tomography) of the object.

In Section 1.2 we discuss very briefly six different approaches to non-invasive imaging techniques. The remaining Sections 1.3—1.8 are focused on Computational Tomography (CT)—a means of getting full reconstructions in one or—simultaneously or sequentially—many slices through the object. The input data is numerous X-ray images captured on 2-D 'film-like' or 1-D line-type electronic detectors. This data is then computationally processed to create the full, spatially true reconstruction. Of the several possible computational approaches to this reconstruction, we will focus on three: least squares (LS), filtered back projection (FBP) and Fourier reconstruction (FR).

The applications of CT extend far beyond medicine. To mention one example: In the 1980s, Exxon developed micro-tomography, mainly in order to explore the detailed pore structures of coal and of oil-saturated sand stones. One might think that non-invasiveness would not be particularly important for such objects, but,

understanding for example how oil flows to wells requires knowledge about how microscopic pores and channels in sand stone are connected. With the grains extremely hard compared to the pores, invasive procedures would inevitably destroy the pore structures before they could be recorded (a little bit like trying to feel the fine structure of a snow flake with bare fingers—the evidence would just 'melt away'). In contrast to medical tomography which achieves mm-size 2-D resolution across a slice of body-sized objects, micro-tomography achieves μm size 3-D resolution throughout cubic mm sized samples. The X-ray source is typically synchrotron radiation from an accelerator. The 10^9pixel ($1000 \times 1000 \times 1000$) volume image sets require the fastest computational inversion (FR), whereas medical imaging traditionally is based on the much slower FBP method. Here the data sets (typically 2-D) are much smaller, and throughput is limited more by patient handling than by equipment speed.

1.2. Non-invasive medical imaging techniques.

In the last few decades, several non-invasive imaging techniques have been discovered which are capable of providing full 3-D information of an object. Earlier methods had either

- loss of spatial information (e.g. standard X-rays, failing to discriminate between overlapping structures), or
- been highly invasive (for example 'serial-section microscopy', typically requiring the object to be frozen and then sliced up).

Important non-invasive imaging methods include

(1) *Ultrasound*. Very high sound frequencies (several MHz) allow beams to remain very narrow. These beams are typically sent / received by a small transducer which is held in contact with the body. It can rapidly change the direction of the beam, making it 'sweep' an angular domain. Echoes from interfaces between different soft tissue types are recorded, with time delays corresponding to depths. The technique is used for a large number of organs, including fetal monitoring during pregnancy. A potential future application - still requiring further developments - is to replace X-rays for mammography. Strong sound absorption by bones somewhat limits its use, for ex. in brain studies. The method gives images in real-time, and the equipment is

inexpensive. It used to be considered entirely safe, but some doubts about this emerged after a recent Swedish study showed that, after two scans, the likelihood of left handedness in babies increased by 32%. Although this is small compared to the 5 times increase that has been reported in cases of premature birth, the fact that it has any effect at all gives rise to concerns. However, present opinion seems to be that the benefits well outweigh the possible dangers.

(2) *CT - Computerized Tomography.* A parallel sheet of X-rays is sent through the object, and recorded by a 1-D row of detectors. From the accumulated data when source and receiver (or object) are rotated 180°, cross-sectional images can be computed. In medical application, the resolution is normally about 0.3 mm. With the use of much more intense X-rays (which would destroy living tissues; such X-rays can be obtained from accelerators in the form of synchrotron radiation), resolutions around 0.001 mm (= 1 μm) are achieved. This is comparable to the best resolution that is possible with optical microscopes, used on sliced samples. Some drawbacks with medical use of X-ray tomography include possible tissue damage from ionization (X-ray absorption depends on the target's electron density), and low contrasts between different types of soft tissues, for example between malignant and healthy tissues.

Mathematical tools needed for successful CT imaging were discovered more than once, not recognized for their potential and then forgotten before successful experimental realizations (employing less effective algorithms) were achieved. For independent pioneering work in experimentally realizing CT and bringing it to medical use, the 1979 Nobel Prize in Physiology and Medicine was awarded jointly to G. Hounsfield and A.M. Cormack. The history of CT and other applications of it are described in more detail in Section 1.3.

(3) *MRI - Magnetic Resonance Imaging* (earlier called NMR - Nuclear Magnetic Resonance). The object that is to be imaged is placed in a very strong, highly uniform magnetic field (e.g. inside a large superconducting magnet). Two different, relatively weak magnetic gradients are introduced — one

stationary and orthogonal to it, one that is stepped in time. When subjected to accurately tuned high frequency radio pulses, many light nuclei (with an odd number of nucleons, such as hydrogen) start to spin. While returning to a state of magnetic alignment, they re-radiate these waves. The frequency is proportional to the local magnetic field, i.e. it carries information about the positions of the different atoms. The numerical techniques needed to create images are similar to those used in CT. However, Fourier inversion is nowadays preferred over back projection-type algorithms.

Advantages of MRI over CT in medical applications include

- high contrast between many different soft tissues,
  - possibility (although little used) to 'tune in' on different atoms with very distinct biological functions (e.g. H^1, Na^{23}, and P^{31} resonate at 42.57, 11.26 and 17.24 MHz respectively in a field of 1 Tesla), and
    - far safer radiation (the frequencies are about 11 orders of magnitude lower than those of X-rays - the associated electromagnetic quanta carry correspondingly less energy, and cannot alter molecules of living tissues). In spite of using wavelengths in the 5—25 meter range, 1—2 mm resolution is obtained.

Disadvantages compared to CT include slightly less resolution and higher cost of equipment. In practical usage, the big risk factor turns out to be that inadvertently present metallic objects can become dangerous projectiles due to the extreme magnetic fields.

The Nobel Prize in Physics for 1952 was awarded to E. Purcell and F. Bloch (at Harvard and Stanford Universities) for their discovery of the NMR phenomenon. The problem of obtaining spatial information from NMR data was considered already in the early 50's and solved in different ways in the mid-70's. Routine medical use began in the mid-80's. Technology improvements have reduced recording times from hours to, in some cases, 30-100 ms for instance, when using echo-planar imaging (EPI)—a high-speed recording technique that permits a full image to be obtained in a single nuclear

excitation cycle, as opposed to a few hundred cycles; cf. [**?**]. A summary of the principles of MRI is given, for example, in [**?**].

(1) *PET - Positron Emission Tomography.* A radioactively labeled substance is injected and follows the blood stream, while emitting positrons. After traveling a very short distance, a positron will encounter an electron, and annihilate it. The energy gets transferred into two gamma rays that are sent off in nearly perfectly opposite directions of each other. When two detectors (out of a big array surrounding the body part—typically the head) detect signals at the same instant, the emission is assumed to have occurred along the straight line between them. This procedure generates data on the accumulated concentrations of the tracer substance along a large number of different lines through the body, thus allowing its distribution to be reconstructed through 3-D generalizations of the 2-D CT algorithms that are described in Sections 1.4—1.8.

When using radioactively labeled glucose, brain activities can be followed in 'real time', since the blood flow (and glucose usage) very quickly responds to areas of activity (however, EPI-type MRI, in connection with the use of contrast agents in the blood, offer competition to PET in this field). Another usage is based on the fact that certain substances tend to concentrate in different tissues, e.g. Cu^{64} can be used to spot some brain abnormalities. Disadvantages include quite low resolution, very high cost, and possibly dangerous radiation levels which are somewhat minimized by the use of radio-isotopes with short half-times. However, this requires the availability of a nearby reactor or accelerator.

The last two methods to be mentioned here are entirely non-invasive also as far as waves and radiation are concerned. However, their ability to provide true imaging is very limited. Both can record brain signals in cases when thousands of neighboring neurons fire in a synchronized manner.

(2) *EEG - Electroencephalography*; electric potentials on the scalp are recorded at tens (or more) locations with time resolutions in milliseconds. The low spatial resolution and the mathematically ill-posed inversion problem makes

the technique more important in studying neural firing patterns than for imaging.

(3) *MEG - Magnetoencephalography*; the very weak magnetic fields from neural activities are picked up outside the scull by SQUIDs (superconducting quantum interference devices), possibly the most sensitive recording devices of any kind. Using low temperature, liquid helium cooled, superconductors, individual flux quanta can be recorded. This can in turn be utilized for a variety of measurement tasks, giving astounding precisions, e.g.

magnetic field	10^{-15}T	(= 1 fT; femto-Tesla); signals from the brain reach about 10-100 fT (measured outside the skull); from the heart 50,000 fT; the earth's field is about 10^{11}fT = 10^{-4}T.
voltage	$10^{-14}V$	about 5 orders of magnitude better than semiconductor voltmeters,
motion	10^{-18}m	about 1/1,000 of the diameter of an atomic nucleus; 1/1,000,000 of the typical diameter of an atom.

'High temperature' (using liquid nitrogen at 77K) superconducting SQUIDs are much cheaper than liquid He-ones; however their ability to detect detect fields of around 25 fT is only barely sufficient for brain studies.

Although SQUIDs operate much faster than neurons, acceptable signal-to-noise ratios when applied to brain imaging require recording times in tens of seconds. Already in 1853, it was shown by Helmholtz that the inversion problem, determining internal currents from external magnetic fields, was not uniquely solvable. Like for X-ray crystallography (Section 30.2), additional data needs to be supplied. Possibilities for MEG include the use of

- simultaneous EEG-data for potentials. This offers the best signal when currents are orthogonal to the scull—the magnetically least visible case, and
  - MRI-provided structural information. This will pinpoint folds in the cortex. As it happens, primary sensor areas tend to be located in such

folds, with the consequence that the key currents become relatively parallel to the skull, i.e. well oriented for a good magnetic signal.

MEG is described in Hämäläinen (1993). One of the inventors of MEG (D. Cohen, MIT) has raised serious questions about the utility of the approach [**?**].

1.3. Additional Background on Computerized Tomography.

From a mathematical point of view, the main challenge in CT lies in the inversion technique. In a purely mathematical form (prompted by an issue relating to gravitational field equations), this problem was solved by the Austrian mathematician Johann Radon [**?**]. His solution assumes that all variables are continuous functions defined on an infinite domain. In practice, one has to work with a finite number of rays at a finite number of angles to produce a reconstruction at a finite number of grid points. In the exercise section of this book, we will see how Radon's inversion method connects to two of the presently most used techniques—*filtered back projection* and *Fourier inversion*.

The paper by Radon was just the first of several which gave solutions to the inversion problem which were not noted by later pioneers. Another case is a paper by R. Bracewell [**?**] (in the context of obtaining images of the sun from microwave data) which describes a Fourier-based reconstruction method. With the use of the FFT (fast Fourier transform) algorithm, Fourier based reconstruction methods are now the fastest ones available. Although surprisingly little used in medical contexts (where filtered back projection dominates), they are preferred in the even more data intensive application of *micro-tomography*.

Allan MacLeod Cormack (1924-1998) was working as a physicist (at University of Cape Town) and assisting a local hospital with routine radiological tasks, when it occurred to him that if enough X-ray projections were taken in a variety of directions, there would be enough data for a full reconstruction. He realized that CT could revolutionize medical imaging and his tests on simple wood and aluminum objects in the late 1950s and early 1960s, showed the concept to be practical. In two seminal papers [**?, ?**], Cormack very clearly outlined the medical implications, and presented still another numerical procedure for the reconstruction. However, his efforts at the time to interest the medical community were not successful.

About a decade after Cormack's pioneering work, Goodfrey Hounsfield (1919-) independently developed the idea of medical tomography (while doing pattern recognition studies at the electronics company EMI Ltd. in Britain). His first apparatus was similar to Cormack's, but used an americum radiation source and a crystal detector. Following very successful preliminary tests, the radionuclide source was replaced with an X-ray tube, reducing data gathering times from well over a week to about 9 hours. Following many further improvements, his work led to the first clinical machine, installed in a hospital in Wimbledon in 1971. By this time the technique had advanced to the point that 180 projections (at 1° separation) could be collected in just under 5 minutes, followed by about 20 minutes for the image reconstruction. These specifications improved even more and current machines provide about 0.3 mm resolution throughout full body slices. The major limiting factor in further improvements comes from the need to keep X-ray doses within safety limits.

There are many applications other than medical ones, of tomography. Below are just a few examples:

astronomy	Marsh and Horne [1988] (binary stars),Gies et al [1994](accretion discs) Hurlburt et.al.[1994] (coronal studies)
oceanography	Worcester and Spindel [1990],Worchester et.al. [1991] Munk et.al. [1995] (acoustic probing of ocean conditions
geophysics	Anderson and Dziewonski [1984] (mantle flows), Frey et al [1996] (aurora) Gorbunow [1996] (atmosphere)
porous media	Hal [1987]

1.4. Model Problem.

Figures 1.4.1 and 1.4.2 show a test object, defined on a 63×63 grid. This object was generated by the code logo.m. The X-ray absorption levels at different locations are displayed as darkness and elevation respectively. Figure 1.4.3 shows how X-ray data can be collected for a sequence of angles $\theta_i = \frac{\pi\, i}{64}, i = 0, 1, ..., 63$. Figure 1.4.4 shows what the scan data would look like in the case of the test object in Figures 1.4.1 and 1.4.2. The scan lines are shown as successive lines (in the r-direction)

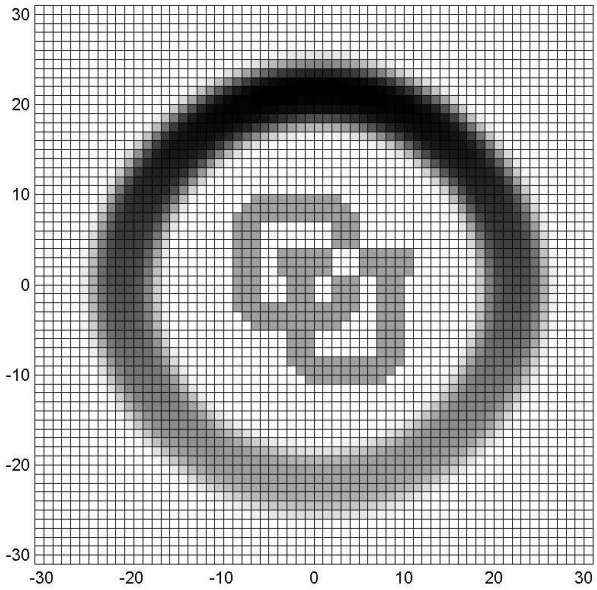


FIGURE 1.4.1. Test object.

from the front left, $\theta = 0$, to the back right, $\theta = \pi$; this last line being an up-down reflection of the first one.

Given the density function $f(x, y)$ of the 2-D object, the scan data can be written as

$$g(r, \theta) = \int_{-\infty}^{\infty} f(x, y)\, ds \tag{1.1}$$

where the coordinate axes are as defined in Figure 1.4.5. Since the (s, r) axes differ from the (x, y) axes by a pure rotation, they are related by

$$\begin{cases} s = x\cos\theta + y\sin\theta \\ r = -x\sin\theta + y\cos\theta \end{cases} \qquad \begin{cases} x = s\cos\theta - r\sin\theta \\ y = s\sin\theta + r\cos\theta \end{cases} . \tag{1.2}$$

This means that we sum all the contributions along lines parallel to the s axis in Figure 1.4.5.

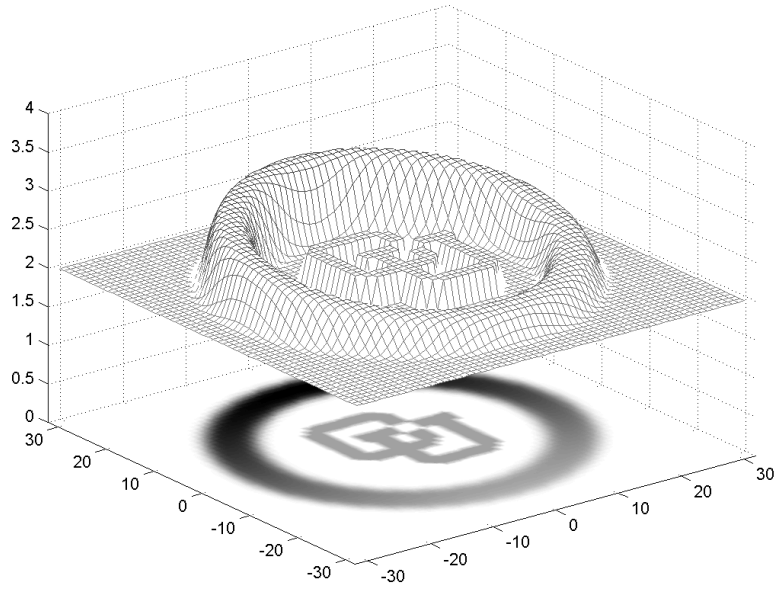


FIGURE 1.4.2. Another view of test object.

Equation (1.1) is known as the *Radon transform*. The computational issue in CT is to invert this transform, i.e. to recover $f(x, y)$ from $g(r, \theta)$.

1.5. Least Squares Approach.

To understand the idea behind the back projection method (how to achieve the reconstruction, how much data is needed etc.) we consider first a very small object—a 3×3 structure of 9 square elements, having unknown densities $x_1,\ x_2, \ldots, x_9$ respectively:

(1.1)
$$\begin{array}{|c|c|c|}\hline x_1 & x_4 & x_7 \\ \hline x_2 & x_5 & x_8 \\ \hline x_3 & x_6 & x_9 \\ \hline \end{array}\ .$$

Suppose that we have knowledge only of the row sums r_1, r_2, r_3 and of column sums s_1, s_2, s_3 (like having done X-ray recordings only horizontally and vertically). This

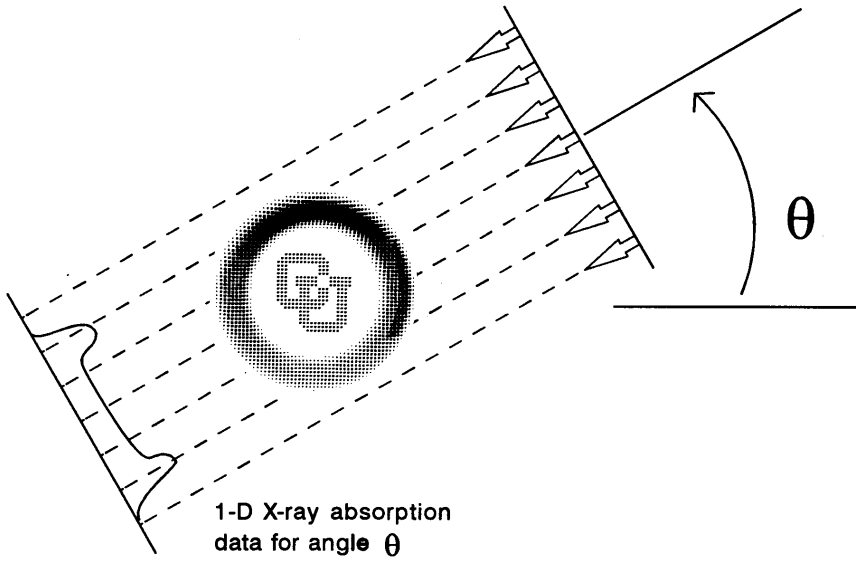


FIGURE 1.4.3. Principle for generation of 1-D scan data from a 2-D object.

gives rise to the linear system of equations

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \end{bmatrix} = \begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ s_1 \\ s_2 \\ s_3 \end{bmatrix} \tag{1.2}$$

The first question that we need to ask ourself is whether it is possible to obtain the densities $x_1,\ x_2, \ldots, x_9$ of the elements from these row- and column sums only. Considering that all the six density patterns shown in Table 1 have the same row- and column sums, it is clear that the problem will not always have a unique answer. Next, we might ask if (1.2) will always have at least one solution, no matter what

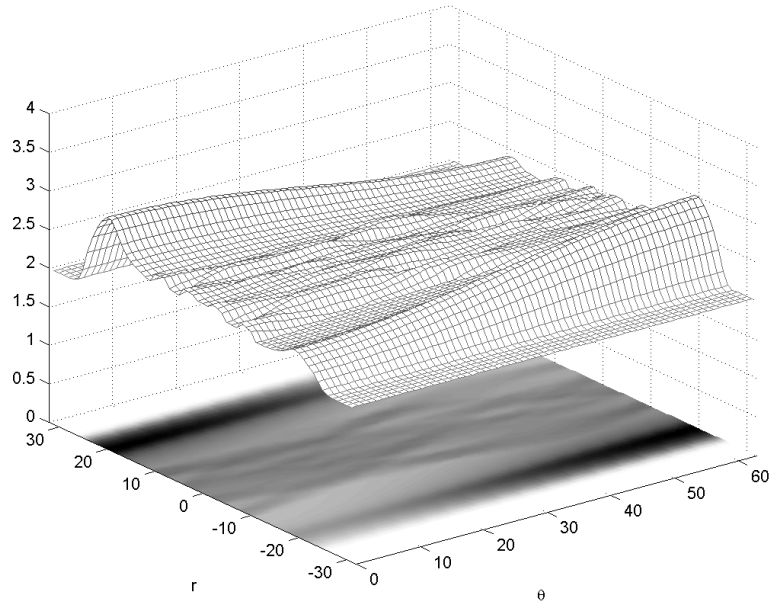


FIGURE 1.4.4. Scan data of the CU-object.

the values are in the right hand side. It is easy to see this can't be the case (in spite of the fact that this system $Ax = b$ has fewer equations than unknowns). If we add rows 1, 2, and 3, we get

$$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 = r_1 + r_2 + r_3$$

while adding rows 4, 5, and 6 gives

$$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 = s_1 + s_2 + s_3$$

Unless

$$r_1 + r_2 + r_3 = s_1 + s_2 + s_3 \tag{1.3}$$

holds, there is no possibility for a solution to exist. We might argue that (1.3) should hold if our data came from any object, such as the one indicated in (1.1). However, all actual data contains errors of some size, and one can never rely on data having to be completely error free. For much bigger linear systems than (1.2), there is no

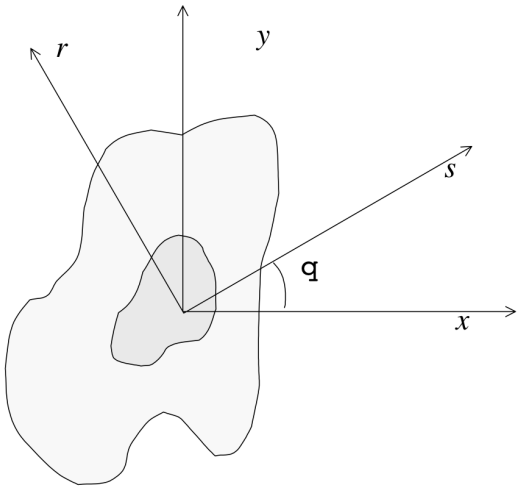


FIGURE 1.4.5. Relation between the (x, y) and (s, r) coordinate systems.

1	0	0	1	0	0	0	1	0	0	1	0	0	0	1	0	0	1
0	1	0	0	0	1	1	0	0	0	0	1	1	0	0	0	1	0
0	0	1	0	1	0	0	0	1	1	0	0	0	1	0	1	0	0

TABLE 1. The six possible permutation matrices of size 3x3

chance to spot multiple solutions or situations with non-existence of solutions in the way we have just dome. What is needed are general results telling just when systems have solutions (and, if so, how many) or do not have any. If there are solutions, how does one effectively find them? Maybe surprisingly, systems with more equations than unknowns—almost invariably lacking exact solutions altogether—is the most important case in applications. And we will soon see that our first approach to tomographic inversion is an illustration of this.

1.5.1. SVD analysis of (1.2). Usually the best way to explore the solvability of any specific linear system starts with performing an SVD factorization of the coefficient matrix A (cf. Section ....; in particular note Figure .... illustrating how the decomposition related to the four fundamental subspaces of a matrix) Writing this decomposition as $A = U\,\Sigma\,V^*$, we get in the particular case of the coefficient matrix A in (1.2)

$$U = \begin{bmatrix} -0.4082 & 0 & 0 & 0.8165 & 0 & 0.4082 \\ -0.4082 & 0 & -0.2599 & -0.4083 & 0.6576 & 0.4082 \\ -0.4082 & 0 & 0.2599 & -0.4083 & -0.6576 & 0.4082 \\ -0.4082 & 0.5393 & -0.5701 & 0 & -0.2253 & -0.4082 \\ -0.4082 & -0.8006 & -0.1493 & 0 & -0.0590 & -0.4082 \\ -0.4082 & 0.2612 & 0.7194 & 0 & 0.2843 & -0.4082 \end{bmatrix},$$

$$\Sigma = \begin{bmatrix} 2.4495 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1.7321 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1.7321 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1.7321 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.7321 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$V^* = \begin{bmatrix} -0.3333 & 0.3114 & -0.3292 & 0.4714 & -0.1301 & 0.6327 & 0.1464 & 0.1503 & -0.0096 \\ -0.3333 & -0.4622 & -0.0862 & 0.4714 & -0.0341 & -0.4331 & 0.2297 & 0.1479 & 0.4269 \\ -0.3333 & 0.1508 & 0.4154 & 0.4714 & 0.1642 & -0.1997 & -0.3762 & -0.2981 & -0.4174 \\ -0.3333 & 0.3114 & -0.4792 & -0.2357 & 0.2496 & -0.2662 & -0.5409 & 0.1831 & 0.2179 \\ -0.3333 & -0.4622 & -0.2363 & -0.2357 & 0.3456 & 0.3042 & 0.0479 & -0.5903 & -0.0332 \\ -0.3333 & 0.1508 & 0.2653 & -0.2357 & 0.5438 & -0.0380 & 0.4930 & 0.4073 & -0.1846 \\ -0.3333 & 0.3114 & -0.1791 & -0.2357 & -0.5098 & -0.3665 & 0.3945 & -0.3333 & -0.2083 \\ -0.3333 & -0.4622 & 0.0638 & -0.2357 & -0.4137 & 0.1288 & -0.2777 & 0.4425 & -0.3937 \\ -0.3333 & 0.1508 & 0.5654 & -0.2357 & -0.2155 & 0.2377 & -0.1168 & -0.1091 & 0.6020 \end{bmatrix}.$$

The last entry (zero) in the main diagonal of Σ tells that the rank of the system is 5 (rather then 6, as might have been expected). The first 5 columns of U form then an orthogonal basis for the column space of A, i.e. the system (1.2) is solvable if and only if the right hand side b of (1.2) lies in this space. The columns of U are orthogonal - so another way to say this is that b needs to be orthogonal to the last

column of U, giving the condition for solvability

$$r_1 + r_2 + r_3 - s_1 - s_2 - s_3 = 0\ . \tag{1.4}$$

We arrived, mainly by chance, earlier at this very same requirement (1.3). The big difference is that we now have derived it in a manner that works for absolutely any system. And we also now see that this is the only requirement that is needed for solvability. Physically, this condition is natural: $r_1 + r_2 + r_3$ and $s_1 + s_2 + s_3$ both express the same quantity, viz. the sum of all the nine unknowns. If (1.4) holds, the general solution is any one particular solution to which we can add any combination of vectors from the null space of A. By the same theory about SVD and the fundamental subspaces, this space will be found in the last 4 rows of V^*.

Once we have the SVD of A, we can alternatively extract all the information above - plus find a particular solution—without referring to the fundamental subspaces. The system $Ax = b$ can be written $U\Sigma V^* x = b$, i.e.

$$\Sigma y = U^* b \tag{1.5}$$

with

$$x = Vy. \tag{1.6}$$

In order for (7.1) not to contain a contradiction in the last row, we need the last element of U^*b to be zero, leading once again to the condition (1.4). With that being the case, (7.1) gives uniquely the first 5 entries of y, but leaves the last 4 free. The most general solution then follows from (7.4). We note again that the solution is undetermined precisely with respect to any combination of the last 4 rows of V^*. This last result is very much more complete than our earlier observation that the six particular solutions represented by the matrices in Table 1 all corresponded to the same RHS for (1.2).

1.5.2. A least-square formulation of the CT inversion problem. The most obvious shortcoming with using only row- and column sums for the model (1.1) is that we do not get enough data for the number of unknowns. Increasing the resolution from 3×3 to $n \times n$ elements does not help; we will then only have $2n$ equations for n^2 unknowns. The 'obvious' remedy to this is to do the scans in many

more directions than just two. There is no limit to how many directions we can use. With m directions and, for each of these, sending n side-by-side rays through an object made up of $n \times n$ elements, we will instead of the $n = 3$, $m = 2$ case represented by (1.2), obtain a linear system of the type

$$\begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & A & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix}_{mn,n^2} \begin{bmatrix} \cdot \\ x \\ \cdot \end{bmatrix}_{n^2} = \begin{bmatrix} \cdot \\ \cdot \\ b \\ \cdot \\ \cdot \\ \cdot \end{bmatrix}_{mn} . \tag{1.7}$$

If we have more equations than unknowns, the system becomes overdetermined, in which case there is typically no exact solution. However, least squares solutions can be readily found (cf. Section 11.4), making the difference between left hand– and right hand sides of the system as small as possible. Typically, using $m >> n$ will not be disadvantageous (as one might think, based on introducing more possibilities of conflict) but instead advantageous (by making the solution more stable against data noise). The entries in the coefficient matrix are now not only 0's or 1's (as in (1.2)), but instead real numbers which reflect how visible each element is to each ray. When we only used horizontal and vertical rays, they either went through the full extent of an element, or they missed the element altogether. With rays going thorough the block-structured object at arbitrary angles, the length of the intersection between an element and a ray can take any value between zero and the length of its diagonal. The coefficient matrix A will be quite sparse, and with a complicated sparsity structure.

With the object consisting of $n \times n$ elements (whose densities are to be determined), the detailed structure of the system (1.7) becomes as shown in Figure 1.5.1. There are n^2 densities that need to be calculated. We rearrange the $n \times n$ set of unknowns as before into a column vector $\mathbf{x}$ of length n^2. The first n equations correspond to the n rays when scanning at the first angle; the next n equations corresponding to the second angle, etc. It is natural to think of A as made up of an $m \times n$ layout of $n \times n$-sized blocks, and x and b of n and m end-to-end vectors,

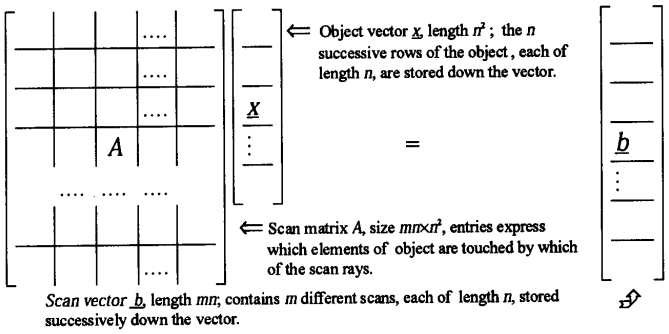


FIGURE 1.5.1. Structure of the overdetermined linear system of the least-square method.

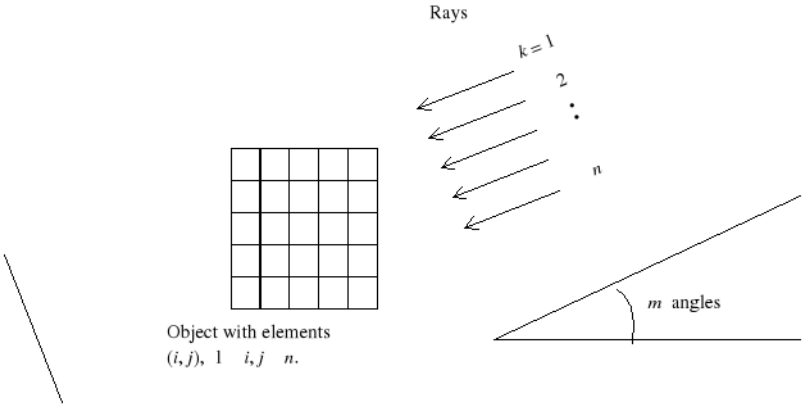


FIGURE 1.5.2. Illustration of how the A-matrix entries are formed.

respectively, each of length n. Figure 1.5.2 illustrates how the linear system is formed (with one block row of A for each scan angle).

1.5.3. Least square solution. We now need to solve the overdetermined system (1.7) in a least squares sense. The best approach will be to use an iterative method that is capable of fully utilizing the sparsity structure of A. One such algorithm is Matlab's *lsqr*—a conjugate gradient implementation of the normal equations. In Section 11.3 it is described how a direct solver can be implemented in a stable

way using QR factorization. Codes for both of these methods are included in the computer code section .... For a simple operation count for a direct solver, we consider the *normal equation* approach (roughly the same count as for the QR method, but less stable numerically, so not recommended). The normal equations of a linear $mn \times n^2$ system $Ax = b$ is a square $n^2 \times n^2$ system

$$A^T Ax = A^T b \ .$$

Forming $A^T A$ will cost $O(mn^5)$ operations, $A^T b$ costs $O(mn^3)$; Cholesky decomposition of $A^T A$ into $L\ L^T$ will add another $O(n^6)$. The final step to get x through two back substitutions, using L and L^T respectively, adds a further $O(n^4)$ operations. Assuming m is just slightly larger than n, the total cost becomes $O(n^6)$ operations.

A very large saving can be realized by noting that the A-matrix is independent of the object we are studying—that will only affect the b- and x- vectors. The L-matrix can therefore be determined once and for all. Each image will then cost 'only' $O(n^4)$ operations. We will soon see that even this is not competitive—the next two methods, filtered back projection and the Fourier method will cost only $O(n^3)$ and $O(n^2 \log n)$ respectively. Table 2 compares the computational time needed by the different approaches.

Hounsfield used the least squares approach in his first successful experimental inversion. Already with a sample as coarse as 8×8, the necessary computing took hours on EMI's then state-of-the-art ICL machine. The code least_sq is a direct implementation of this approach. Figure 1.5.3 shows the result if our test object is scanned with 31 rays and 40 angles, followed by reconstruction with this method to a 31×31 grid. This level of resolution is too low for a good reconstruction, but we can still see a rough version of the ring and the central lettering. With the iterative solver least_sq_iter we can easily afford a 63×63 inversion on a standard PC. The result of this inversion is shown in Figure 1.5.3

Current medical imaging uses the filtered BP-method, to be described next.

1.6. Back Projection method.

The idea of back projection is conceptually very straightforward, it is easy to implement, and the computational cost is moderate. In its most direct form, the

Comparisons of times for algorithms of different complexity on a system performing 10^8 floating point operations per second

Size $n \times n$ pixels	n^6 (least squares)	n^4 (least squares)	n^3 (filtered BP)	$n^2 \log n$ (Fourier)
$n = 100$	3 h	1 s	0.01 s	0.2 ms
$n = 1000$	320 y	3 h	10 s	0.03 s

TABLE 2. Comparison of computational efficiencies for some non-iterative inversion methods.

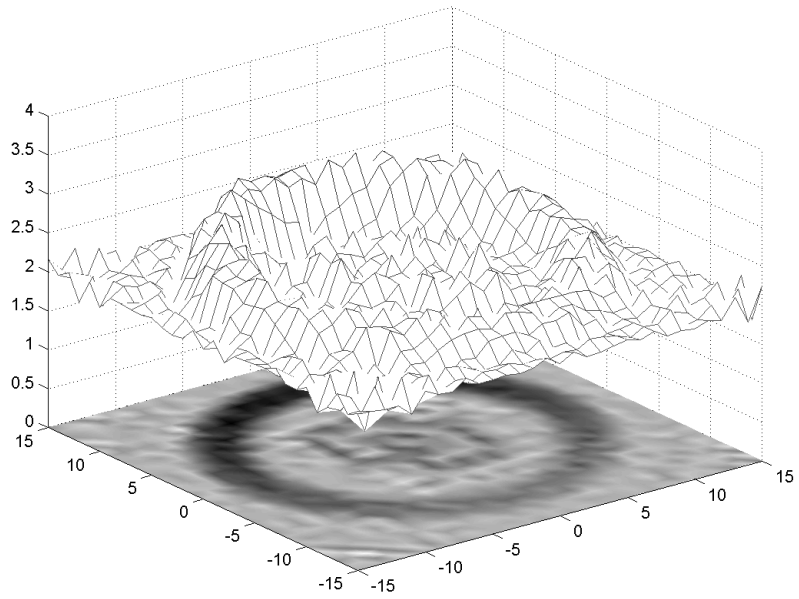


FIGURE 1.5.3. Reconstruction with the least squares method when the logo was scanned with 31 rays at 40 angles, followed by a reconstruction to a 30×30 grid.

reconstruction comes out 'smeared'. However, the addition of a simple filter all but resolves this. It is of little surprise that Filtered Back Projection (FBP) has become the most widely used reconstruction process in the medical community where it very well meets the requirements placed on it.

For an $n \times n$ image reconstruction, the FBP method will cost $O(n^3)$ operations. In many contexts, this cost is acceptable or rather, it became accepted at a time when the major alternatives were even more costly. The computing time using FBP is quite fast in comparison to the other tasks involved such as patient handling etc. However, in an application such as micro-tomography, the situation is very different. There the resolution can be as high as 1000 simultaneously recorded slices, each to be imaged on a 1000×1000 grid, giving a 3-D 1μm resolution throughout a cubic millimeter sample. Each full inversion for such a cube of using back projections would cost on the order of 10^{12} operations. This is likely to become a slow process in comparison with the rapid one of automated sample handling where data for the 1000 slices are collected simultaneously by using a 2-D rather that a 1-D array of X-ray sensors. We describe in Section 1.7 an inversion algorithm which cuts this cost by some orders of magnitude—in this case to about 10^{10} operations.

1.6.1. Immediate back projection. Figure 1.6.1(a) shows a point-type object, and 1.6.1(b) its scan data. Let us remind ourselves how the scan data is obtained. At each angle, the total absorption of each ray is recorded; we do not have any information about the contribution from any specific location. The most obvious thing to do is to assume that all locations along the ray contributes an equal amount. As illustrated in Figure 1.6.2 the simplest form of back projection consists of drawing parallel bands across the image area, with the darkness of the band corresponding to the absorption that was recorded for each ray. Mathematically back projection is described by

$$h(x, y) = \int_0^\pi g(r, \theta) d\theta. \tag{1.1}$$

The right part of Figure 1.6.1 shows the result of this process in this case of the point object. The only error is that the point has turned into a 'smeared out' cone-type mound. The sharp edge of the original point object has been lost, as areas near the point are also covered by some of the bands. However, the position of the recovered mound is precisely the same as that of the original point-object. Also, the amplitude and shape of the mound is position invariant—it takes the same values wherever the original point object was located.

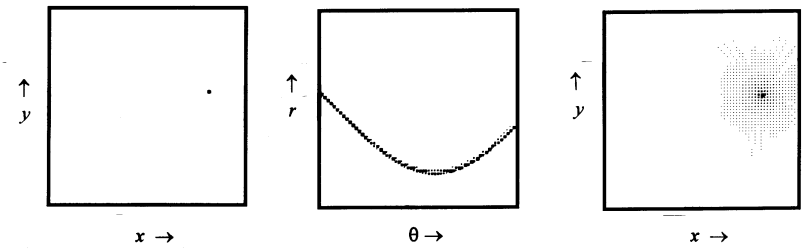


FIGURE 1.6.1. Point-type object, its scan data, and the image recovered through immediate back projection.

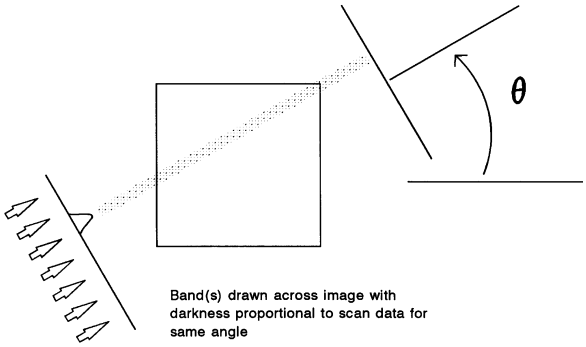


FIGURE 1.6.2. Principle behind back projection (when applied immediately to scan data - no filtering) shown here in the case of a point object.

To appreciate the significance of this example with a point object, we need to note that both the scanning and the back projection phases are *linear*. If there had been two point objects, the scan data would just have been the *sum* of the scan data for the two objects if recorded independently. Similarly, the back projection produces in that case a result which is the sum of the back projections of the two objects, if they were treated separately. Finally, the darkness of the back projected result of each object by itself is proportional to the original darkness. The process of scanning followed by back projection satisfies the two criteria for a function (here

a matrix-valued function with a matrix input) to be *linear*:

$$\begin{aligned} f(x+y) &= f(x)+f(y) \\ f(\alpha x) &= \alpha\, f(x). \end{aligned}$$

From this linearity follows the important conclusion that the reconstruction must also work for full images and not just point objects.

In summary, wherever the imaged object x had a gray pixel, the image f will feature a smeared one centered at the same location, and with a darkness proportional to the darkness of the one in the original. From the linearity follows that if x was a sum of two images with a different gray pixel in each, f will become the sum of the two corresponding images. Continuing this observation: Since every image is a combination of pixels, this linearity implies that f must become a (smeared) representation of the original object. Immediate back projection using the scan data shown in Figures 1.6.1 leads to the reconstruction seen in Figure 1.6.3. The original object is recovered but smeared out. In the next section we discuss a simple method of improving the situation.

1.6.2. Filtered back projection. Comparing the original object in Figure 1.6.1 with Figure 1.6.3 (both featuring the same grid density of 63×63 points), we clearly see a loss in sharpness. Hence, we look for some way to enhance the output from direct back projection in order to reduce the smearing. The step from original object to scan data is essentially outside our control (dictated by X-rays and physics). Options remaining include

- Based on the smeared image, apply some filter which sharpens all gradients (such filters can for example be based on FFTs), and
- Since the cause of the smearing is understood (for the point image), try to alter the scan data in a way that off-sets the back projection smearing.

Both options above are viable; filtered back projection pursues the second one. Two key questions become:

(1) Does there exist any special type of (simulated) scan data for which the back projection method will give a nearly point like result?

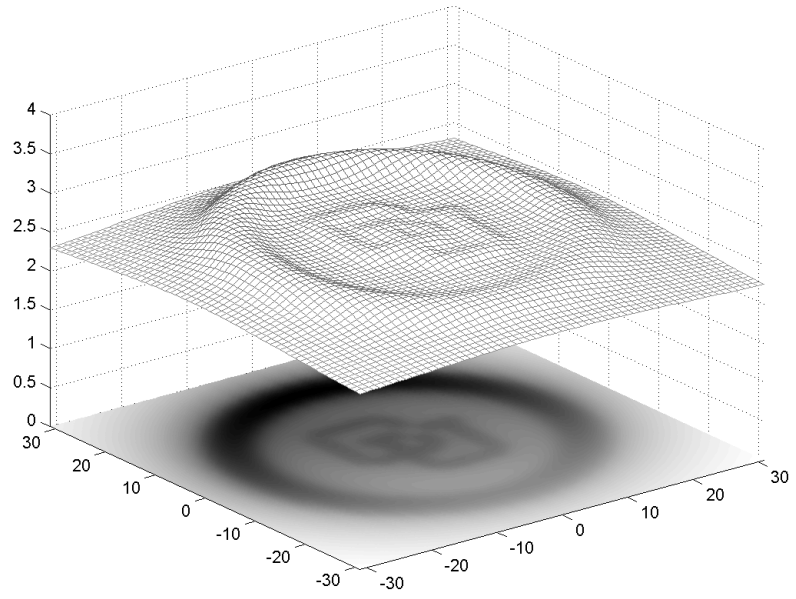


FIGURE 1.6.3. Immediate back projection of the scan data for the test object.

(2) Is there any operator—linear, location preserving, and not altering the darkness at the point itself—that we can apply to turn the actual scan data for the point-type test object into the form that we looked for in point 1 above?

Addressing the first issue, we note that we can replace the 'single-hump' data by a 'hump' at the same place, but with a bright band on each side of it. The contributions for all the angles will still superimpose to an equally dark spot at precisely the desired location, but at nearby locations, the bright sidebands might just cancel some of the undesired darkness, as the contributions for different angles θ are superimposed.

To turn the scan data for a fixed angle, $(0,0,...,0,1,0,...,0)^T$, into a vector with a bright (negative) entry on each side of the 'one'—wherever it is located—can be achieved by multiplying the scan data from the left by a symmetric $n \times n$ tri-diagonal

Toeplitz band matrix

$$E = \begin{bmatrix} 1 & -\beta & & & & & & \\ -\beta & 1 & -\beta & & & & & \\ & -\beta & 1 & -\beta & & & & \\ & & \ddots & \ddots & \ddots & & & \\ & & & \ddots & \ddots & \ddots & & \\ & & & & -\beta & 1 & -\beta & \\ & & & & & -\beta & 1 & -\beta \\ & & & & & & -\beta & 1 \end{bmatrix} \tag{1.2}$$

Applying this idea to the test object, Figure 1.4.1, the results are shown in Figure 1.6.4. To get these figures, we multiplied every single scan data vector with this matrix E, before back projecting where of course, we exploit the sparse structure of E.

This modification preserves

- location and strength of images of point objects—they will appear less smeared, and
- linearity, meaning that a general image will be as good as is the treatment of point objects. Thus linearity also implies that an optimal value of β should be obtainable from considering a point object, a question to be returned to in Section 1.8.

Choosing β= 0.3, 0.4, 0.5 and 0.6 respectively in (1.2) gives, with the scan data of our test object, the reconstructions that are shown in Figures 1.6.4. In spite of the filter being narrow (tri-diagonal; wider filters could be 'tuned' better) and the optimization of the filter coefficient being crude—we simply pick the best-looking of the four cases, we get excellent inversions.

1.7. Fourier transform method.

The Fourier transform (FT) method requires considerably more mathematical background than did the previous two methods. Computationally, it is the fastest known method. In the exercises we shall see how one can mathematically derive from this method both the filtered back projection method and Radon's original inversion

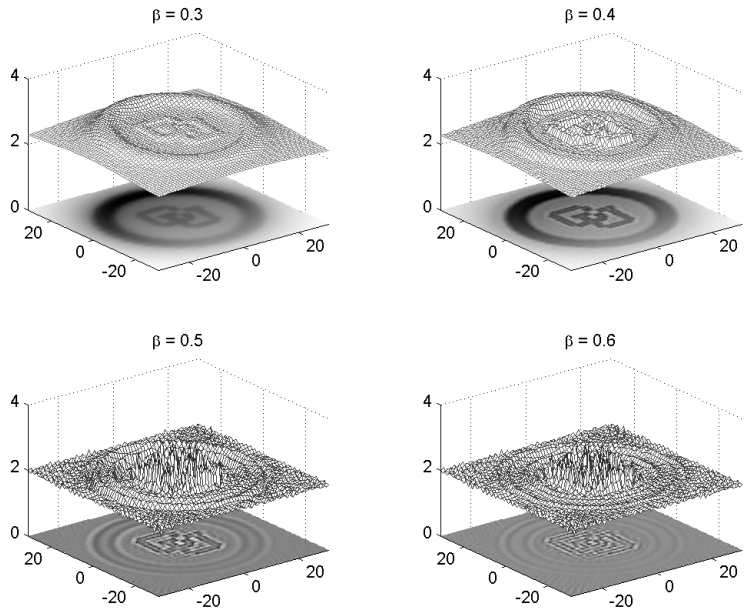


FIGURE 1.6.4. Filtered back projection with some choices of simple tri-diagonal filters.

formula (which, as we noted before, is mathematically compact but numerically impractical).

1.7.1. Analytical description. We assume as before that the density of the object is represented by a density function $f(x, y)$ where x and y denote the two spatial directions. The 2-D Fourier transform of the density function is given by

$$\hat{f}(\omega_x, \omega_y) = \frac{1}{(2\pi)^2} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y)\, e^{-i\omega_x x}\, e^{-i\omega_y y}\, dx\, dy. \tag{1.1}$$

The density function $f(x, y)$ can then be recovered by

$$f(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \hat{f}(\omega_x, \omega_y)\, e^{i\omega_x x}\, e^{i\omega_y y}\, d\omega_x\, d\omega_y \tag{1.2}$$

We will give two descriptions of how one can arrive at the FT method. The most heurisic one follows below. A more concise but less intuitive approach is given in the exercise section.

The key step is to turn the scan data into $\hat{f}(\omega_x, \omega_y)$. This rests on two observations:

- Noting what happens if we send in the X-rays in a direction parallel to the x-axis: The left part of Figure 1.7.1 illustrates how we obtain the scan data $g(y) = \int_{-\infty}^{\infty} f(x,y)dx$. Its 1-D Fourier transform is

$$\begin{aligned}\hat{g}(\omega_y) &= \frac{1}{2\pi}\int_{-\infty}^{\infty} g(y)\, e^{-i\omega_y y}dy \\ &= \frac{1}{2\pi}\int_{-\infty}^{\infty}\left[\int_{-\infty}^{\infty} f(x,y)dx\right] e^{-i\omega_y y}dy \\ &= \frac{1}{2\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-i0x}e^{-i\omega_y y}dxdy \\ &= 2\pi\hat{f}(0,\omega_y),\end{aligned}$$

  i.e. we have obtained $\hat{f}(\omega_x, \omega_y)$ along a vertical line through origin in the (ω_x, ω_y)-plane (cf. Figure 1.7.1(b)).
- Considering the difference if the X-rays would have entered from another direction. If the X-rays had entered from an angle θ (Figure 1.7.3(a)), the collected scan data will be exactly the same as if instead, the object had been turned an angle of $-\theta$ (Figure 1.7.3(b)). As is shown in Section 7.3 on Fourier transforms, turning an object turns its Fourier transform through exactly the same angle. Therefore, we obtain in this case values for $\hat{f}(\omega_x, \omega_y)$ along the line through the origin in the (ω_x, ω_y)-plane orthogonal to the X-ray direction in the physical plane (Figure 1.7.3(c), (d)). When we have taken X-ray images for $0 \le \theta < \pi$ (and applied the 1-D Fourier transform to them), we have actually obtained $\hat{f}(\omega_x, \omega_y)$ along all radial lines through the origin, i.e. throughout the complete (ω_x, ω_y)-plane. Thus we find that

$$\hat{f}(\omega_r, \theta) = \frac{1}{(2\pi)^2}\int_{-\infty}^{\infty} g(r,\theta)e^{-i\omega_r r}dr. \tag{1.3}$$

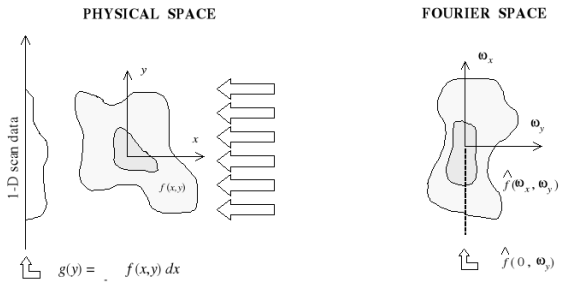


FIGURE 1.7.1. Recording with rays parallel to the x-axis, and the corresponding data set in Fourier space.

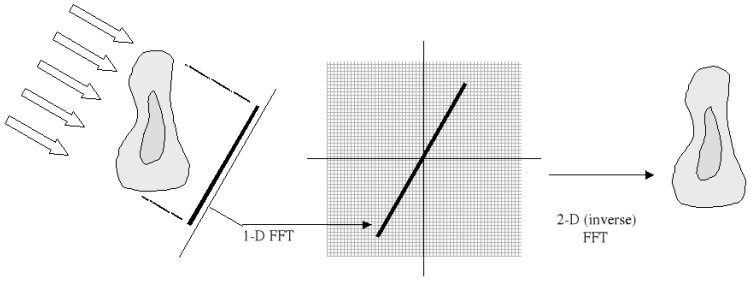


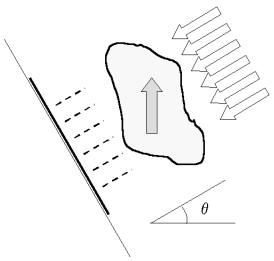
FIGURE 1.7.2. Schematic illustration of the steps in the Fourier reconstruction method.

The density function $f(x, y)$ is then recovered by the 2-D Fourier transform (1.2).

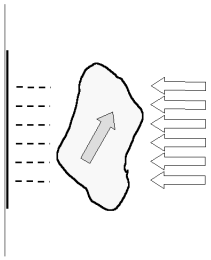
1.7.2. Numerical results with FT method. Following the description above gives a reconstruction as shown in Figure 1.7.4.

Although the details in the center is near-perfect, we see a quite disturbing wobble in the base level of the reconstruction. The FT method that is implemented in the Matlab code therefore contains one more refinement. Each scan vector is 'padded' to double length by just adding zeros at each end of it. Once brought to Fourier space, it is laid out on a correspondingly enlarged 2-D (ω_x, ω_y)-plane. After returning (by

a. Scan with incoming X-rays at an angle θ. The density of the object is $f(x, y)$.



b. Equivalent recording as in a. The image is now rotated through an angle θ, and the X-rays enter horizontally.



c. The $1D$ Fourier Transform of the recording of part b provides a line of data along the ω-axis of the $2D$ Fourier Transform.

d. Since the FT rotates through an angle θ when the image rotates through the same angle, we have now obtained a function $\widehat{f}(\omega_x, \omega_y)$ along a line in the (ω_x, ω_y) plane, sloping the same way as the scan line of part a.

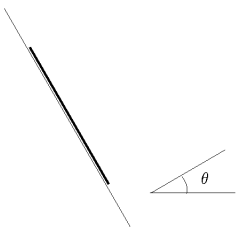


FIGURE 1.7.3. Fundamental principle behind the FT method for tomographic reconstruction.

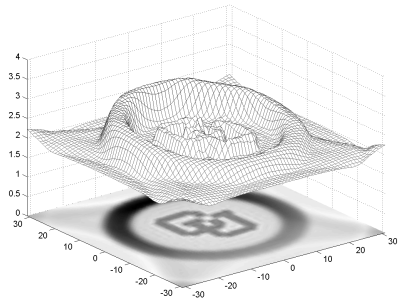


FIGURE 1.7.4. Reconstruction by direct FT method from 63 ray, 64 angle scan data to a 64×64 grid.

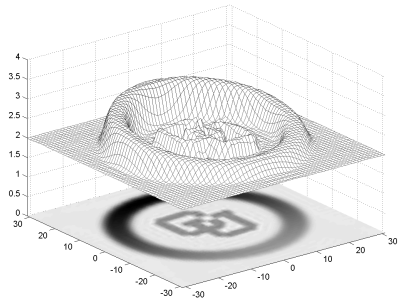


FIGURE 1.7.5. Same Fourier reconstruction as in the previous figure, but with the spatial domain 'padded' by a factor of two within the FT algorithm.

the inverse 2-D FFT) to the (x,y)-plane, we keep only the central square; i.e. disregard the borders (containing 3/4 of the total reconstructed area—these borders only contain an image of the padding areas). The resulting picture is seen in Figure 1.7.5.

There are a couple of ways to understand why this padding idea helps:

- Very heuristically: We are using periodic FFTs instead of infinite-domain transforms. Periodic images of the object are then present near the boundaries of the shown domain. Discrepancies between the concept of periodicity in polar- and in Cartesian coordinates cause a difficult-to-analyze error pattern. From this loose argument, one might expect that padding would improve the image by increasing the distances to unphysical ghost images.

- More theoretically: Extending the spatial domain by a factor of two means that in each direction a twice as *dense* set of Fourier modes become available. For example, in a 1-D spatial domain of [-π, π], modes

$$\dots e^{-3ix},\ e^{-2ix},\ e^{-ix},\ e^{0ix},\ e^{1ix},\ e^{2ix},\ e^{3ix}, \dots$$

are available. If the domain is extended to $[-2\pi, 2\pi]$, also the intermediate modes

$$\dots e^{-\frac{5}{2}ix},\ e^{-\frac{3}{2}ix},\ e^{-\frac{1}{2}ix},\ e^{\frac{1}{2}ix},\ e^{\frac{3}{2}ix},\ e^{\frac{5}{2}ix} \dots$$

become present. The interpolation from polar to Cartesian grids occurs in Fourier space and with a denser grid in that space, interpolation becomes more accurate.

It is important to use better than-linear-interpolation and our code uses cubic interpolation. We can illustrate this in 1-D by trying to represent a half-integer mode $e^{i(n+\frac{1}{2})x}$ on a grid in Fourier space that only has integer modes e^{ikx}, $k = \dots, -3, -2, -1, 0, 1, 2, 3, \dots$ available. Linear (second order) interpolation gives

$$e^{i(n+\frac{1}{2})x} \approx \frac{1}{2}e^{inx} + \frac{1}{2}e^{i(n+1)x} = e^{i(n+\frac{1}{2})x}\ \left(\frac{e^{-i\frac{1}{2}x} + e^{i\frac{1}{2}x}}{2}\right) = e^{i(n+\frac{1}{2})x}\ \cos\frac{x}{2}$$

and fourth order interpolation (cf. Table 2 of Section 12.5)

$$e^{i(n+\frac{1}{2})x} \approx -\frac{1}{16}e^{i(n-1)x} + \frac{9}{16}e^{inx} + \frac{9}{16}e^{i(n+1)x} - \frac{1}{16}e^{i(n+2)x} = e^{i(n+\frac{1}{2})x}\ (\frac{9}{8}\cos\frac{x}{2} - \frac{1}{8}\cos\frac{3x}{2})$$

The interpolation would have been perfect, had the factors (referred to as the damping factors) multiplying $e^{i(n+\frac{1}{2})x}$ in the RHSs in the two equations above been equal to one. Figure 1.7.6 displays the actual factor for interpolation of different orders. Note that this factor depends on x only, i.e. not on n. Interpolation by order 4 and above achieves excellent results in the center of the domain. Hence, we can expect good reconstruction where, in the padded case, the object is located.

Although a factor two padding leads internally to a larger grid, the operation count still remains $O(n^2 \log n)$, only the proportionality constant has increased around a factor of four.

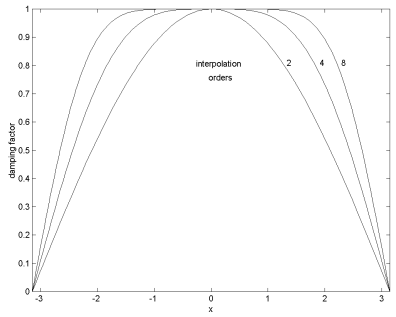


FIGURE 1.7.6. The damping factor at different physical locations across the domain when a half-integer Fourier mode is interpolated *in Fourier space* to integer frequencies.

1.8. Filtered BP method derived from the FT method.

The Fourier inversion method is exact—assuming continuous functions and a doubly infinite domain. The 'raw' BP method gave a quite smeared reconstruction, but it became very good after applying an empirically found filter to each of the scan data vectors. Although we arrived at the two methods—BP and FT—by quite different arguments, they ought to be somehow related.

A little bit of notation to start with: With the coordinate axes as shown in Figure 1.4.5, we can write the scan data function as

$$g(r,\theta) = \int_{-\infty}^{\infty} f(x,y)ds \tag{1.1}$$

where $x = x(s,r,\theta),\ y = y(s,r,\theta)$. Immediate back projection, as shown in Figure 1.6.2, gives a reconstruction

$$h(x,y) = \int_0^{\pi} g(r,\theta)\, d\theta \tag{1.2}$$

with $r = r(x,y,\theta)$. The result of the immediate back projection was shown in Figure 1.6.3. Although it is a reasonably good recovery, it is clear that $h(x,y) \neq f(x,y)$. In the next section, we will show that if we replace $g(r,\theta)$, as produced by (1.1), with

$$g(r,\theta) \;\rightarrow\; \frac{1}{(2\pi)^2}\int_{-\infty}^{\infty}\left[\int_{-\infty}^{\infty} g(\rho,\theta)\, e^{-i\omega_r\rho}d\rho\right]\, |\omega_r|\; e^{i\omega_r r}d\omega_r \tag{1.3}$$

$$= \frac{1}{2\pi}\int_{-\infty}^{\infty}\hat{g}(\omega_r,\theta)|\omega_r|e^{i\omega_r r}dr \tag{1.4}$$

before substituting into (1.2), we get $h(x,y) = f(x,y)$—i.e. exact reconstruction.

Note that (1.4) is the Fourier transform of the product $\hat{g}(\omega_r,\theta)|\omega_r|e^{i\omega_r r}$. Writing $|\omega_r|$ as the Fourier transform of a function that is to be determined at the end of this section, we can interpret (1.4) as a particular filter applied to the function $g(r,\theta)$.

There is another way to express (1.4) that is mathematically equivalent:

$$g(r,\theta) \;\rightarrow\; \frac{1}{2\pi^2}\int_{-\infty}^{\infty}\frac{\partial g(\rho,\theta)/\partial\rho}{r-\rho}\,d\rho. \tag{1.5}$$

This expression was found by J. Radon in 1917. It is mathematically very elegant, shorter than (1.4), and superficially looks simpler (just a single integral), but turns out to be far less practical for computational use. Derivatives are usually more difficult to approximate well than integrals. Also, the integral has a 'principal value' singularity at $\rho = r$ which adds computational difficulty.

1.8.1. Derivation of replacement formula for $g(r,\theta)$. Both the FT method as well as (1.4) are exact. It is therefore natural to suppose that (1.4) can be derived from the FT method. Starting from

$$f(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\hat{f}(\omega_x,\omega_y)e^{i\omega_x x+i\omega_y y}d\omega_x d\omega_y$$

we change to the scan data coordinates (1.2) (see also Figure 1.4.5),

$$\omega_x = -\omega_r\sin\theta,\;\; \omega_y = \omega_r\cos\theta,\;\; \frac{d(\omega_x,\omega_y)}{d(\omega_r,\theta)} = \omega_r,$$

to obtain

$$f(x,y) = \int_0^{2\pi}\int_0^{\infty}\hat{f}(\omega_r,\theta)e^{-i\omega_r\sin\theta x+i\omega_r\cos\theta\, y}\omega_r\; d\omega_r d\theta.$$

If we now alter the description of the standard polar domain $0 \le \omega_r < \infty,\ 0 \le \theta \le 2\pi$ to $-\infty < \omega_r < \infty,\ 0 \le \theta \le \pi$ and recalling from Section 1.4 that $-x\sin\theta + y\cos\theta = r$, it follows that,

$$f(x,y) = \int_0^{\pi} \int_{-\infty}^{\infty} \hat{f}(\omega_r, \theta)\ e^{i\omega_r r}\ |\omega_r|\ d\omega_r d\theta.$$

Finally, the key result of the FT method (1.3) gives

$$f(x,y) = \frac{1}{(2\pi)^2} \int_0^{\pi} \left\{ \int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} g(r,\theta) e^{-i\omega_r r} dr \right] |\omega_r|\, e^{i\omega_r r} d\omega_r \right\} d\theta.$$

Comparing this with (1.2) and (1.4), we see that (1.4) is established.

1.8.2. Interpretation of the replacement formula as a filter. We have just shown that the back projection method would be exact if each scan vector was modified according to (1.4) before being actually used for back projecting. The formula (1.4) amounts, independently for each θ, to a *convolution* of g with some yet undetermined function whose Fourier transform in the r-direction is $|\omega_r|$. Simplifying our notation a bit ($r \to x,\ \omega_r \to \omega$), we need to ask ourselves what function $e(x)$ has the Fourier transform $|\omega|$, i.e.

$$|\omega| = \frac{1}{2\pi} \int_{-\infty}^{\infty} e(x)\ e^{-i\omega x}\ dx$$

or

$$e(x) = \int_{-\infty}^{\infty} |\omega|\ \ e^{i\omega x}\ d\omega \tag{1.6}$$

This first attempt runs into a roadblock—the integral (1.6) is divergent.

The integral is divergent because it is defined on an infinite interval. In practical work, we only consider finite intervals. So one way to gain insight into what (1.6) is 'trying to tell us' is to discretize it and look at the DFT of the function $|\omega|$. We can then see if there is any clear pattern emerging as $N \to \infty$. We choose N even and

for frequency:	$-\frac{N}{2}+1$	$-\frac{N}{2}+2$	$\cdots$	-2	-1	0	1	2	$\cdots$	$\frac{N}{2}-2$	$\frac{N}{2}-1$	$\pm\frac{N}{2}$
enter value:	$\frac{N}{2}-1$	$\frac{N}{2}-2$	$\cdots$	2	1	0	1	2	$\cdots$	$\frac{N}{2}-2$	$\frac{N}{2}-1$	$\frac{N}{2}$

After having normalized the FFT output by multiplying it with $\frac{4}{N^2}$ we get

$N = 32$	...	$-.0176$	0.0	$-.0464$	0.0	$-.4066$	1.0000	$-.4066$	0.0	$-.0464$	0.0	$-.0176$	...
$N = 64$	...	$-.0165$	0.0	$-.0454$	0.0	$-.4056$	1.0000	$-.4056$	0.0	$-.0454$	0.0	$-.0165$	...
$N = 128$	...	$-.0163$	0.0	$-.0451$	0.0	$-.4054$	1.0000	$-.4054$	0.0	$-.0451$	0.0	$-.0163$	...
.....													
$\lim_{N\to\infty}$	...	$-.0162$	0.0	$-.0450$	0.0	$-.4053$	1.0000	$-.4053$	0.0	$-.0450$	0.0	$-.0162$	...
$=$	...	$-\left(\frac{2}{5\pi}\right)^2$	0	$-\left(\frac{2}{3\pi}\right)^2$	0	$-\left(\frac{2}{1\pi}\right)^2$	1	$-\left(\frac{2}{1\pi}\right)^2$	0.0	$-\left(\frac{2}{3\pi}\right)^2$	0.0	$-\left(\frac{2}{5\pi}\right)^2$	...

The three entries

$$-0.4053 \quad 1.0000 \quad -0.4053$$

very much dominate the other entries. We have arrived theoretically at just the same type of filter as was empirically proposed in Section 1.6. The value $\beta \approx 0.4$ we used there is indeed nearly optimal.

1.9. Exercises.

We recall from Section 1.4 that given the density function $f(x, y)$ of a 2-D object, the scan data can be written

$$g(r, \theta) = \int_{-\infty}^{\infty} f(x, y)\, ds \tag{1.1}$$

where the rotated coordinate axes are as defined through

$$\begin{cases} s = x\cos\theta + y\sin\theta \\ r = -x\sin\theta + y\cos\theta \end{cases} \qquad \begin{cases} x = s\cos\theta - r\sin\theta \\ y = s\sin\theta + r\cos\theta \end{cases} \quad \cdots$$

- Determine the scan data produced by the function $f(x, y) = \begin{cases} 1 & x^2 + y^2 \le 1 \\ 0 & \text{otherwise} \end{cases}$

  Answer: $g(r, \theta) = \begin{cases} 2\sqrt{1 - r^2} & |r| \le 1 \\ 0 & \text{otherwise} \end{cases}$.

- Determine the scan data produced by the function $f(x, y) = \begin{cases} 1 & \max(x, y) \le 1 \\ 0 & \text{otherwise} \end{cases}$

  Answer: Define $\phi = \text{mod}(\theta + \frac{\pi}{4}, \frac{\pi}{2}) - \frac{\pi}{4}$ (suffices to solve for $0 \le \theta \le \frac{\pi}{4}$; remaining θ-intervals reflections of this one). Then

$$g(r,\theta) = \begin{cases} \frac{2}{\cos\phi} & |r| \le \cos\phi - \sin\phi \\ \frac{\cos\phi + \sin\phi - r}{\cos\phi\ \sin\phi} & \cos\phi - \sin\phi < |r| \le \cos\phi + \sin\phi \\ 0 & \text{otherwise} \end{cases} .$$

- Verify from the definition (1.1) that $g(r,\theta)$ is a *linear* function of $f(x,y)$.

  Hint: Write down the two requirements for linearity, and test these on $g(r,\theta)$.

- We let R denote the Radon transform operator, i.e. $Rf(x,y) = g(r,\theta)$.

  a. Show that $R\,f(\alpha x, \alpha y) = |\alpha|\ g(\alpha r, \theta)$.

  b. Show that if we change independent variable $\begin{pmatrix} \xi \\ \eta \end{pmatrix} = A \begin{pmatrix} x \\ y \end{pmatrix}$ where A is a 2×2 matrix; $B = A^{-1}$, then $R\,f(\xi,\eta) = |\det B|\ g(...,...)$.

- Show that for $g(r,\theta) = \begin{cases} 1 & |r| \le 1 \\ 1 - \frac{|r|}{\sqrt{r^2-1}} & \text{otherwise} \end{cases}$ immediate back projection leads to a reconstruction $h(x,y) = \begin{cases} 1 & x^2 + y^2 \le 1 \\ 0 & \text{otherwise} \end{cases}$.

CHAPTER 2

Facial Recognition

2.1. Introduction

How can one tell whether the criminal who has just been convicted of a crime, is a first– or a habitual offender? This is a most serious question since the sentence depends on the answer. The habitual offender can of course expect a harsher sentence and will do everything possible to hide his/her real identity. This was exactly the situation in Europe during the second half of the nineteenth century. There was no reliable system in place to identify individuals and the police had to rely almost entirely on personal recognition. People were often misidentified—with sometimes disastrous consequences. A case in point, as late as 1896, Adolf Beck was misidentified as John Smith, a conman and repeat offender, and sentenced to seven years in prison. The fact that according to descriptions, John Smith had brown eyes while Beck had blue eyes, that Smith was circumcised and Beck not, made no difference—it was ascribed to administrative error. Just too many witnesses were willing to swear that Beck was indeed the perpetrator. It was only after Beck's release that John Smith was arrested on a charge of hoaxing two actresses out of their rings that the full sorry tale was revealed, see [**?**].

A reliable personal identification system was clearly long overdue.

At that time two rival personal identification systems were being developed. William James Herschel, not to be confused with his grandfather, the eminent astronomer, also William Herschel, was experimenting with hand prints, and a little later with fingerprints in India. But the most influential individual in the development of fingerprints as a personal identification system was Henry Faulds. His discovery was accidental. Because of his anthropological interests, he was in the habit of collecting fingerprints of his students and friends, his collection soon numbered in the thousands. At about this time he noticed that the supply of medical

alcohol at his hospital started to run inexplicably low. When he then discovered a cocktail glass in the form of a laboratory beaker with an almost complete set of fingerprints, the culprit was promptly identified. This was exactly the spark that was needed, although it took at least another 20 years before fingerprints were widely accepted as forensic evidence.

The second system was developed in France during the 1870's by one Alphonse Bertillon. His system was the result of frustration and bitterness over a mindlessly boring and futile job. He was required to write police descriptions into the five million police files gathering dust in their massive archive. A typical description would say, 'Stature: average', 'Face: ordinary'. These descriptions were obviously totally useless. He then hit on the idea of measuring an individual. Maybe if one takes enough measurements that total set would be unique for an individual. His system employed eleven separate measurements: height, length and breadth of head, length and breadth of ear, length form elbow to end of middle finger, lengths of middle and ring fingers, length of left foot, length of the trunk, and length of outstretched arms from middle fingertip to middle fingertip. Apart from being to able distinguish between different individuals, it also allowed a classification system that enabled Bertillion to quickly locate the file of a criminal, given only the measurements. The system was so successful that France was one of the last countries to adopt fingerprints for personal identification, see [**?**].

Our modern technological society relies heavily on personal identity verification, be it to gain access to bank accounts or secure areas, or just to login on a computer. Automated identity verification is a complex problem and many different options have been pursued. Some old favorites, including

- Personal Identification Numbers (PIN)
- Passwords
- Identity documents, etc

are easy to copy and encourage fraud. The problem is that these systems do not contain any personal information about the individual. Passwords and PIN's are of course totally divorced from the individual using it—it is impossible to verify that the person offering the password is in fact authorized to use it. It is not surprising that

fraudulent transactions based on the misuse of identification systems have become a most serious problem.

In recent years one therefore finds an increasing move away from systems relying on something the individual *own or know* as a means of identification, to systems recognizing something the individual *is*. Thus much effort has gone into the development of automated personal identification/verification systems based on the characteristics unique to each individual, the so-called Biometric Personal Identification Systems. Ideally these system are based on personal characteristics that even the individual is not able to alter (disguise). A number of biometric identification systems are already commercially available. It is possible to login onto your computer using your fingerprint or go through immigration after a retina scan has establish you identity. Dynamic signature verification is totally dependent on the participation of the individual and is ideal in situations where one has to endorse a transaction; more of this in Chapter 22.

Facial recognition on the other hand, is a passive system requiring no participation from the individual. No wonder that it is becoming increasingly popular for surveillance systems. For humans it is also the most natural identification system available. It is indeed difficult to imagine a world without the human face as we know it: flat (i.e. no muzzle), hairless and with its characteristic features, eyes, protruding nose, mouth and chin. Yet the true human face appeared only about 180 000 years ago with Homo sapiens in Africa. The human face is a most remarkable object. It houses four of the five senses (sight, smell, taste and touch) and we learn from the earliest infancy to rely on it for identifying each other. Equally important is its use for communication. It might be argued that the human face has evolved to its present form for no other reason than to improve communication. Indeed, facial hair hides expression and the fact that the facial muscles are directly attached to skin, allows for an infinity of facial expressions, reflecting an infinitude of emotional subtleties. No wonder that the human face has held such a fascination for artists through all the centuries. Some of the greatest works of art convey such a complex of emotions that it defies description. For some the smile of the Mona Lisa by Leonardo da Vinci is ‘divinely pleasing’, someone else believes she is flirting and yet another senses a ‘hateful haughtiness’. See [**?**] for a fascinating discussion of the human face.

Serious scientific studies of the visual qualities of the human face date back at least to the same Leonardo da Vinci who made detailed studies of the interaction of light and face and recently of course, it has become the object of intensive scientific study. Although the scientist may have very different goals, its fascination with the geometric structure, the interaction of this structure with light, its moods and expressions, is no less keen than that of the artist. For it is exactly these very human characteristics that provide a person his/her visual individuality—one of the major concerns of the biometric scientist.

A number of different ideas have been developed for automated facial recognition , see for example [**?**]. In this chapter we concentrate on systems based on the so-called eigenface technique. The basic idea, first introduced by Sirovich and Kirby [**?**], has subsequently been developed into some of the most reliable facial recognition systems available. In particular, the eigenface-based system developed at the Media Laboratory at MIT ([**?**],[**?**] and [**?**]) has consistently performed among the best in the comprehensive FERET tests [**?**].

Eigenfaces are derived from a carefully constructed set of facial images, the training set. The training set is a substitute for all the facial images the system is expected to encounter and should therefore represent the characteristics of all relevant facial images. The aim of the eigenface approach is to distill these characteristics in the form of the eigenfaces. The idea is simple: find an orthonormal basis for the subspace spanned by the images in the training set, easily achieved through the Singular Value Decomposition (SVD). The power of this approach lies in the fact that the facial images in the training set lie inside a low dimensional subspace of the general image space. This subspace is identified through the nonzero, or very small, singular values. The eigenfaces are the orthonormal basis elements associated with the remaining nonzero singular values. Assuming that the training set is really representative of all faces, the eigenfaces therefore form a low dimensional, orthonormal basis for the linear subspace containing all facial images (note that we try to formulate this very carefully—faces themselves do not form a linear subspace). Any facial image can be orthogonally projected onto the eigenfaces with the result that any particular face is represented by its projection coefficients. In a facial recognition system the

similarity of the projection coefficients of different faces is used to decide whether two images are from the same individual or not.

In this very basic description of a facial recognition system based on eigenfaces important practical issues have been ignored. For example, experiments by Pentland and co-workers, see [**?, ?**], show that the efficiency of the system is improved significantly if a comparison of global, eigenface expansions is augmented by a local eigenfeature expansion, consisting for example of the eyes, noses and mouths. Since these 'local' features are also compared by expanding in their 'eigenfeatures', the ideas described in this chapter also apply to these situations.

2.2. An Overview of Eigenfaces.

The idea behind the eigenface technique is to extract the relevant information contained in a facial image and represent it as efficiently as possible. Rather than manipulating and comparing faces directly, one manipulates and compares their representations.

Assume that the facial images are represented as 2D arrays of size $m = p \times q$. Obviously, m can be quite large, even for a coarse resolution such as 100×100, $m = 10,000$. By 'stacking' the columns, we can rewrite any $m = p \times q$ image as a vector of length m. Thus, we need to specify m values in order to describe the image completely. Therefore, all $p \times q$ sized images can be viewed as occupying an $m = pq$-dimensional vector space. Do facial images occupy some lower dimensional subspace? If so, how is this subspace calculated?

Consider n vectors with m components, each constructed from a facial image by stacking the columns. This is the training set and the individual vectors are denoted by $\mathbf{f}_j$ where $j = 1, \ldots, n$. Obviously, it is impossible to study every single face on earth, so the training set is chosen to be representative of all the faces our system might encounter. This does not mean that all *faces* one might encounter are included in the training set, we merely require that all faces are adequately *represented* by the faces in the training set. For example, it is necessary to restrict the deviation of any individual face from the average face. Some individuals may be so unique that our system simply cannot cope. See [**?, ?**] for a detailed analysis of issues relating

to training sets. Obviously, the training set must be developed with care. Also, typically $n \ll m$.

As mentioned above, one should ensure that the faces are normalized with respect to position, size, orientation and intensity. All faces must have the same size, be at the same angle (upright is most appropriate) and have the same lighting intensity, etc, requiring some nontrivial image processing (see [**?**]). Assume it is all done.

Since all the values for the faces are nonnegative and the facial images are well removed from the origin, the average face can be treated as a uniform bias of all the faces. Thus we subtract it from the images as will be explained in more detail in Section 11.4. The average and the deviations, also referred as the caricatures, are

$$\mathbf{a} = \frac{1}{n}\sum_{j=1}^{n} \mathbf{f}_j, \tag{2.1}$$

$$\mathbf{x}_j = \mathbf{f}_j - \mathbf{a}. \tag{2.2}$$

We illustrate the procedure using the Surrey database [**?**]. This consists of a large number of RGB color images taken against a blue background. Using color separation it is easy to remove the background and the images were then converted to gray-scale by averaging the RGB values. A training set consisting of 600 images (3 images of each of 200 individuals) was constructed according to the Lausuanne Protocol Configuration 1 [**?**]. We should point out that the normalization was done by hand and is not particularly accurate as will become evident in the experiments. Figure 2.2.1 shows three different images of two different persons—the first two images are of the same person where the first image is part of the training set. The second image is not inside the training set, taken on a different day and one should note the different in pose as well as facial expression. The third image is of a person not in the training set.

We need a basis for the space spanned by the $\mathbf{x}_j$. These basis vectors will become the building blocks for reconstructing any face in the future, whether or not the face is in training set. In order to construct an orthonormal basis for the subspace spanned by the faces in the training set, we define the $m \times n$ matrix X with columns

FIGURE 2.2.1. Sample faces in the database.

corresponding to the faces in the training set,

$$X = \frac{1}{\sqrt{n}} \left[\begin{array}{cccc} \mathbf{x}_1 & \mathbf{x}_2 & \dots & \mathbf{x}_n \end{array} \right], \tag{2.3}$$

where the constant $\frac{1}{\sqrt{n}}$ is introduced for convenience. The easiest way of finding an orthonormal basis for X is to calculate its Singular Value Decomposition (SVD), as explained in detail in Section **??**. Thus we write

$$X = U\Sigma V^T.$$

For an $m \times n$ dimensional matrix X, U and V orthonormal matrices with dimensions $m \times m$ and $n \times n$, respectively. Σ is an $m \times n$ diagonal matrix with the non-negative singular values, $\sigma_j,\ j = 1, \dots, \min(m, n)$ arranged in non-decreasing order on its diagonal. If there are r nonzero singular values then the first r columns of U form an orthonormal basis for the column space of X. In practice one can also discard those columns associated with very small singular values. Let us say we keep the first ν columns of U where $\nu \leq r$ and $\sigma_{\nu+1}$ is regarded to be sufficiency small so that it can be discarded.

In Figure 2.2.2, we plot the normalized singular values—normalized so that the largest singular value is one—for our training set of facial images. Typically these eigenvalues decrease rapidly. In this case the 150th singular values is already quite small. One would therefore expect a rather good representation using the first 150 columns of U, i.e. $\nu \approx 150$. These basis vectors, $\mathbf{u}_j,\ j = 1, \dots, \nu$, are the eigenfaces. Some of them are shown in Figure 2.2.3. Note how the higher eigenfaces have more detail added to them. This is in general the case—the lowest eigenfaces contain

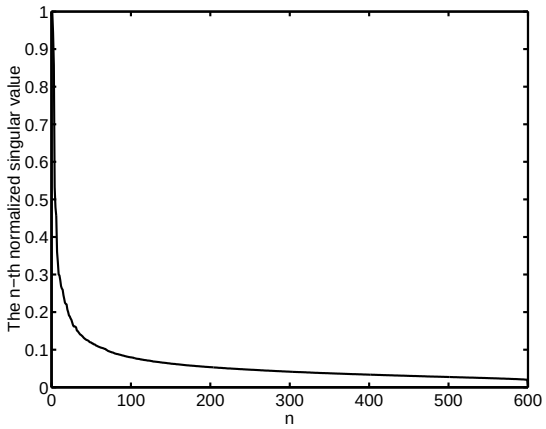


FIGURE 2.2.2. The normalized singular values.

general face-like information with the detail required to distinguish an individual provided by the higher order eigenfaces. For a more detailed explanation see [**?**, **?**]. In this case one should point out that the almost complete lack of structure of the first eigenface is an indication that our training set is not well normalized.

If our training set were representative, all facial images are in a 150 dimensional linear subspace. However, the faces in our training set were selected arbitrarily and not optimized to be as widely representative as possible. More importantly, our normalization is crude and we'll describe a simple experiment indicating just how sensitive the system is to the normalization. One should also realize that for a automated facial recognition system, visually perfect reconstruction is not required. Reconstructions that include sufficient information to distinguish between different individuals suffice. Estimates in the literature range from 40 to about 100 eigenfaces based on experiments with a heterogeneous population (see [**?**]), indicating that faces are described very efficiently in terms of eigenfaces—the effective dimension of the linear vector space containing the facial images is of order 100 (rather than $m = pq$) for a general $p \times q$ image.

One might ask how important it is to subtract the average $\mathbf{a}$. Imagine that the facial images are clustered around the average face, far from the origin. The first eigenface indeed points in the direction of the average face. The problem is that the rest of the eigenfaces are constrained to be orthogonal to this rather arbitrary

(a) Eigenface 1

(b) Eigenface 10

(c) Eigenface 50

(d) Eigenface 100

FIGURE 2.2.3. Eigenfaces number 1, 10, 50, and 100.

direction, reducing their ability to describe the distribution of the faces clustered around the average face. Experiments show that if the average is not removed, the first eigenface is indeed closely related to the average, but the singular values decrease slightly slower.

2.3. Calculating the eigenfaces.

One striking feature of the calculation of the SVD is the relative sizes of m and n. Because $m = pq$ it can easily become very large, the number n of images in the training set is typically one or two orders of magnitude smaller. The question arises whether one can exploit this fact for numerical purposes. This turns out to be a little tricky. The main problem is that one has to be very careful to ensure that numerical errors do not destroy the orthogonality of the orthogonal matrices. For example one might be tempted to follow the procedure prescribed by Fukunaga [**?**, p39] and first calculate the reduced V_+ from the $n \times n$ symmetric eigenvalue problem (11.2), using any of the efficient methods available for calculating the eigensystem of symmetric matrices, including the QR algorithm, Rayleigh quotient iteration, divide and conquer methods, etc. See [**?, ?**] for more details. Although there is already a loss in accuracy by computing $X^T X$ (see Section 11.4 for more detail), it is the next step where one can go seriously wrong. For example, it is tempting to calculate U_+ from

$$XV_+ = U_+\Sigma_+, \tag{2.1}$$

i.e. $U_+ = XV_+\Sigma_+^{-1}$. This is not a good idea. Note, for example that round-off error destroys the orthogonality of the columns of U_+.

In order to exploit the fact the $n << m$ in a numerically stable way, we suggest the following procedure. Given an $m \times n$ matrix X with $n << m$, we use of another of the great matrix factorizations, namely the QR factorization with column permutations (the basis of the QR algorithm mentioned above), to calculate $XP = QR$ where Q is an $m \times m$ matrix with orthonormal columns and R is an $m \times n$ upper triangular matrix. P is a permutation matrix rearranging the columns of X such that the diagonal entries of R are in non-increasing order of magnitude. Since the last $m - n$

rows of R consist of zeroes, we can form the reduced QR factorization by keeping the first n columns of Q and the first n rows of R. Thus we obtain $X = Q_+R_+$ where R_+ is an $n \times n$ upper triangular matrix. The next step is to calculate the SVD of $R_+ = U_R\Sigma V^T$. Thus we get, $XP = (Q_+U_R)\Sigma V^T$, or $XP = U\Sigma V^T$ with U the product of two orthonormal matrices, $U = Q_+U_R$.

2.4. Using Eigenfaces

In the previous section we discussed the reasons why faces can be efficiently represented in terms of eigenfaces. To obtain the actual representation is quite straightforward. The idea is to project any given face orthogonally onto the space spanned by the eigenfaces. More precisely, given an face $\mathbf{f}$ we need to project $\mathbf{f} - \mathbf{a}$ onto the column space of $U_\nu := [\mathbf{u}_1 \cdots \mathbf{u}_\nu]$. This means that we wish to solve

$$U_\nu \mathbf{y} = \mathbf{f} - \mathbf{a} \tag{2.1}$$

in a least squares sense. Since the columns of U_ν are orthonormal, it follows that the eigenface *representation* is given by

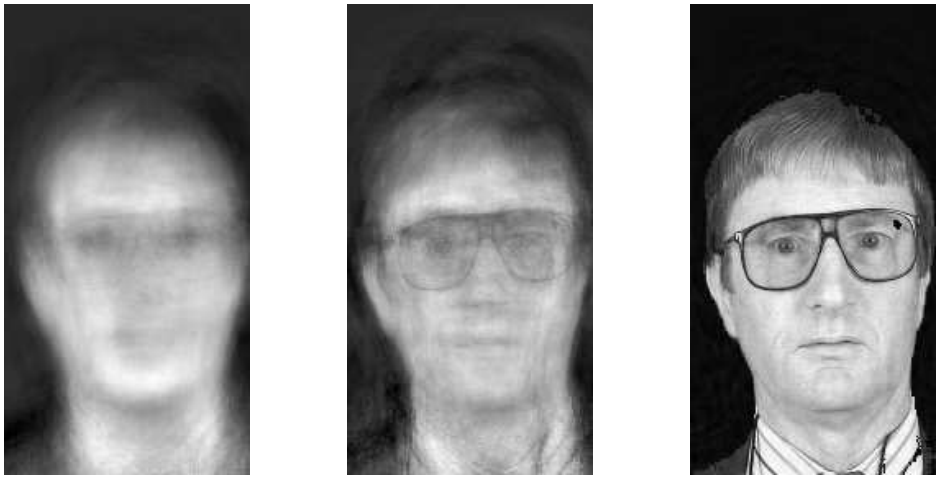
$$\mathbf{y} = U_\nu^T(\mathbf{f} - \mathbf{a}). \tag{2.2}$$

This representation captures all the features associated with the face $\mathbf{f}$ and instead of comparing faces directly, we rather compare features.

The eigenface *reconstruction* of $\mathbf{f}$ is given by

$$\widetilde{\mathbf{f}} = U_\nu \mathbf{y} + \mathbf{a}. \tag{2.3}$$

The eigenface reconstruction of the first face in Figure 2.2.1 (the face in the training set) is shown in Figure 2.4.1, using 40, 100 and 450 eigenfaces. The root mean square (rms) errors of the three reconstructions—the 2-norm of the difference between the original and the reconstruction, or equivalently, the magnitude of the first neglected singular value—are given by 6742, 4979 and 753. Certainly up to 100 eigenfaces, the reconstruction is visually not particularly good, despite the fact the



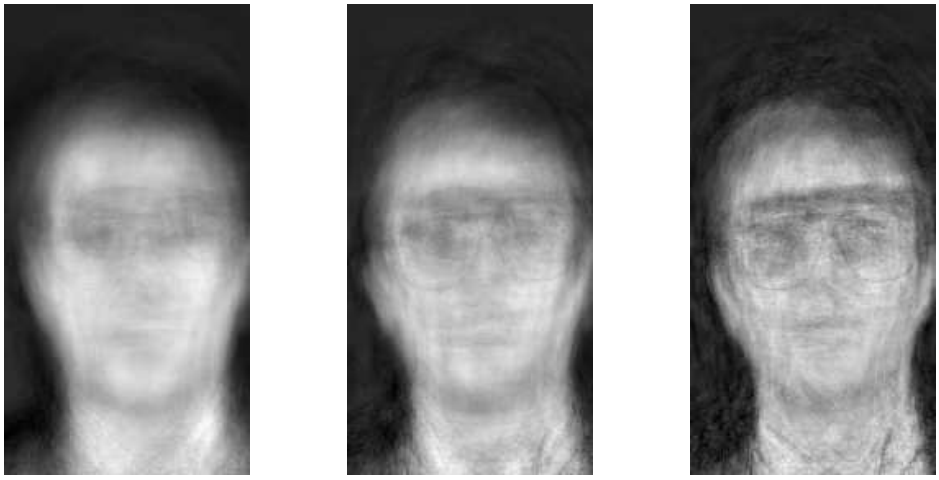
(a) 40 Eigenfaces (b) 100 Eigenfaces (c) 450 Eigenfaces

FIGURE 2.4.1. The reconstruction of a face in the training set.

the singular values (rms values) are already relatively small. The 2-norm is clearly not a good norm in which to measure visual correspondence. It is only for rms values less than about 1000 that one finds good visual reconstruction.

We do the same for the second image of the first person, not in the training set (the second image of Figure 2.2.1), and the results are shown in Figure 2.4.2. Although the result is not visually pleasing, the important question is whether enough information is available to recognize the individual.

In Figure 2.4.3 we show the reconstruction of the third face of Figure 2.2.1, the face not in the training set, again using 40, 100 and 450 eigenfaces. Although the reconstruction is visually not particularly good, the individual is already recognizable using 100 eigenfaces. In fact this reconstruction is better than that of Figure 2.4.2. The reason is that most of the images in the training set are facing the camera directly. Thus, although not in the training set, the individual with a similar pose are better reconstructed in Figure 2.4.3 than the individual in the training set but with a pose not represented in the training set.



(a) 40 Eigenfaces (b) 100 Eigenfaces (c) 450 Eigenfaces

FIGURE 2.4.2. The reconstruction of a face not in the training set.

(a) 40 Eigenfaces (b) 100 Eigenfaces (c) 450 Eigenfaces

FIGURE 2.4.3. The reconstruction of a person not in the training set.

The final experiment demonstrates the sensitivity of the system to the normalization. In this experiment we shifted the first image of Figure 2.2.1 two pixels down from its original position. Figure 2.4.4 shows its reconstruction using 450 eigenfaces. The result is clearly a significant deterioration of the visually good reconstruction we obtained in Figure 2.4.1.

FIGURE 2.4.4. Reconstruction of a non-normalized face.

Up to this point the gray-scale images have been 2D representations of the interaction of light with a 3D structure. This poses several difficulties. For example, since no 3D information is available it is hard to correct for essentially 3D distortions such as out-of-plane rotations. We had more problems reconstructing the image with out-of-plane rotation of a person inside the training set, than of a person not in the training set, but directly facing the camera. Of course facial expression also plays a important role and is not easily corrected for. Gray-scale images are also heavily dependent on the strength and direction of the light source and it is not easy to compensate for different lighting conditions [**?**]. Working with gray-scale images where the different shades of gray represent 3D structure directly remove these problems. This idea is pursued in Chapter 32. Provided a simple camera model is used, the reconstruction again relies on the SVD.

CHAPTER 3

Global Positioning Systems

3.1. Introduction.

The history of navigation is one of the longest and most important quests in the evolution of civilization. With GPS, technology has at last provided a near-perfect solution, now freely available for everybody to use. GPS receivers have quickly become indispensable, not only by professionals such as seamen or pilots, but also to hikers and basically to anybody enjoying the outdoors. The accuracy of present small hand-held units is of the order of 10—15 meters after *selective availability* — degrading the precision for non-military users—was eliminated in 2000. One can the for example use these to record the position where one leaves a car in a big outdoor car park. Used simply as clocks, they providing local time to better than 10^{-6} seconds—far higher accuracy of course than is ever needed in everyday life. Together with *differential correction* and a *phase locking* technique, accuracies can be as high as one millimeter, potentially making all other surveying techniques obsolete, both for global (e.g. piloting) and local (e.g. construction site) usage.

Section 3.2 recalls very briefly the history of navigational devices, from ancient days until the breakthrough of GPS. Section 3.3 summarizes the principles of GPS. We formulate in Section 3.4 a test problem and then solve it in two different ways. This is followed by some error analysis in Section 3.5 and a discussion of *pseudo-random sequences* in Section 3.6.

3.2. A Brief History of Navigation.

Early man observed the sun and the stars, and presumably used these for navigation long before leaving any written records about it. As late as the Viking age (800-1100 AD), little further help was available for navigation on open seas. Rough estimates of the Polar star's height over the horizon, together with 'dead reckoning'

(a phrase originating from 'deduced reckoning'; estimating distances based on course, currents, winds and speeds), did not always suffice to find the intended destinations. Innumerable marine disasters have been caused by navigational errors.

> One particularly gruesome incident occurred on a foggy night in October, 1707 when a group of four British warships with about 2,000 men on board ran aground just off the English SW coast. Only two men reached shore. One happened to be the fleet commander, who was promptly murdered upon reaching shore by a local woman for a ring he was wearing. Maybe there was some justice in this. The fleet officers knew full well before the accident that their navigation had been faulty; nevertheless, a seaman who had kept a perfect log, and dared to very carefully and respectfully offer this to an officer the day before—knowing the risk but hoping to help avoid a disaster—was immediately hanged for insubordination. This particular incident was a contributing factor to a British competition that about half a century later finally led to a successful means to determine longitude at sea—latitude is much easier—and hence to much safer sea travels.

The concept of longitudes and latitudes goes back at least to Ptolemy. All 27 sheets of his world atlas from 150 AD have such lines drawn, together with a separate list of coordinates for all its named locations. The equator was on his atlas marked as the zeroth parallel (latitude) and the Canary Islands defined the zero meridian (longitude). This latter choice was quite arbitrary, and indicative of the coming difficulties in determining the longitude at sea. Before settling at Greenwich, 'prime meridians' were at times placed at the Azores, Cape Verde Islands, Rome, Copenhagen, Jerusalem, St. Petersburg, Pisa, Paris, Philadelphia and many other places as well.

The big advances in navigation from the days of the Vikings have been

- discovery of the compass,
- finding the longitude,
  (latitude can be read off easily from the height of stars—e.g. the pole star—over the horizon),
- navigation by radio beacons (LORAN), and

- GPS.

The first compasses were simple chunks of loadstone (magnetite, a common iron ore) which tend to orient themselves in a fixed direction, when suspended freely (by a string, or floated in a container of water). Their first documented use for navigation occurred in the Mediterranean during the 12th century. Magnetic compasses remain to this date indispensable on all ships, at the very least as a navigational back-up device. Gyro-compasses work by a completely different principle—a suitably suspended rapidly rotating disc will keep its axis aligned with that of the earth. These compasses will always point to true north, and are insensitive to variations in the magnetic field (which can be due to geological anomalies or electrical storms on the sun). Although far more complicated than magnetic compasses, they are nowadays used in most larger ships and aircraft, often in connection with 'inertial guidance' devices that compute changes in positions from sensed accelerations.

The lack of any reliable means for determining the longitude at sea caused great hazards for sea travels until the chronometer was developed in the second half of the 18th century. If it was not possible to simply follow coastlines which can be dangerous, especially at night and in bad weather, it was common practice to try to reach ports by first finding the appropriate latitude, and then follow it until the destination appeared in sight. This procedure was not very satisfactory for several reasons

- it works less well for coastlines facing north or south, as opposed to east or west,
- when aiming for a small island, the approach to the desired latitude had to be quite far off to the east or to the west in order to leave no ambiguity about the direction to finally proceed in,
- it forced sailing ships to follow paths that might not be suitable with regard to shoals, winds and currents, and
- it offered opportunities for pirates to lie in wait out at sea at the latitudes of main harbors.

The main competing approaches for finding the longitude all required that the local time which is easily available by the position of the sun, be compared to the simultaneous time at some fixed reference location such as Greenwich. Ideas to determine this reference time included

- Observing the position of our moon relative to the sun and the stars. Newton's law of gravity was discovered first in 1684, and the complicated orbit of the moon—a quite non-circular path influenced by both the earth and the sun—could not be predicted with enough precision until well after the whole approach had been made obsolete by the chronometer.
- Observations of Jupiter's moons. Since their orbits could be tabulated accurately, the moons can serve as an accurate clock in the sky. Eclipses when a moon disappears in the shadow of the planet, happens roughly every two days for each of the inner moons and are near-instantaneous events, allowing very accurate time readings. Although this worked very well on land (for example to determine the location of islands), even in good weather it proved to be utterly impractical at sea.
- The chronometer—basically an accurate clock, designed to be insensitive to motions and changes in temperature, humidity and gravity. This became the winner in the longitude competition. John Harrison's produced a series of increasingly accurate chronometers, culminating in 1760 with the pocket-sized 'H-4'. On its first sea trial—UK to Jamaica, arriving in January 1762—it lost only 5 seconds. This corresponds to an error of only 2 km after 81 days at sea. However, this was somewhat lucky—an error of about one minute or 24 km could have been expected; even that a vast improvement over other methods. By 1780, chronometers were starting to come in wide use throughout the British and other navies. These were often privately purchased by the Captains, as official navy channels still were slow in providing them.

More exotic ideas at the time included

- Placing light-ships at known strategic locations. These would then every-so-often send up a rocket that exploded brightly—deemed to be visible at

night for up to 60 to 100 miles, providing travelers within that range with a time signal, and

- Mapping the vertical inclination of the earth's magnetic field. Lines of equal inclination would generally intersect the lines of constant latitude (or the angle could be mapped), thus together with the latitude providing complete positional information. However, not only does the earth's magnetic field change slowly with time, it can also fluctuate dramatically with solar activities, up to about 10 degrees—enough to cause positional uncertainties as wide as an ocean.

The history of how the longitude problem got resolved though the chronometer recounted in many books; a recent one being 'Longitude' by Sobel [**?**]. Even then, navigation was not always easy. After the loss of his ship, the Endurance, in the pack-ice of the Antarctic, the famous British explorer, Ernst Shackleton, and his whole crew ended up on Elephant island, one of the most remote spots on earth. This was April 1916, Europe was at war and no one in any case had the faintest idea of the critical situation of the Shackleton expedition. The nearest help was at the South Georgia islands, about 800 miles across some of the stormiest oceans imaginable. Their most seaworthy vessel was the James Caird, an open lifeboat, totally unsuitable for the task ahead. With no other options left, Shackleton and a small crew, including the navigator and captain of the Endurance, Frank Worseley, sailed from Elephant island on April 24, 1916 on their 800 mile journey. Due to bad weather Worseley was forced to navigate mainly through dead-reconing. As they approached the South Georgia islands, the situation became critical. In the words of Worseley [**?**]:

On the thirteenth day we were getting nearer to our destination. If we made the tragic mistake of passing it we could never retrace our way on account of the winds and the currents, it therefore became essential that I should get observations. But the morning was foggy, and if you cannot see the horizon it is impossible to measure the altitude of the sun to establish your position. Now, the nearer your eye is to the surface of the sea, the nearer is the horizon. So I adopted the expedient of kneeling on the stones in the bottom of the boat, and by this means succeeded in taking a rough

> observation. It would have been a bold assumption to say that it was a correct one; but is was the best I could do, and we had to trust it. Two observations are necessary, however, to fix your position, and my troubles were far from over; for at noon, when I wanted to observe our latitude, I found conditions equally difficult. The fog, which before had been on a level with us and therefore did not altogether obscure the sun, had now risen above us and was hovering between the sun and ourselves, so that all I could see was a dim blur. I measured to the centre of this ten times, using the mean of these observations as the sun's altitude.
>
> With serious misgiving I worked out our position and set my course by it to sight South Georgia, near King Haakon Sound, the next day.
>
> Thus, against all odds, the whole expedition was saves without the loss of a single man.

The first radio-based navigation technique amounted to determining the direction to a known transmitter by rotating a direction-sensitive antenna. Much higher precision was offered by a series of systems known as OMEGA, DECCA, GEE and LORAN (long range navigation). These were developed around the time of World War II. By the timing difference in arrivals of radio signals from a 'master' and a 'slave' transmitter which re-transmitted the master signal the moment it received it, a ship could locate itself along a specific curve, In the 2-D plane case, it is a hyperbola which is the curve with constant difference from two points. By also receiving signals from another transmitter pair, the ship could determine its location from the intersections of the two curves. This system gave a typical accuracy of around 1 km and a useful range of about 1000 km at daytime, and about double that at night time. Radio navigation systems were the first ones that could give positional fixes in any weather conditions.

The GPS idea is to have a number of satellites in orbit, each transmitting both its orbital data and very accurate time pulses. A receiver can then time the arrivals of the incoming time pulses. Knowing the speed of light, the distances to the satellites can be found (c = 299,792,458 m/s in vacuum). This is exact and serves since 1983 as the definition of the meter based on an existing definition of the second. From knowing their orbits, the receiver's position can be found. With the high velocity

of the satellites (and the high speed of light!), the demands on the precision of the equipment are extreme.

> The cesium or rubidium clocks in the GPS satellites operate at 10.22999999545 MHz rather than the nominal 10.23 MHz to compensate for both the special relativity effect of a moving source and the general relativity effect of operating from a point of higher gravitational potential. The master clock at the GPS control center near Colorado Springs is set to run 16 ns a day fast to compensate for its location 1830 m above sea level.

The military's need for the system was also extreme—it was developed towards the end of the cold war as a means of accurately guiding ICBMs. Hence, it is hardly surprising that there are today two parallel fully operational systems in place, one created by the US Department of Defense and one by its Soviet counterpart. The cost for getting the GPS systems operational was staggering—at least 12 B$ (i.e. $12{\cdot}10^9$$) for the US system. The fact that both systems now are available to the general public, without any charge, is almost as impressive as their technical capabilities. With low cost handheld receivers (around 100 dollars), anyone can now determine his/her position to better than 100 meters at any time, in any weather, at any point on earth. With the best, and much more expensive, receiving equipment available, that can be improved to an amazing 1 mm in both horizontal and vertical coordinates. Surprisingly, GPS is still not used routinely in aviation (in 2000)—possibly because neither of the two signal providers is officially committed to providing uninterrupted public service. For most civil and private usage, this concern is far outweighed by its practical advantages.

3.3. Principles of GPS.

Table 1 summarizes some technical specifications for GPS and GLONASS. These American and Russian systems are very similar in most respects. Before concentrating on GPS, let us note one difference: All the GPS satellites broadcast on exactly the same frequency (in order to save bandwidth); their transmission of separate pseudo-random (PR) sequences—described in Section 3.6—allows this without causing any signal confusion. Two GLONASS satellites exactly opposite each other use

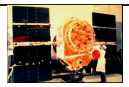
	GPS	GLONASS
		
Operated by	US DOD (Department of Defense)	Russia
Control center	Falcon Air Force Base (near Colorado Springs, CO)	?
First satellite launched	1978	1982
System operational	1993	1993
Satellite constellation		
Number of satellites	$24 + 3$ spares	$24 + 2$or 3 spares
Satellite distribution	4 spaced 90^o apart in 6 planes	8 spaced 45^oapart in 3 planes
Orbital inclination to equatorial plane	55^o(limited by possible orbits of the space shuttle for launching and servicing)	64.8^o
Average elevation (from center of earth)	27,560 km (about 3.0 earth radii above its surface—at the	25,510 km outer edge of upper van Allen belt)
Orbital period	11 h 58 min (one half sidereal day)	11 h 15 m 45 s
Frequency of orbital information update from ground	every hour	every half hour
Radio frequencies (civilian)	1575.42 MHz (Navigational information)	$1602 + n \cdot 0.5625$MHz, $n = 0, 1, \ldots, 12$
Length of pseudo-random code	$1023 = 2^{10} - 1$ bits	$511 = 2^9 - 1$ bits
Chip rate; repeat time of pseudor. code	1.023 MHz; 1.0 ms	0.511 MHz; 1.0 Ms
Data package; rate, length	50 bits/s; 30 s	50 bits/s ; 30 s

TABLE 1. Some technical specifications of GPS and GLONASS

the same frequency—the 24 satellites require therefore 12 separate frequencies. The GLONASS satellites also use a PR sequence, but the same sequence is used by all the satellites.

In this section, we will briefly describe how a position is determined and why we need to receive signals from four satellites for this.

We start with the simplest possible situation, assuming that

(1) The satellites and the receiver are constrained to lie in a 2-D plane, and
(2) Both the satellites and the receiver have perfect clocks.

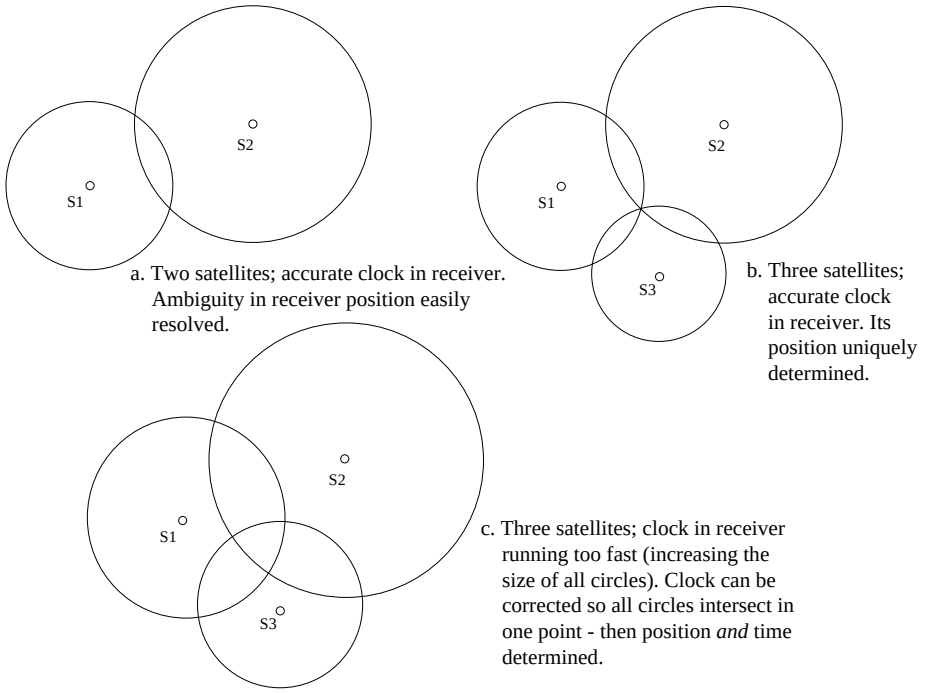


FIGURE 3.3.1. Principle behind how receiver position is calculated in case of 2-D satellite and receiver configuration.

The satellites send out a data package that tells their precise orbits, so their position at any time can be assumed to be perfectly known. They send out their time pulses at exactly known times, and the receiver records accurately when these arrive. Also knowing the speed of light, the receiver can therefore calculate how far it is away from the (known positions) of the satellites. In a 2-D model world, we get a picture as shown in Figure 3.3.1 where we listen in to two satellites. The receiver can be at either of the two places where the circles intersect. In Figure 3.3.1(b), a third satellite is added, and the position becomes uniquely determined.

The assumption of perfect clocks is quite true for the satellites; although their cesium or rubidium clocks have the phenomenal accuracy of up to one part in 10^{13}, they are still corrected from the ground several times a day. The cost and bulk of similar clocks in the receivers would be prohibitive. In reality, the receivers have built-in clocks not much better than a typical wrist watch—the errors can be in the

order of seconds or even minutes. Figure 3.3.1(c) shows what happens if the receiver clock has gone a bit too fast—the receiver would think that all signals had been traveling for a longer time than the actually have. Hence, all the circles will have their radii too large, but all are increased with the same amount. The three circles that should have intersected in one point don't do that any longer. The receiver uses this to calculate a *clock correction*—it determines how much its clock needs to be corrected so the three circles again intersect in one point. After that, it is accurate both in position and in time.

In 3-D, the situation is very similar—only that we need four satellites to determine both position and time, using exactly the same ideas. Two spheres intersect along a circle; a third sphere selects out two possible positions. It takes a fourth satellite to get us a discrepancy allowing the clock to be corrected. So when receiving from four satellites, we can determine both position and time in a 3-D space.

3.4. Test Problem with Numerical Solutions.

With 24 GPS satellites in the sky (not counting spares), as many as 10-12 might be above the horizon at the same time. The orbits are designed so that at least 4 will be in fairly good positions at all times and from any point on earth. Therefore, one is always assured of being able to get a GPS positional fix. To get the best accuracy, it makes sense to utilize information from all satellites that are available. Finding a position usually becomes an over-determined problem ; we have more data that what is minimally needed to get a unique solution.

We will next formulate a numerical test problem, and then discuss two different methods of solving it.

3.4.1. Test problem. We assume that we have six satellites (S1 - S6) and a receiver (R) located as seen in Figure 3.4.1 and described in Table 2.

This data set is 'rigged' so that our answer, receiver position and clock error, all will be integers. This has no significance for any of the algorithms—it just makes the equations shorter to write, and also makes it easier to follow how the convergence in the algorithms is progressing.

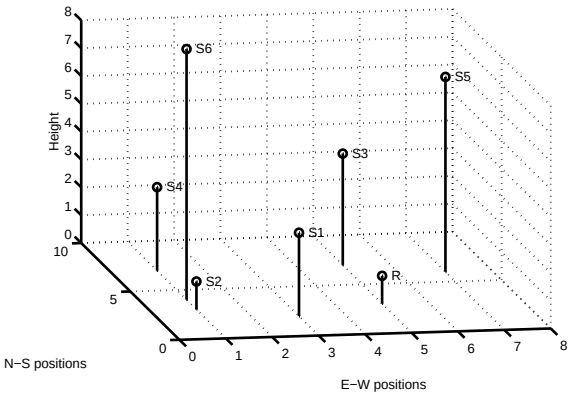


FIGURE 3.4.1. Location of satellites and receiver in the test problem (distance units 1,000 km).

Satellite and receiver position in test problem

GIVEN DATA		
Transmitters (satellites)		Recorded delay (ms) between accurate transmission time and the receive time
Nr	x,y,z - locations (in units of 1,000 km)	according to inaccurate receiver clock
S1	3, 2, 3	10010.00692286
S2	1, 3, 1	10013.34256381
S3	5, 7, 4	10016.67820476
S4	1, 7, 3	10020.01384571
S5	7, 6, 7	10023.34948666
S6	1, 4, 9	10030.02076857
TO BE DETERMINED (4 quantities)		
Receiver location		Clock error
R	5, 3, 1	t = 10,000

TABLE 2. Satellite and receiver locations in test problem

3.4.2. Numerical Solutions. With (x, y, z, t) denoting the unknowns, receiver position and clock error, the nonlinear system to be solved can be written as

$$
\begin{aligned}
(x-3)^2+(y-2)^2+(z-3)^2-[(10010.00692286-t)\cdot c]^2 &= 0\\
(x-1)^2+(y-3)^2+(z-1)^2-[(10013.34256381-t)\cdot c]^2 &= 0\\
(x-5)^2+(y-7)^2+(z-4)^2-[(10016.67820476-t)\cdot c]^2 &= 0\\
(x-1)^2+(y-7)^2+(z-3)^2-[(10020.01384571-t)\cdot c]^2 &= 0\\
(x-7)^2+(y-6)^2+(z-7)^2-[(10023.34948666-t)\cdot c]^2 &= 0\\
(x-1)^2+(y-4)^2+(z-9)^2-[(10030.02076857-t)\cdot c]^2 &= 0
\end{aligned}
\tag{3.1}
$$

where $c = 0.299792458$ (in the unit of 1,000 km/ms). The two numerical methods we will describe below are linearization and Newton's method.

Linearization

The equations (3.1) are nonlinear, but if we expand all the squares, each equation will take the form $x^2+y^2+z^2+t^2c^2+\{\text{linear terms}\}=0$, i.e. if we subtract one of the equations say, the last one, from the rest, all nonlinearities vanish, and we are left with the linear system

$$
\begin{bmatrix}
4 & -4 & -12 & 3.59751\\
0 & -2 & -16 & 2.99792\\
8 & 6 & -10 & 2.39834\\
0 & 6 & -12 & 1.79875\\
12 & 4 & -4 & 1.19917
\end{bmatrix}
\begin{bmatrix} x\\ y\\ z\\ t \end{bmatrix}
=
\begin{bmatrix}
35971.1\\
29957.2\\
24031.4\\
17993.5\\
12059.7
\end{bmatrix}
$$

This overdetermined system can be solved in the least square sense with the methods in Section 11.5, giving

$$x = 5.0000,\quad y = 3.0000,\quad z = 1.0000,\quad t = 10,000\quad .$$

If we have only four satellites visible, the elimination of the nonlinear terms would give us three linear equations in four unknowns. The *echelon form* (Section **??**) allows us to express three of the unknowns in terms of the fourth one. Substituting these expressions into the last one would give us a quadratic equation in the remaining unknown. This quadratic will typically have two solutions, only one of which will correspond to a 'reasonable' position. We can then immediately find the remaining unknowns.

Newton's method

It was a very unusual circumstance that allowed the nonlinear terms in all but one of the equations (3.1) to be eliminated. Linearization—as it occurs in Newton's method—is much more general. It does not rely on any particular coincidences in the structure of the equations, nor do we end up with one equation less to work with. With numerical linearization, the solution process becomes iterative, and we need to provide a starting guess. How close such a guess has to be varies from problem to problem. As we shall see, this is not a difficulty in the present case.

Applying Newton's method, as described in (*to be written*) to (3.1) gives the iteration

$$x_{n+1} = x_n + \Delta x_n,\ y_{n+1} = y_n + \Delta y_n,\ z_{n+1} = z_n + \Delta z_n,\ t_{n+1} = t_n + \Delta t_n,$$

where the updates are obtained from solving the (overdetermined) linear system

$$\begin{bmatrix} 2(x_n-3) & 2(y_n-2) & 2(z_n-3) & 2c^2(10010.00692286 - t_n) \\ 2(x_n-1) & 2(y_n-3) & 2(z_n-1) & 2c^2(10013.34256381 - t_n) \\ 2(x_n-5) & 2(y_n-7) & 2(z_n-4) & 2c^2(10016.67820476 - t_n) \\ 2(x_n-1) & 2(y_n-7) & 2(z_n-3) & 2c^2(10020.01384571 - t_n) \\ 2(x_n-7) & 2(y_n-6) & 2(z_n-7) & 2c^2(10023.34948666 - t_n) \\ 2(x_n-1) & 2(y_n-4) & 2(z_n-9) & 2c^2(10030.02076857 - t_n) \end{bmatrix} \begin{bmatrix} \Delta x_n \\ \Delta y_n \\ \Delta z_n \\ \Delta t_n \end{bmatrix} =$$

$$= - \begin{bmatrix} (x_n-3)^2+(y_n-2)^2+(z_n-3)^2-[(10010.00692286 - t_n)\cdot c]^2 \\ (x_n-1)^2+(y_n-3)^2+(z_n-1)^2-[(10013.34256381 - t_n)\cdot c]^2 \\ (x_n-5)^2+(y_n-7)^2+(z_n-4)^2-[(10016.67820476 - t_n)\cdot c]^2 \\ (x_n-1)^2+(y_n-7)^2+(z_n-3)^2-[(10020.01384571 - t_n)\cdot c]^2 \\ (x_n-7)^2+(y_n-6)^2+(z_n-7)^2-[(10023.34948666 - t_n)\cdot c]^2 \\ (x_n-1)^2+(y_n-4)^2+(z_n-9)^2-[(10030.02076857 - t_n)\cdot c]^2 \end{bmatrix}$$

The next issue is to find a start guess (x_0, y_0, z_0, t_0). Knowing that the travel times for signals cannot be negative, one can for example choose t_0 as the shortest time recorded, i.e. $t_0 = 10010.00692286$. Let us also guess that we are at the location

$x = y = z = 0$. This is an extremely coarse guess; the errors of 5000 and 3000 km in the x- and y-directions are the size of a continent.

The iterations proceed as follows:

n	x	y	z	t
0	0.0	0.0	0.0	10010.007
1	6.368727	0.374601	-2.403971	9985.218
2	4.984063	3.018241	1.046303	10000.266
3	5.000160	2.999808	0.999585	9999.998
4	5.000000	3.000000	1.000000	10000.000

We see the typical signs of quadratic convergence—a doubling of correct digits once the iterations have 'settled in'. Here, convergence to all the precision we want is obtained after just 4 iterations—a common situation when a reasonable guess is available. The numerical errors are at this point reduced to better than 1 m in distance and 1 μs in time.

3.5. Error Analysis.

A recording of a position is not of much use if it is not accompanied by some form of error estimate. There are many sources of errors encountered in the GPS process. Table 3 is a very schematic summary of how much different sources typically contribute to the error in the readings.

We will here carry out one example of error analysis to illustrate the process of tracing how different sources of errors in input data can carry through to errors in the computed position. One key feature this will illustrate is that error analysis generally is *linear*, provided of course that the errors remain small enough; the final effect of different error sources can be studied separately, and effects can be added together for a 'worst case' estimate.

> If there are many measurements available for the same quantity, it often happens that the errors will fluctuate randomly and partly cancel when averaging. Statistical tools should then be applied so as not to get unduly

Typical errors (in meters) in computed distance to each satellite due to different error sources

Source	**Standard GPS**
Satellite clocks[1]	1.5
Orbit errors	2.5
Ionospheric delays[2]	5.0
Tropospheric delays	0.5
Receiver noise	0.3
Multipath	0.6
Typical resulting positional accuracy	
Horizontal	10
Vertical	40

[1] Clock stability: Rubidium 10^{-11}–10^{-12}, cesium $10^{-12} - -10^{-13}$. (Cf. hydrogen maser 10^{-16} and typical watch 10^{-6})

[2] Can be reduced to 0.5 - 1.0 m if receiving on two frequencies; delay proportional to (number of electrons)$/f^2$where f is the signal frequency.

TABLE 3. Summary of error sources

pessimistic 'worst case' errors only. For example, with n estimates, the expected error often decreases like $1/\sqrt{n}$.

We suppose here that we have only the top four equations of the set (3.1) available. We write these in the form

$$
\begin{aligned}
f_1 \equiv\ & (x-3)^2 + (y-2)^2 + (z-3)^2 - [(T_1 + t_1 - t)\cdot c]^2 &= 0 \\
f_2 \equiv\ & (x-1)^2 + (y-3)^2 + (z-1)^2 - [(T_2 + t_2 - t)\cdot c]^2 &= 0 \\
f_3 \equiv\ & (x-5)^2 + (y-7)^2 + (z-4)^2 - [(T_3 + t_3 - t)\cdot c]^2 &= 0 \\
f_4 \equiv\ & (x-1)^2 + (y-7)^2 + (z-3)^2 - [(T_4 + t_4 - t)\cdot c]^2 &= 0
\end{aligned}
\tag{3.1}
$$

where the recorded time delays according to the receiver's very inaccurate clock are

$$
\begin{aligned}
T_1 &= 10010.00692286 \\
T_2 &= 10013.34256381 \\
T_3 &= 10016.67820476 \\
T_4 &= 10020.01384571
\end{aligned}
$$

We have also introduced additional variables t_1, t_2, t_3, t_4 which represent further errors in the timing signals from each of the four satellites. Causes for these errors could for example be ionospheric delays. We want to estimate the uncertainty is in the position x, y, z and corrected time t, as functions of variations in t_1, t_2, t_3, and t_4 .

Simplified one variable / one equation situation:

Had our system of equations been just one scalar equation in one variable

$$f_1 \equiv (x-3)^2 - [(T_1 + t_1 - t)\cdot c]^2 = 0 \tag{3.2}$$

we would first set $t_1 = 0$ i.e. assume this extra error was not there, and solve for x. Next, we would re-introduce t_1 and ask how the variations in t_1 will influence x. Hence, we view x as a function of t_1 : $x = x(t_1)$. Differentiating (13.1) with respect to t_1 gives

$$\frac{df_1}{dt_1} = 2(x-3)\frac{dx}{dt_1} - 2c^2(T_1 + t_1) = 0.$$

Now we again set $t_1 = 0$ and solve for $\frac{dx}{dt_1}$. That derivative is precisely what we want—a measure of how much x will change for small changes in t_1.

Original four variable / four equation situation:

In (3.1), we similarly have $x = x(t_1, t_2, t_3, t_4)$, $y = y(t_1, t_2, t_3, t_4)$, $z = z(t_1, t_2, t_3, t_4)$, $t = t(t_1, t_2, t_3, t_4)$. Differentiating f_i with respect to t_j becomes an exercise in using the chain rule:

$$\frac{df_i}{dt_j} = \frac{\partial f_i}{\partial x}\frac{\partial x}{\partial t_j} + \frac{\partial f_i}{\partial y}\frac{\partial y}{\partial t_j} + \frac{\partial f_i}{\partial z}\frac{\partial z}{\partial t_j} + \frac{\partial f_i}{\partial t}\frac{\partial t}{\partial t_j} + \frac{\partial f_i}{\partial t_j} = 0, \qquad i, j = 1, ..., 4.$$

This is most clearly written in matrix form

$$\begin{bmatrix} \frac{\partial f_1}{\partial x} & \frac{\partial f_1}{\partial y} & \frac{\partial f_1}{\partial z} & \frac{\partial f_1}{\partial t} \\ \frac{\partial f_2}{\partial x} & \frac{\partial f_2}{\partial y} & \frac{\partial f_2}{\partial z} & \frac{\partial f_2}{\partial t} \\ \frac{\partial f_3}{\partial x} & \frac{\partial f_3}{\partial y} & \frac{\partial f_3}{\partial z} & \frac{\partial f_3}{\partial t} \\ \frac{\partial f_4}{\partial x} & \frac{\partial f_4}{\partial y} & \frac{\partial f_4}{\partial z} & \frac{\partial f_4}{\partial t} \end{bmatrix} \begin{bmatrix} \frac{\partial x}{\partial t_1} & \frac{\partial x}{\partial t_2} & \frac{\partial x}{\partial t_3} & \frac{\partial x}{\partial t_4} \\ \frac{\partial y}{\partial t_1} & \frac{\partial y}{\partial t_2} & \frac{\partial y}{\partial t_3} & \frac{\partial y}{\partial t_4} \\ \frac{\partial z}{\partial t_1} & \frac{\partial z}{\partial t_2} & \frac{\partial z}{\partial t_3} & \frac{\partial z}{\partial t_4} \\ \frac{\partial t}{\partial t_1} & \frac{\partial t}{\partial t_2} & \frac{\partial t}{\partial t_3} & \frac{\partial t}{\partial t_4} \end{bmatrix} = - \begin{bmatrix} \frac{\partial f_1}{\partial t_1} & \frac{\partial f_1}{\partial t_2} & \frac{\partial f_1}{\partial t_3} & \frac{\partial f_1}{\partial t_4} \\ \frac{\partial f_2}{\partial t_1} & \frac{\partial f_2}{\partial t_2} & \frac{\partial f_2}{\partial t_3} & \frac{\partial f_2}{\partial t_4} \\ \frac{\partial f_3}{\partial t_1} & \frac{\partial f_3}{\partial t_2} & \frac{\partial f_3}{\partial t_3} & \frac{\partial f_3}{\partial t_4} \\ \frac{\partial f_4}{\partial t_1} & \frac{\partial f_4}{\partial t_2} & \frac{\partial f_4}{\partial t_3} & \frac{\partial f_4}{\partial t_4} \end{bmatrix}$$

Taking partial derivatives of (3.1) gives all the entries of the first and last matrices above

$$2\begin{bmatrix} x-3 & y-2 & z-3 & c^2(T_1+t_1-t) \\ x-1 & y-3 & z-1 & c^2(T_2+t_2-t) \\ x-5 & y-7 & z-4 & c^2(T_3+t_3-t) \\ x-1 & y-7 & z-3 & c^2(T_4+t_4-t) \end{bmatrix} \begin{bmatrix} \frac{\partial x}{\partial t_1} & \frac{\partial x}{\partial t_2} & \frac{\partial x}{\partial t_3} & \frac{\partial x}{\partial t_4} \\ \frac{\partial y}{\partial t_1} & \frac{\partial y}{\partial t_2} & \frac{\partial y}{\partial t_3} & \frac{\partial y}{\partial t_4} \\ \frac{\partial z}{\partial t_1} & \frac{\partial z}{\partial t_2} & \frac{\partial z}{\partial t_3} & \frac{\partial z}{\partial t_4} \\ \frac{\partial t}{\partial t_1} & \frac{\partial t}{\partial t_2} & \frac{\partial t}{\partial t_3} & \frac{\partial t}{\partial t_4} \end{bmatrix} =$$

$$= 2\begin{bmatrix} c^2(T_1+t_1-t) & 0 & 0 & 0 \\ 0 & c^2(T_2+t_2-t) & 0 & 0 \\ 0 & 0 & c^2(T_3+t_3-t) & 0 \\ 0 & 0 & 0 & c^2(T_4+t_4-t) \end{bmatrix}.$$

Writing this as $A\,X = B$, we can solve for X by simply multiplying by A^{-1} from the left, or better still—view this as a linear system of equations with four RHSs and four side-by-side solution vectors. Using the known values for T_i, setting $t_i = 0$ and using our numerical solution $x = 5,\ y = 3,\ z = 1$ and $t = 10,000$ gives

(3.3)

$$\begin{bmatrix} \frac{\partial x}{\partial t_1} & \frac{\partial x}{\partial t_2} & \frac{\partial x}{\partial t_3} & \frac{\partial x}{\partial t_4} \\ \frac{\partial y}{\partial t_1} & \frac{\partial y}{\partial t_2} & \frac{\partial y}{\partial t_3} & \frac{\partial y}{\partial t_4} \\ \frac{\partial z}{\partial t_1} & \frac{\partial z}{\partial t_2} & \frac{\partial z}{\partial t_3} & \frac{\partial z}{\partial t_4} \\ \frac{\partial t}{\partial t_1} & \frac{\partial t}{\partial t_2} & \frac{\partial t}{\partial t_3} & \frac{\partial t}{\partial t_4} \end{bmatrix} = \begin{bmatrix} 0.149896 & -0.349758 & -0.624568 & 0.824429 \\ 0.149896 & 0.249827 & 0.124914 & -0.524637 \\ -0.449689 & 0.749481 & 0.374741 & -0.674533 \\ -0.500000 & 2.166667 & 2.083333 & -2.750000 \end{bmatrix}$$

This tells how sensitive each variable x, y, z, t is to the small errors in the timings t_1, t_2, t_3, t_4 for the signals from the four satellites.

If the timings are all accurate to within 0.1 μs = 0.0001 ms, the worst case errors in the results can be calculated using the formula,

$$\Delta x \approx \frac{\partial x}{\partial t_1}\Delta t_1 + \frac{\partial x}{\partial t_2}\Delta t_2 + \frac{\partial x}{\partial t_3}\Delta t_3 + \frac{\partial x}{\partial t_4}\Delta t_4,$$

to obtain

$$\begin{array}{lll} x\text{-dir} & (0.1499 + 0.3498 + 0.6246 + 0.8244) & \cdot\, 0.1 \text{ km} \quad \approx 195 \text{ m} \\ y\text{-dir} & (0.1499 + 0.2498 + 0.1249 + 0.5246) & \cdot\, 0.1 \text{ km} \quad \approx 105 \text{ m} \\ z\text{-dir} & (0.4497 + 0.7495 + 0.3747 + 0.6745) & \cdot\, 0.1 \text{ km} \quad \approx 225 \text{ m} \\ t\text{-err} & (0.5000 + 2.5000 + 2.0833 + 2.7500) & \cdot\, 0.1\ \mu\text{s} \quad \approx 0.75\ \mu\text{s} \end{array}$$

In this case, the positional error turns out to be largest in the z-direction. The best time the receiver can calculate is about 7.5 times less accurate than the precision of the incoming signals.

The analysis above was based on reception from only the first four of our six satellites, giving us a 4×4 matrix in (3.3) with sensitivity information. In a similar way, we could have analyzed the full 6-satellite case to arrive at

$$\begin{bmatrix} \frac{\partial x}{\partial t_1} & \frac{\partial x}{\partial t_2} & \frac{\partial x}{\partial t_3} & \frac{\partial x}{\partial t_4} & \frac{\partial x}{\partial t_5} & \frac{\partial x}{\partial t_6} \\ \frac{\partial y}{\partial t_1} & \frac{\partial y}{\partial t_2} & \frac{\partial y}{\partial t_3} & \frac{\partial y}{\partial t_4} & \frac{\partial y}{\partial t_5} & \frac{\partial y}{\partial t_6} \\ \frac{\partial z}{\partial t_1} & \frac{\partial z}{\partial t_2} & \frac{\partial z}{\partial t_3} & \frac{\partial z}{\partial t_4} & \frac{\partial z}{\partial t_5} & \frac{\partial z}{\partial t_6} \\ \frac{\partial t}{\partial t_1} & \frac{\partial t}{\partial t_2} & \frac{\partial t}{\partial t_3} & \frac{\partial t}{\partial t_4} & \frac{\partial t}{\partial t_5} & \frac{\partial t}{\partial t_6} \end{bmatrix} =$$

$$= \begin{bmatrix} -0.0362 & -0.1061 & -0.0267 & 0.2221 & -0.4185 & 0.3655 \\ 0.1240 & 0.2097 & -0.1619 & -0.3125 & 0.2007 & -0.0602 \\ -0.0643 & 0.3353 & 0.0169 & -0.0456 & 0.2505 & -0.6213 \\ -0.3616 & 1.1765 & 0.0047 & -0.5127 & 1.4550 & -1.4852 \end{bmatrix}$$

This time, the worst-case errors are notably smaller even though none of the inputs are any more accurate:

$$\begin{array}{ll} x\text{-dir} & \approx 118 \text{ m} \\ y\text{-dir} & \approx 107 \text{ m} \\ z\text{-dir} & \approx 133 \text{ m} \\ t\text{-err} & \approx 0.50\,\mu\text{s} \end{array}$$

The improvement in expected errors is better still—the probability that the sign and size of all errors conspire to create a maximum error situation is far less likely the more independent input variables that enter, i.e. cancellation of errors becomes increasingly likely.

3.6. Pseudorandom Sequences.

Two of the problems that arise in connection with transmitting timing pulses from the satellites are

(1) how to send a very sharp pulse, so that its arrival can be very accurately timed, without needing to use a wide bandwidth (this will be explained in more detail in a moment), and
(2) how to allow all the satellites to transmit on exactly the same frequency without their signals interfering with each other.

One mathematical construct—pseudo-random (PR) sequences—resolves both these problems very nicely. Once we have seen how they resolve the first problem, it will be clear how the solution to the second one follows.

To appreciate the dilemma posed in point 1, we recall the function-Fourier transform pair $f(x) = e^{-\alpha x^2}$, $\widehat{f}(\omega) = \frac{1}{\sqrt{4\pi\alpha}} e^{-\omega^2/(4\alpha)}$(c.f. Table 2 in Section 9.3). If the parameter α is large, the function $f(x)$ will be a sharp spike, allowing its position to be accurately pinpointed (think here of x as time). However, $\widehat{f}(\omega)$ will then have a very broad maximum, i.e. occupy a lot of frequency space (bandwidth). On the other hand, making the parameter α small will make the pulse broad and difficult to accurately determine its position (e.g. center of its peak).

The answer to this dilemma turns out to be that after all one does not need a sharp pulse in order to achieve a fine time resolution—a suitably structured signal of long duration can also achieve a fine time resolution. Each satellite sends its own PR signal as illustrated in Figure 3.6.1; very small up/down-variations in frequency according to a pattern that looks random, but is fixed for each satellite, and repeats periodically after 1023 chip times of 1 μs each. Thus, the whole pattern repeats roughly each ms. During each 1 μs chip time, the carrier (at 1575.42 MHz) goes through about 1,500 cycles—enough to detect which one of the two very nearby frequency levels that is used. A receiver knows the pseudo-random sequence (for each satellite), and slides its copies relative to the received signal until the match gets perfect (c.f. again Figure 3.6.1).

The correlation function $f(x)$ of two functions, $g(x)$ and $h(x)$, is defined as,

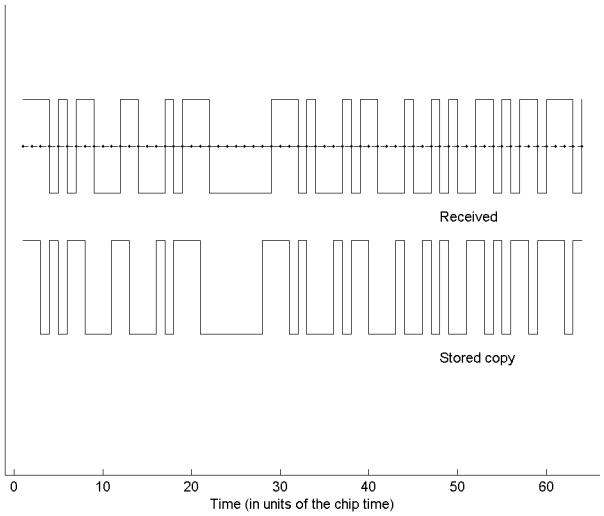


FIGURE 3.6.1. Pseudo-random (PR) received code compared with a copy stored in the receiver (in this picture shifted one chip time).

$$f(x) = \int g(s)h(s+x)ds,$$

i.e. it measures the 'similarity' between two functions as they are shifted aver each other. Figure 3.6.2 shows the correlation function of the two sequences of Figure 3.6.1 (where summation replaces the integral), displayed as a function of the sideways shift in a case of a periodic PR sequence of length 128.

Whenever the shift is a multiple of 128, we get here a very sharp spike of perfectly triangular shape with a base width of 2 chip times. Even a small misalignment in time—a fraction of the chip time—will notably bring down the correlation. A timing error of about 1/20th of a chip time would lead to a distance error of about 15 m; typical of the hardware capabilities of present low cost GPS receivers.

Of course, with 24 satellites in orbit, each receiver has to test the received signal against 24 copies.

The whole PR sequence repeats about each 1 ms. Given the speed of light, this corresponds to about 300 km. All distances to satellites become undetermined with respect to multiples of this distance. The easiest way to avoid this ambiguity

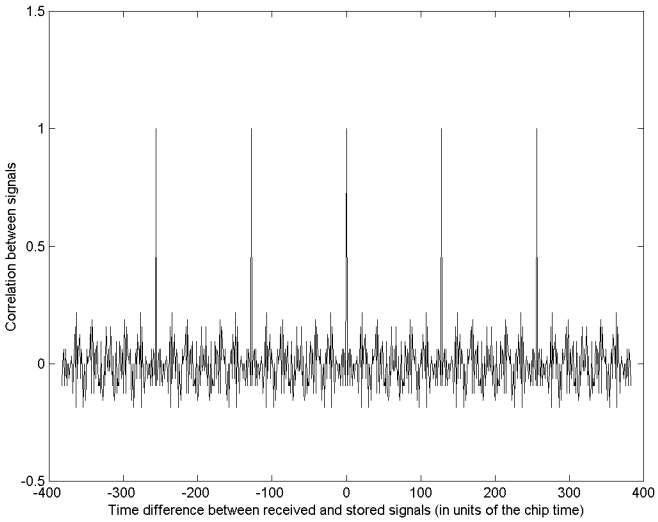


FIGURE 3.6.2. Correlation between a periodic PR code of length 128 and a translated copy of itself. The sharp spikes appear here every 128 chip times.

is to require that the receiver has to be initialized—the user has first to enter an approximate position that is accurate to better than 150 km, thus removing the arbitrariness. There are (at least) two other ways to resolve this problem:

i.: Guessing wrong by 300 km (when receiving signals from the minimal number of satellites required to get a position) is very likely to produce a position hundreds of kilometers under ground or above ground—easily rejected for all receivers not used in space crafts, and

ii.: When receiving signals from more satellites, a wrong guess will quite certainly produce a contradiction—no single point will satisfy all the distance requirements.

This second observation allows us to state the problem of finding our position as one of locating a point so that the distance to every observed satellite is equal to an unknown integer plus a known (measured) fractional part of this 300 km distance. Determining these integers would be an example of *integer programming*—the task of

finding best approximations to a problem for which several unknowns are constrained to integer values only.

> Integer programming is actually of great importance for GPS, but in a slightly different way. To get the highest possible accuracy, we apply the idea not to the 300 km PR sequence repeat distance, but to the approximately 0.2 m wavelength of the 1745.42 MHz carrier wave. Trying to lock on to the phase angle of the carrier oscillations, we get distances to within an unknown multiple of 0.2 m. With a good positional guess, say from differentially corrected GPS, we may have only about 50—100 multiples to be concerned with. Finding a position that gives a correct carrier phase for all available satellites can pinpoint just which carrier multiple we are locked onto for each of the satellites in view, i.e. an error better than 0.2 m. Finally, being locked onto exactly the right integer multiple of the carrier oscillation, the phase angle can be reconciled to maybe one part in 200. The accuracy is now down to about 1 mm—not bad considering that the radio signals are of quite narrow bandwidth and how fast these satellites fly far out in space! For more discussion on this issue of locking onto an individual carrier wave and its phase angle—and on integer programming in the context of GPS—see Strang and Borre [**?**].

CHAPTER 4

Radar Scattering from Aircraft

4.1. Introduction.

Chapter to be written. Another suggestion: A chapter on Climate Modeling?

CHAPTER 5

FREAK OCEAN WAVES

5.1. Introduction.

Large waves on the oceans usually appear as the result of storms, and they tend to arrive in lengthy wave trains. However, numerous marine disasters are also caused by more sporadically appearing isolated giant waves. For example Bascom (1980) tells about ships being lost and of survivors describing the cause as a huge solitary wave. He also recounts that both the liners Queen Mary and Queen Elizabeth were fortunate to survive dramatic freak wave encounters in the North Atlantic (at one point causing the former to roll to within two degrees of its point of no-return while carrying 15,000 US troops to the UK in World War II).

Although freak waves can arise in all oceans, one particular stretch of one of the major shipping routes is particularly prone to these. This is within the Agulhas Current off the SE coast of South Africa, approximately between Durnford Point (shortly north of Durban) and Port Elizabeth—see Figure 5.1.1. Table 1 lists some large ships which have been severely damaged by freak waves within this area during the 11 years 1981-1991 (not including minor ships such as fishing vessels etc.).

Incidents in the Agulhas Current before this time period include

- Passenger liner Waratah, lost in 1909 while carrying 211 crew and passengers by the end of the homeward leg of her maiden voyage (to Australia). No trace of the ship was ever recovered.
- Supertanker World Glory, sunk in 1968 after being broken in two by a single freak wave,
- Supertanker Neptune Saphire in 1973 lost its front 60 m bow section.

Figure 5.1.2 shows the cargo liner Bencruchan with its front quarter bent down after a freak wave incident off the SA coast (also in 1973). Figure 5.1.3 shows a giant wave breaking over the front deck of the super-tanker Esso Nederland in 1978 (with no

DATE		VESSEL	TYPE	DWT	DATE		VESSEL	TYPE	DWT
81	4	Energy Endurance	T	205,807	87	7	Goldstar	OBO	145,057
	8	Schelderin	T	230,679		10	Bocita	B	
	8	Rimula	OBO	227,412	88	11	Atlantic Emperor	T	292,641
82	4	Alva Sea	OBO		89	7	Arabian Sea	T	315,695
	7	Marofa	T	135,000		11	Paaficos	T	268,467
	7	Antonios	T	290,558	90	1	Rokko San	OBO	200,000
	7	Theodora	B	137,519		9	Dorado Star	T	
	7	Victoria	T	236,810		11	Samjohn Captain	B	65,051
	9	Torvanger	CH-T	17,057	91	4	Vasso	B	51,181
84	7	Merity	CH-T			5	Alborz	T	230,673
	11	Alva Sea	OBO	225,010		8	World Renown	T	262,267
85	2	Musashi	B			8	Mimosa	T	357,647
86	6	World Scholar	T	268,000		8	Novelty	T	233,399
	8	Formosa Fortune	OBO			8	Settebello	T	317,354
						9	Atlas Pride	T	248,602

TABLE 1. Major ships severely damaged in the Agulhas Current 1981-1991. Ship types: T - Tanker, OBO - Oil/Bulk ore, B - Bulk, CH-T Chemical tanker. Source: Pentow Marine Salvaging Co, S.A.

damage suffered to the vessel). Figures 5.1.4 and 5.1.5 show the damages suffered by the first and the last ships that were listed in Table 1.

Ships tend to travel in the direction of a current (to gain speed), and freak waves tend to move against a current, causing frontal damage to ships to be predominant. A freak wave is often preceded by a long sloping trough, in which the ship accelerates downward before being hit by the wave. The frontal damage can arise either when the bow gets buried into the freak wave, or when the high pressure is suddenly released as the ship emerges out of it. Many questions relating to freak waves remain open. For example, little is known about their typical sideways extent and the distance they can travel before dissipating or otherwise breaking up.

5.2. Mechanism for freak ocean waves.

Different explanations have been proposed for freak waves. For example, Dawson (1977) suggests they can be attributed quite directly to the bottom topography, an idea "found to be fallacious" according to Shillington and Schumann (1993). The

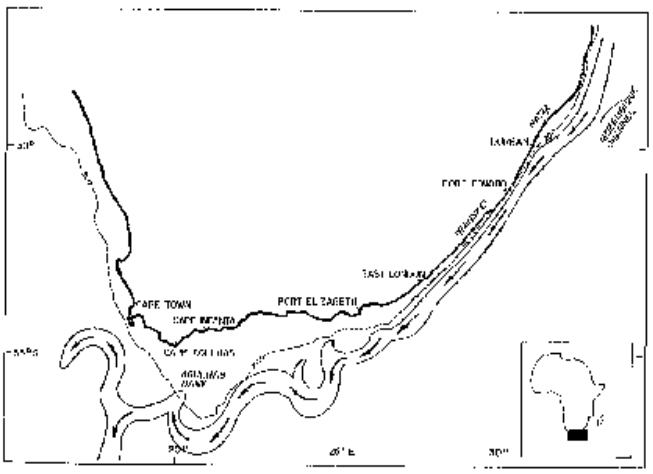


FIGURE 5.1.1. Schematic view of the southern part of Agulhas current.

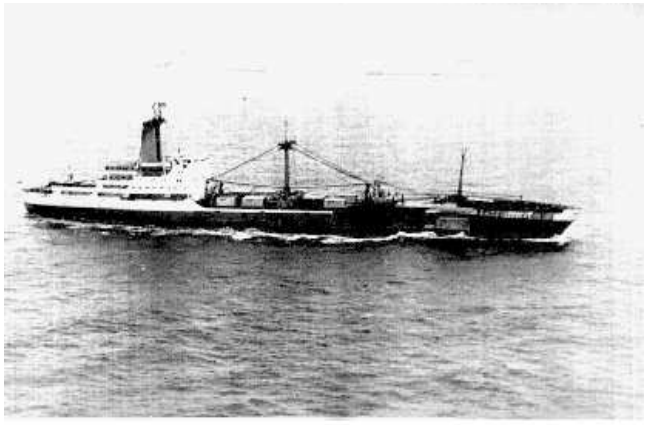


FIGURE 5.1.2. Cargo liner Benchruchan—with front quarter of the ship bent down after a freak wave incident.

most plausible mechanism is a focusing process, described by Crapper (1984) and further studied by in Gerber (1993), (1996).

Empirical statistics on ocean waves tell that, on the average, one wave in 23 is over twice the mean wave height, one in 1,175 over three times it, and one in 300,000 over four times the average height. Although rough weather appears to favor the emergence of freak waves—their energy has to come from somewhere—they do not seem to simply represent the top end of this distribution.

FIGURE 5.1.3. Large wave in the Agulhas current breaking over the front deck of the supertanker Esso Nederland.

FIGURE 5.1.4. Bow damage to Energy Endurance.

FIGURE 5.1.5. Bow damage to Atlas Pride.

Regarding the SE coast of South Africa (known as the "Wild Coast"), there are however several special circumstances that should be noted:

- One of the fastest and largest ocean currents in the world flows along the coast (cf. Figure 5.1.1). The Agulhas Current can reach around 2-3 m/s (4-6 knots) and transports about $70 \cdot 10^6$ m^3/second (250 km^3/hour) of water. It follows very closely the edge of the continental shelf, and it is fairly free of side-eddies while it makes a slow turn in the area which is most prone to the freak wave phenomenon.
- This current meets nearly head-on a steadily incoming near-monochromatic wave swell, constantly generated in the 'roaring forties', see Figure 5.2.1.

Waves can travel very long distances before loosing energy due to internal viscosity. Crapper (1984) notes that a wave with time period $T = 9$ s (average for ocean swell; wavelength $\lambda \approx 126$ m) looses only about 1/10th of its height in a distance corresponding to three times around the earth. For a short $T = 2$ s - wave ($\lambda \approx 6.2$ m), the distance is much smaller - about 62 km.

- Freak waves have been found to occur particularly frequent during weather situations such as the one shown in Figure 5.2.2. This supports the notion of the incoming swell as a major factor.

Waves meeting a current not only get their wave length reduced (coming closer to breaking) - their directions also change. This will turn out to be the key effect in modeling the freak wave phenomenon.

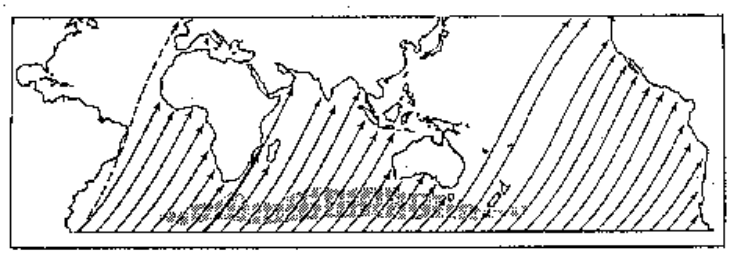


FIGURE 5.2.1. Typical swell paths. In the shaded area - part of the 'roaring forties' - average wave heights exceed 4.5 m. (Data from Chelton, Hussey and Parke, 1981).

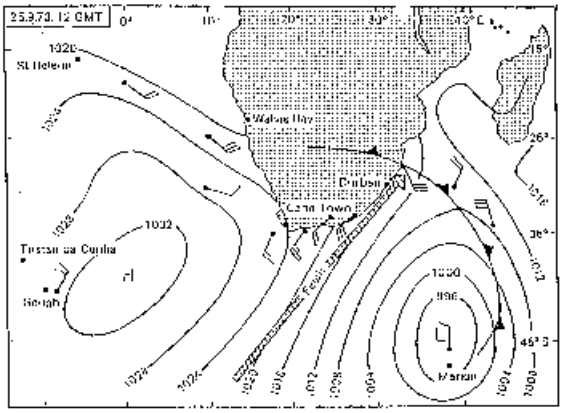


FIGURE 5.2.2. Weather situation most likely to generate freak waves (long 'fetch' strengthening the incoming swell; illustration from Mallory, 1974).

5.3. Derivation of the governing equations.

We start by assuming that the incoming wave amplitudes are low enough that their propagation can be approximated as linear - different waves will then mainly superpose on each other. This simplification will quite certainly not hold for 30 meter high giant waves near breaking, but we are here primarily interested in the mechanism that can lead to such waves rather than in their dynamics once they have built up to giant size. Let the wave (for now in 1-D) have the form

$$\eta(x,t) = a\ \cos(k\ x - \omega t) = f\mathrm{Re}\ a\ e^{i\ (k\ x - \omega\ t)} \tag{5.1}$$

For simplicity in writing, we will follow the convention of omitting "Re" in most of what follows, i.e. we will be working with

$$\eta(x,t) = a\ e^{i\ (k\ x - \omega\ t)} \tag{5.2}$$

For deep water, we note in Section 8.2 the dispersion relation

$$\omega = \sqrt{gk}. \tag{5.3}$$

Here, g is the acceleration of gravity, approximately 9.8 m/s^2. It follows from (5.2) that the wave length λ satisfies

$$\lambda = \frac{2\pi}{k} \tag{5.4}$$

and also that the phase speed $c_p = \sqrt{\frac{g}{k}} = \sqrt{\frac{g\lambda}{2\pi}}$ and the time period $T = \frac{2\pi}{\omega}$.

A typical wave swell might have $T \approx 10$ s, and hence $\lambda \approx 160$ m (swells of up to $\lambda \approx 800$ m have been recorded). The highest possible steady waves can be shown to have $h/\lambda \approx 0.142$ (with h being the vertical distance between crest and trough). For $\lambda = 160$ m, the maximal steady height would thus be about 23 meters. That is about 5 times the height that is typical in the ocean south of South Africa. Waves that are so much lower than the steady ones of maximal height behave reasonably linearly, and (5.2) is a good enough wave model for studying the onset of focusing.

A usual way to derive governing equations is to look at the mechanics of a phenomenon in detail—see what happens on local time and space scales of Δt and Δx and, in the limit of these getting smaller, obtain some governing ODEs or PDEs (as we did in the case of a string under tension in Section 8.3). The approach we used in that section for obtaining Maxwell's equations was somewhat different in that we started out with some laws in terms of integrals along arbitrary contours, and we then applied techniques of calculus to modify these into volume integrals, again giving governing PDEs as the volumes were decreased to zero. For the present problem, a third approach is needed. The first of the approaches above is clearly impossible; there is far too much data and complexity involved in modeling ocean dynamics all the way down to its individual waves. This would require discretization into trillions of mesh points, and totally astronomical computer resources (for both operation count and memory).

The totally different approach needed here is described in detail in Section 8.7. The key observation is that the phase of the wave field is sufficiently well-behaved to allow one to identify local frequency ω and wave number $\mathbf{k}$. In addition one notes that the frequency and wave numbers are related through a dispersion relation,

$$\omega = W(\mathbf{x}, \mathbf{k}, t).$$

Using these two ideas it is rather straightforward to derive a Hamiltonian system for the characteristic curves for $\mathbf{x}$ and $\mathbf{k}$

$$\begin{aligned} \frac{dx_j}{dt} &= \frac{\partial W}{\partial k_j}, \\ \frac{dk_j}{dt} &= -\frac{\partial W}{\partial x_j}, \end{aligned} \tag{5.5}$$

where $j = 1, \ldots, n$ and n is the spatial dimension of the problem. Thus the main task therefore has to be to identify the dispersion relation. Very surprisingly, this general observation, together with Hamilton's equations contain all the physics that is needed for modeling wave focusing in the presence of arbitrary currents.

5.3.1. The Hamilton equations for a 2-D wave field. In Section 8.4, it was shown that the dispersion relation for deep water waves in the absence of any current, is given by

$$\omega = \sqrt{g\,|\mathbf{k}|}.$$

Since it does not depend on $\mathbf{x}$, Hamilton's equations show that the characteristic curves are straight lines which does not allow any interesting phenomena. The situation changes significantly if one introduces a current moving with velocity $\mathbf{U}(\mathbf{x})$. The time frequency, as seen by a stationary observer, would include a Doppler correction, and the dispersion relation becomes,

$$\omega = W(\mathbf{x}, \mathbf{k}) = \sqrt{g\,|\mathbf{k}|} + \mathbf{k} \cdot \mathbf{U}.$$

According to Hamilton's equations, the quantity remain constant along the characteristic curves. will hold at all spatial locations.

In 2-D, $\mathbf{x} = [x_1, x_2]$ and $\mathbf{k} = [k_1, k_2]$ and the plane wave

$$\psi(\mathbf{x}, t) = e^{i(\mathbf{k}\cdot\mathbf{x} - \omega t)}$$

travels in the direction of the $\mathbf{k}$-vector. The dispersion relation takes the form

$$\omega = W(x_1, x_2, k_1, k_2) = \sqrt{g}\,(k_1^2 + k_2^2)^{1/4} + k_1\,U_1(x_1, x_2) + k_2 U_2(x_1, x_2) \tag{5.6}$$

Hamilton's equations (5.5) are now explicitly written

$$\left\{\begin{array}{rcl} \frac{dx_1}{dt} &=& \frac{\partial W}{\partial k_1} \\ \frac{dx_2}{dt} &=& \frac{\partial W}{\partial k_2} \\ \frac{dk_1}{dt} &=& -\frac{\partial H}{\partial x_1} \\ \frac{dk_2}{dt} &=& -\frac{\partial W}{\partial x_2} \end{array}\right. \tag{5.7}$$

will hold. Next step is to use (5.6) to calculate the partial derivatives that appear in the RHSs of (5.7). Straightforward algebra gives

$$\left\{\begin{array}{ll} \frac{\partial W}{\partial k_1} & = \alpha\ k_1 + U_1 \\ \frac{\partial W}{\partial k_2} & = \alpha\ k_2 + U_2 \\ \frac{\partial W}{\partial x_1} & = k_1 \frac{\partial U_1}{\partial x_1} + k_2 \frac{\partial U_2}{\partial x_1} \\ \frac{\partial W}{\partial x_2} & = k_1 \frac{\partial U_1}{\partial x_2} + k_2 \frac{\partial U_2}{\partial x_2} \end{array}\right.$$

where $\alpha = \frac{\sqrt{g}}{2|\underline{k}|^{3/2}}$. Therefore, the coupled system of ODEs which needs to be solved in order to find $\mathbf{k}(\mathbf{x})$—describing the wave field everywhere—becomes

$$\left\{\begin{array}{lcll} & \text{With no} & & \text{Terms due} \\ & \text{current} & & \text{to current} \\ \frac{dx_1}{dt} & = \alpha k_1 & + & U_1 \\ \frac{dx_2}{dt} & = \alpha k_2 & + & U_2 \\ \frac{dk_1}{dt} & = & - & \left(k_1 \frac{\partial U_1}{\partial x_1} + k_2 \frac{\partial U_2}{\partial x_1}\right) \\ \frac{dk_2}{dt} & = & - & \left(k_1 \frac{\partial U_1}{\partial x_2} + k_2 \frac{\partial U_2}{\partial x_2}\right) \end{array}\right. . \tag{5.8}$$

What this tells is that, if the current $\mathbf{U}(\mathbf{x})$ is given, and we at some location $\mathbf{x} = [x_1, x_2]$ specify an initial wave (by giving $\mathbf{k} = [k_1, k_2]$ there), solving the four coupled ODEs (5.8) forward in the parameter t will trace out a path in the (x_1, x_2)-plane along which we at the same time obtain k_1 and k_2. Starting from many places along a (spatial) domain boundary, we can trace out a dense set of paths in the $\mathbf{x}$-plane—hence obtain k_1 and k_2 throughout a domain. Each such path represents how a ray progresses. Rays are a natural concept in case of light, but for water a more abstract concept; a path that is orthogonal to the wave fronts. Wave energy can be shown to follow these rays—if rays cross each other, wave energy has focused. Solving this coupled system of ODEs will therefore produce *paths* in the $\mathbf{x}$-plane along which

energy travels. The independent variable t is for the moment just a path parameter, although it actually turns out to correspond to physical time describing the speed by which energy propagates along the paths.

Note:

- Well hidden in the interpretation of the rays are a few physical assumptions than have not yet been pointed out. A notable one is that the wavelength λ must be small in comparison with any objects or flow features (such as currents). This tends to be the case for light, making rays then a particularly clear concept. It is much less clear in the case of sound—such waves travels easily around everyday objects. Another key difference between the linear propagation of light and the non-linear case of water waves is that the former can cross each other without any interference. Large water waves coming together are likely to break, and therefore change their character. The ray model is mainly useful to highlight areas with potentially high wave energies, but it can not be expected to give any details after waves have passed through focusing areas.

5.4. Test problem - Circular current.

Figure 5.4.1 shows a test problem for ray tracing that has been used by some investigators (Gerber, 1993, White and Fornberg, 1997)—a circular current $U(x, y)$ with inner radius 40 km and outer radius 160 km.

Between these boundaries, the velocity profile (a function of radius r only) is parabolic, with a maximal velocity of 2 m/s.

We now denote the spatial directions x and y (rather than x_1 and x_2), and the velocity components in these directions $u(x, y)$ and $v(x, y)$ respectively. The easiest way to specify a current field $u(x, y)$ and $v(x, y)$, which needs to satisfy the incompressibility requirement

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \tag{5.1}$$

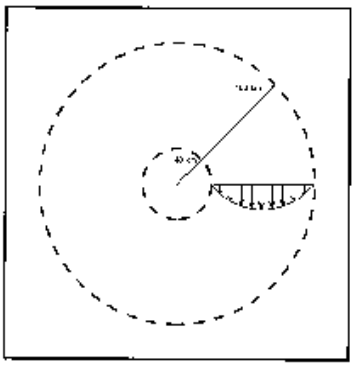


FIGURE 5.4.1. Test example of circular current. Parabolic velocity profile with maximum velocity of 2 m/s.

may be to note that (5.1) is equivalent to the existence of a stream function $\psi(x, y)$ such that

$$\left\{ \begin{array}{ll} u & = \psi_y \\ v & = -\psi_x \end{array} \right. . \tag{5.2}$$

Given any (reasonably smooth) function $\psi(x, y)$, the velocities $u(x, y)$ and $v(x, y)$ it produces through (5.2) will automatically satisfy (5.1). Figure 5.4.2 shows the $\psi(x, y)$ function (undetermined with respect to an additive constant) corresponding to a circular current.

Looking along the x-axis, we have

$$v(x,0) = \left\{ \begin{array}{ll} 0 & 0 < x < 40e3 \\ \frac{(x-40e3)(x-160e3)}{1.8e9} & 40e3 < x < 160e3 \\ 0 & x > 160e3 \end{array} \right. .$$

Hence, from (5.2) and noting that $\psi(x, y)$ becomes a function of r only ($r = \sqrt{x^2 + y^2}$), we obtain

$$\psi(r) = \left\{ \begin{array}{ll} -\frac{1.76e6}{27} & 0 < r < 40e3 \\ -\frac{32r}{9} + \frac{r^2}{1.8e4} - \frac{r^3}{5.4e9} & 40e3 < r < 160e3 \\ 0 & r > 160e3 \end{array} \right. .$$

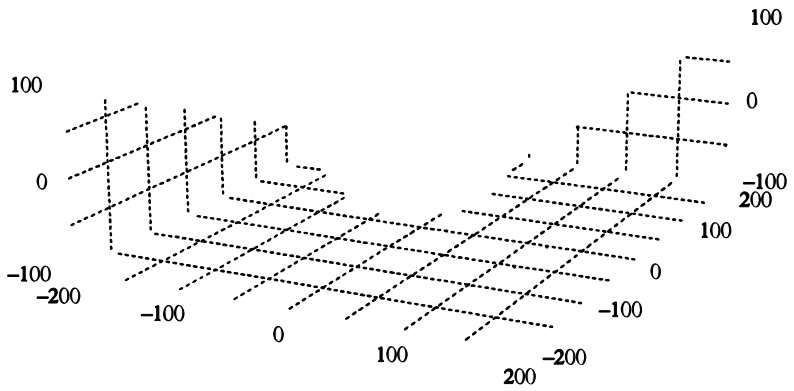


FIGURE 5.4.2. Stream function $\psi(x, y)$ for the annular current (with horizontal scale in kilometers, vertical in $1000m^2s^{-1}$

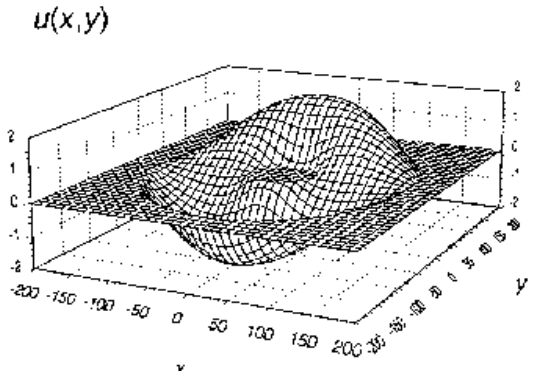


FIGURE 5.4.3. Velocity component in x-direction of circular current.

Figure 5.4.3 shows the resulting $u-$field (as to be expected with its largest value at (0,100) (when measured in km), most negative at (0,-100), and zero along $y = 0$).

In the wave ray field shown in Figure 5.4.4, each wave has a time period $T = 10$ s, i.e. according to (5.3) $\omega = 5/\pi$.

From (5.3) follows then $|\underline{k}| = 25/(9.8 \cdot \pi^2)$. At equi-spaced positions $\underline{x}$ along the left and bottom edges of the domain, we start all the rays with $\underline{k} = (1, 1) \cdot |\underline{k}| / \sqrt{2}$ (a vector of the length corresponding to $T = 10$ s, and pointing towards NE). The computation of the paths involves the following steps:

- Generate the $u-$ and $v-$fields for the circular current, and

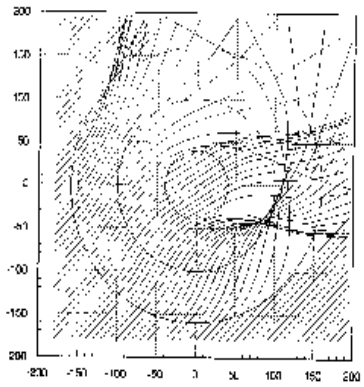


FIGURE 5.4.4. Ray pattern for circular current in case of incoming monochromatic waves from SW with a time period $T = 10$ s (corresponding to a wavelength $\lambda \approx 160$ m.

- For each ray, solve the four coupled ODEs (5.8) for as long time as is needed until the ray exits the domain.

In our code a second order, 2-stage Runge-Kutta method is used as the ODE solver. Many other choices would have worked equally well (cf. the discussion of ODE solvers in Chapter 16). The other main numerical issues are interpolation and finite difference approximations. At every (discrete) $(x, y)-$position that we reach along a ray path, the values for $u,\ v,\ u_x,\ u_y, v_x,\ v_y$ must all be obtained by interpolation from adjacent $(x, y)-$grid positions.

A ray path picture like Figure 5.4.4 is very effective in illustrating focus spots of wave energy. In other cases, it might be more useful to display the wave energy (roughly corresponding to the ray density) as a function of spatial position as in Figure 5.4.5 - generated from the rays in Figure 5.4.4.

5.5. Atlas Pride incident revisited.

To apply the numerical ray tracing technique to the region SW of South Africa, we need data on the current and the incoming wave swell. The current can be seen quite well in infrared images from weather satellites. Figure 5.5.1 is a black-and-white reproduction of a false-color high resolution original which shows warmer water in brighter colors. Small eddies can readily be identified, and their motions tracked on

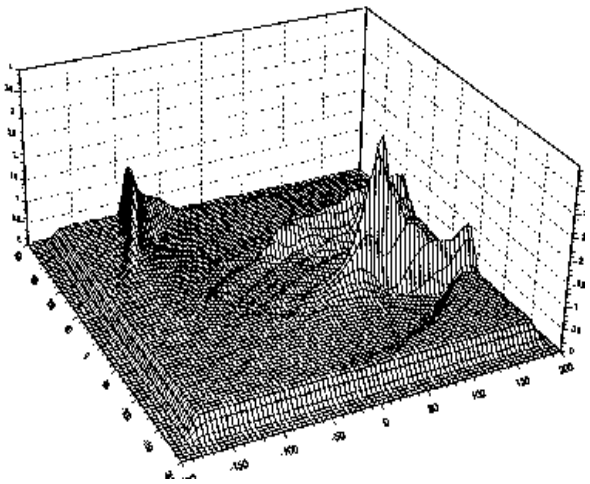


FIGURE 5.4.5. Ray pattern translated into a surface display of ray density - an approximate measure of wave energy.

FIGURE 5.5.1. Infra-red weather satellite picture of southernmost part of the Agulhas current, as it breaks up into eddies just south of the main danger zone. (The sloping line drawn in the rectangular box shows the path of space shuttle Challenger during an unsuccessful attempt to record freak wave patterns from space by radar).

a sufficiently frequent basis for providing all the needed current velocity data. Since the incoming wave swell can be approximated well by a monochromatic wave train, wave number (a 2-vector) and amplitude (a scalar number) suffice as input wave data. Buoys in the ocean record this.

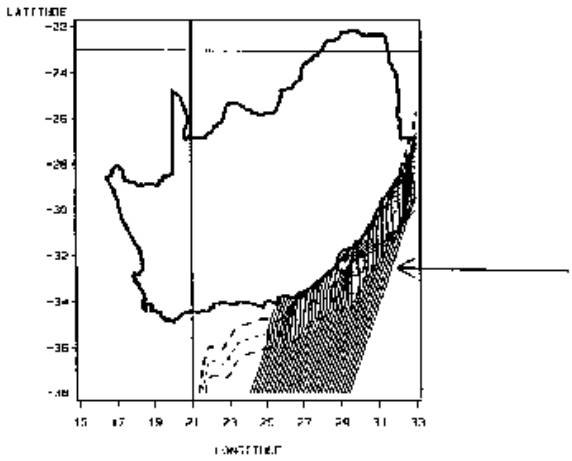


FIGURE 5.5.2. Calculation of wave (ray) paths based on data for current and incoming swell on 4/7/91. A primary focus is clearly visible near the inner edge of the current, which further north acts like a wave guide.

Figure 5.5.2 shows the result of a ray tracing based on data for 4/7/91. A very strong focus can be seen shortly south of Durban. North of that, we can see how waves have got trapped within the current, acting like a wave guide. The ray picture changes on a time scale of several hours, and is significantly different on the date of the Atlas Pride accident later that year (9/8/91), cf. Figure 5.5.3 (showing a larger scale detail of the area south of Port Elizabeth). The small arrow points at the position of Atlas Pride at the time of the accident, again seen in different types of displays in Figures 5.5.4 and 5.5.5 (both generated from this same simulation).

The calculation highlights as a danger area maybe 1/1000th of the total area off the SE coastline. In a second case (the Alborz, 1991), the agreement between computed danger area and site of actual incident proved equally good. The probability for a 2-out-of-2 score having occurred purely by chance is small.

5.6. Creation of freak waves in an energy-rich ocean state.

The focusing mechanism, described in the previous sections, show how danger areas can arise and their position be computed. Since these areas are considerably smaller in extent than the width of the current, shipping (if receiving proper forecasts)

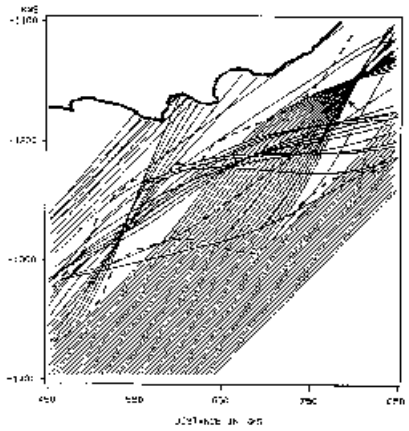


FIGURE 5.5.3. Ray picture for the Atlas Pride accident. Small arrow marks the position of the ship.

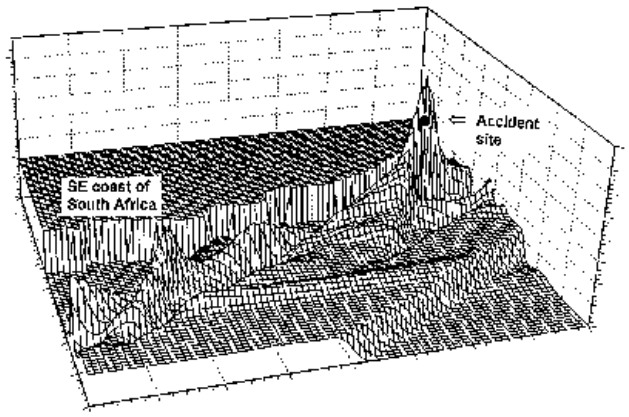


FIGURE 5.5.4. Ray intensity represented as surface elevations.

can still safely utilize the 4-5 knots extra speed it offers on travels along its path. However, an energy rich sea state does not imply that freak waves need to arise. Even under such circumstances, freak waves is a relatively rare phenomenon. In particular, our analysis has so far given no indications about issues like

- Onset mechanism for freak wave from an energy-rich sea state,
- General characteristics of freak waves. These are often, by eye witnesses, described as a slowly sloping long trough followed by one or two giant waves,

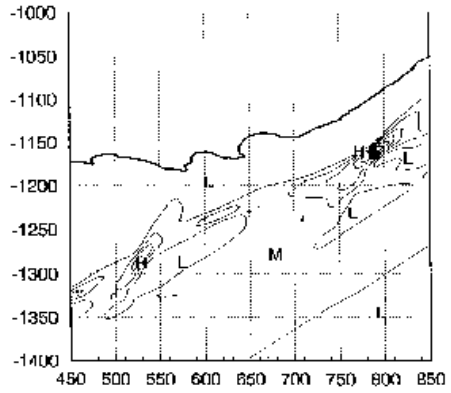


FIGURE 5.5.5. Contour plot of the ray intensity.

appearing as near-vertical walls of water, as schematically indicated in Figure 5.6.1)

- Sideways extent of freak waves,
- Duration in time and distance traveled by freak waves before disintegrating back to regular waves.

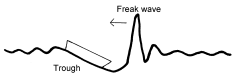


FIGURE 5.6.1. Schematic illustration of a freak wave preceeded by a trough, moving against a current.

The onset process is clearly a highly nonlinear process. A recent numerical study by V.E. Zakharov et.al. (2002) may provide some insights in this. Their numerical calculation is in 1-D (meaning one horizontal space dimension together with a vertical one plus time). This work incorporates a number of analytical and numerical techniques (conformal mapping, Hamiltonian equations, FFT treatment of a Hilbert transform, etc.) and produces time simulations of highly nonlinear wave evolution.

While the 'classical' Benjamin-Feir instability demonstrates how a uniform 1-D wave train on deep water will develop instabilities over time, it deals only with relatively weak nonlinearities. Over longer periods in time, it models recurrencies of regular and less regular states - but it does not address what happens in case of

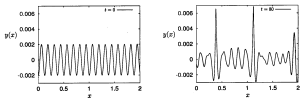


FIGURE 5.6.2. Emergence of freak waves in a 1-D simulation, starting from a uniform wave train.

very high waves. So far, only numerical computations can give insights in such flow regimes (since experimentally realizing a true 1-D wave appears to be exceedingly difficult). Figure 5.6.2 (from [..]) shows how the fully nonlinear mechanisms in large waves indeed can produce a dramatic energy concentration into single waves. Regarding freak waves in the ocean, one possibility is that the process illustrated in Figure 5.6.2 'kicks in' only when the original wave state happens to be locally be very nearly one-dimensional. If so, this could explain why energy-rich sea states (as we computed by the ray tracing method) are necessary but not at all sufficient for freak waves to be generated.

CHAPTER 6

PATIENT POSITIONING

6.1. Introduction.

The human stereo vision system consists of two basic components—the retinas (hardware) that record the images, and the brain (hardware) that process the images and do the three dimensional reconstruction. Studies (cite Stephen Pinker) indicate that we are not born with this ability, but that it is acquired shortly after birth—the software requires a training process. It is this 3D ability that allows us to become so proficient in different ball games, to judge the distance to the threat, to accurately navigate at high speed, and so on. There is little mystery behind the reconstruction process, the brain has to identify corresponding points on the two retinal images, and from their offset estimates the depth. The reader will be familiar with the following experiment: Hold you finger in front of you, alternatively close first the one eye and then the other and observe how the position of your finger changes with respect to the background. Now move your finger closer to your eyes and note how its positions against the background change from the previous experiment. It is this offset between the relative positions that allows the brain to estimate depth. You may also want to close one eye and look around. Do you note how different the world looks.

Incidentally, if you ever find yourself in a situation where you see double (such as after partying to much), it is an indication that your brain is malfunctioning—the two images from your retinas are no longer combined into a single 3D view.

This 3D stereo reconstruction is so useful that it is no wonder that we want to do the same on a computer. Again the challenge can be described quite easily, given two images of an object recorded at different angles, find the offset of the features and use that to do a 3D reconstruction of the features. In case it is not clear, by 'feature' we simply mean a point that is present in the image. If it is identifiable on both images, it is called a corresponding feature. It is the 3D coordinates of these

features that we are after. The 3D coordinates of the corresponding features from part of the structure we want to reconstruct, the more features the more accurate the reconstruction.

As pointed out above, 3D reconstruction from stereo images (there is a number of other ways of estimating 3D structures including X-ray tomography discussed in Chapter 1), find various applications. In this chapter we want to discuss a somewhat unusual, but important application of stereo reconstruction, namely its use to position patients undergoing proton therapy. In the next section we give some background information of the problem before we proceed to discuss the solution. Apart from the mainstream mathematics such as the Singular Value Decomposition discussed in detail in Section 11.5, stereo reconstruction relies heavily on an unfortunately much neglected, but very elegant geometric framework called projective geometry. Although we are not in a position to treat the subject in any depth, we will give you a flavor of it. And we trust that you will be able to put it to good use, not only for stereo reconstruction.

6.2. Proton Therapy.

Proton therapy is fast becoming one of the most significant modes of treatment of malignant tumours. In order to understand why it is so successful we really only need to know that an atom consists of a nucleus made up of positively charged protons, and neutrons, and surrounded by a cloud of negatively charged electrons. Assume that we are somehow able to get hold of a free proton and have a mechanism to send it through living tissue. Since the proton is positively charged, it attracts the negatively charged electrons surrounding the atoms as it passes through the tissue. As it it attracts the electron it transfers some of its momentum to the electron, slowing it down. As it slows down, the interaction with the nearby electrons increases, slowing it down even more, increasing its pull on the electrons. At some point the pull becomes strong enough to knock the electron out of its orbit, a process known as ionization. The effect this has on living tissue is to mess up the DNA thereby destroying the living cells. This is exactly what we need to destroy cancerous cells, but it is bad for healthy tissue. Whatever we do, including proton therapy, healthy cells are going to be destroyed in the process. All cancer treatment today is based on a single

principle: Healthy tissue recovers faster than the malignant tumors. Therefore, if one administer the treatment sessions over a period of time, the tumors are completely destroyed while the healthy tissue has time to recover between sessions.

What makes proton therapy so special is that one can localize the ionization much better than in other treatment procedures, thereby confining the damage to the tumor with little damage to the nearby healthy tissue. The basic idea was already conceived by Robert R Wilson in 1946 and illustrated in Figure 6.2.1 Note how sharply the relative dosage increases with depth in the tissue as described above, an effect known as the Bragg peak. Also note how well the proton dosage is localized in comparison with other treatments. In sense it is too localized, one wants to destroy the whole tumor not only a small part of it. For that reason the Bragg peak is modulated. First the energy with which the protons inter the body is calculated (it depends on the distance the protons need to travel through the healthy tissue in order to reach the tumor) so that the Bragg peak occurs inside the tumor. The proton beam is then modulated by placing a spinning disc of variable thickness in front of the beam. Some of the protons pass through thicker regions of the disc slowing then down. If the variation in thickness is just right, the Bragg peaks of the protons with different energies cover the total depth of the tumor. The proton beam obviously also has to be shaped in the form of the profile of the tumor. This can be done by placing a thick lead disc with a cut-out in the shape of profile of the tumor in front of the beam.

Although already proposed by Wilson in 1946, the first treatment on a regular basis only started in 1990 at Loma Linda University Medical Center. One of the reasons for the delay was due to the inability of doing 3D imaging; it was just not possible to determine the exact position and extent of the tumor. This changed when CT scans, based on the X-ray tomography discussed in Chapter 1 became available. At the moment (2007) some 26 facilities are in operation world wide, with many more under construction.

Of course there is the matter of getting hold of protons with sufficient energy to penetrate to the tumor. Since protons are positively charged they can be accelerated in a cyclotron. Proton therapy therefore requires a cyclotron, not exactly standard hospital equipment, at least not yet. Modern facilities newly constructed are in

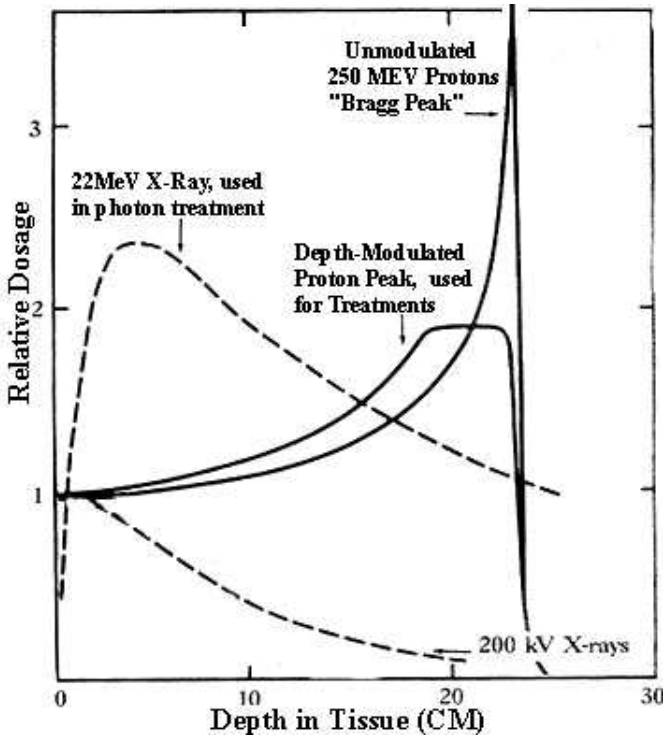


FIGURE 6.2.1. Bragg peak.

a position to design their cyclotrons specifically to treat malignant tumors. The designers however, face a formidable problem—the protons need to be accelerated to energies of the order of 250MeV. If this does not mean much to you, the electrons do about 5 million trips around the accelerator at about 600 000 000 rpm's. In the process the protons reach speeds about half that of light—they go fast. All our talk about the Bragg peak and precise localization of the ionization is in vain unless one is able to align the tumor exactly with the proton beam. Again the positive charge of the proton is useful, using giant magnets the protons can be deflected from their path. Since the protons move so fast these magnets are truly gigantic. A typical gantry system used to precisely align the proton beam to within a fraction of a millimeter is a 200 ton, three-story high construction. It will utilize several 2000kg magnets consuming 3000 ampere each. It is not energy efficient, the whole system may consume a million watts to deliver $\frac{1}{5}$ watt for $\frac{1}{3}$s to the tumor.

Although this is a solution that is common with modern treatment centers, there are a number of cyclotrons all over the world that are no longer powerful enough to do any useful physics that can be and are converted for medical purposes, including proton therapy. The proton beams from these cyclotrons are fixed and construction of a gantry system is in many instances prohibitively expensive. There has to be a cheaper solution.

The cases in point are the cyclotrons at iThemba laboratories near Cape Town in South Africa (see Figure 6.2.2) , and the original cyclotron (dating from very late 1940's) at the Joint Institute for Nuclear Research in Dubna, north of Moscow. Instead of aligning the proton beam, both these facilities position the patient, using completely different methods. The Dubna facility basically position the patient by taking a CT scan of the patient strapped to a maneuverable chair in treatment vault. The iThemba Lab solution employs stereo vision, described in the next section.

6.3. Patient Positioning.

The IThemba proton therapy procedure at the time of writing only treats tumors in the brain. Since the patient is manipulated into position internal organs shift and it is hard to determine the precise 3D location of the tumor with respect to the proton beam. The brain encased in the skull van be treated as a rigid object with no or minimal shift of the internal structures. This is can be exploited by first determining the position of the tumor in relation to external markers. For this purpose a tight-fitting mask is constructed for each patient. On the mask is a number of reflective markers as shown in Figure 6.3.1

This mask is then fitted on the patient and a CT scan is taken of the head with the mask in position. The CT scan shows the position of the tumor in relation to the CT scanner coordinate system. At the same time the 3D coordinates of the markers in the CT coordinate system are also determined using a stereo camera setup as shown in Figure6.3.2 (a). Figure 6.3.2(b) is an image of the CT scanner with cameras in position. Don't get too concerned at this stage, the rest of the chapter explains how to calculate the 3D coordinates using stereo vision.

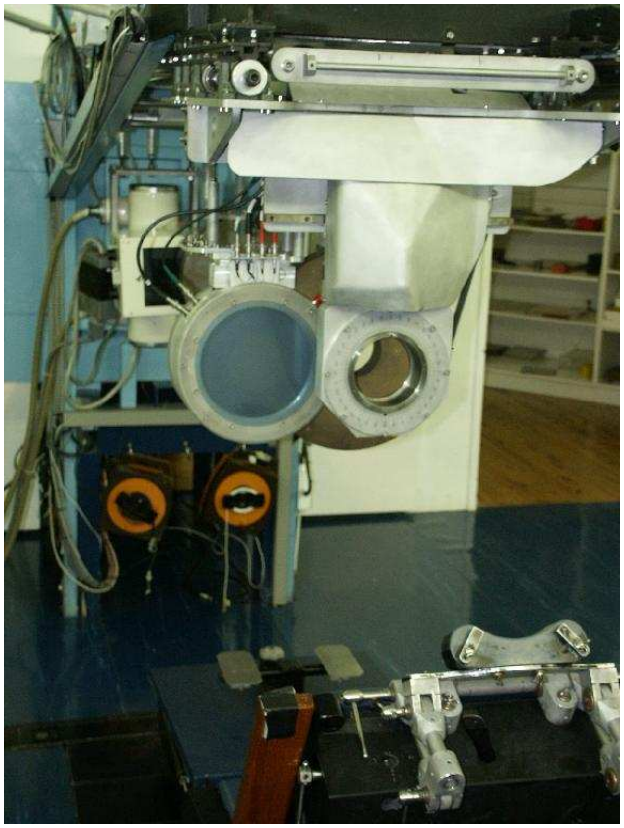
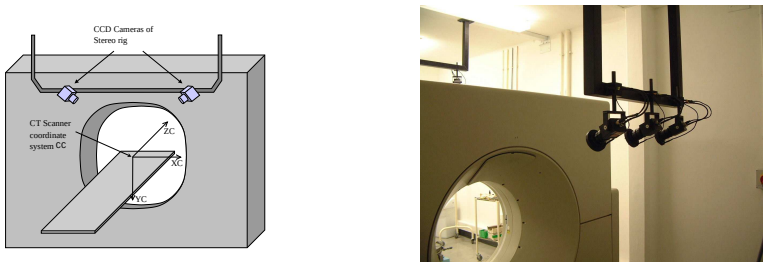


FIGURE 6.2.2. The exit of the proton beam from the cyclotron at IThemba Labs.

With the position of the tumor known relative to the markers, the patient is then immobilized on a mechanical chair, or a robot arm. Figure 6.3.3 shows the mechanical chair used before it was replaced with a mechanical robot arm.

The treatment vault is equipped with a number of cameras (9 at the time of writing). The reflective markers are automatically detected by at least 3 cameras (otherwise the marker is rejected) and it 3D position calculated with respect to the world coordinate system of the proton beam. Thus the 3D position of the markers, hence the 3D location of the tumor is known with respect to the proton beam coordinate system. Since the desired position is known—as determined by the clinical professionals—the patient is moved into position by manipulating the chair or the

FIGURE 6.3.1. Patient mask with reflective markers.



(a) (b)

FIGURE 6.3.2. CT scanner. (a) Coordinate system. (b) Actual scanner with cameras in position.

robot arm. A final X-ray is then taken to confirm that the patient is in position before the proton beam is activated.

Just to complete the picture, the cameras are also used to monitor the motion of the patient as it is moved into position. It also monitors the patient during treatment. If any motion id detected the beam is immediately shut down.

From the discussion above it should be clear that the heart of the system consists of a multiple camera setup that is used to calculate the 3D position of the. In the next sections we explain how that is done for stereo camera pairs.

FIGURE 6.3.3. Mechanical chair.

6.4. Planar geometry and the 2D projective plane.

We usually represent a point in the plane by a pair of coordinates $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$, thus we often identify the plane with $\mathbb{R}^2$. In this lecture we introduce homogeneous coordinates as a representation of points in a plane. This unifies the concept of the intersection of two lines (even parallel lines will be shown to intersect in a well-defined point), and leads us straight into the projective plane $\mathbb{P}^2$, the computation of intersection points of two lines and other useful concepts.

6.4.1. Homogeneous representation of lines. A line in the plane is represented by an equation of the form

$$ax + by + c = 0, \tag{6.1}$$

with different choices of a, b and c yielding different lines. The lines

$$2x + y - 1 = 0, \tag{6.2}$$

with $a = 2$, $b = 1$ and $c = -1$, and

$$x - 1 = 0,$$

with $a = 1$, $b = 0$ and $c = -1$, are illustrated in Figure 6.4.1 below.

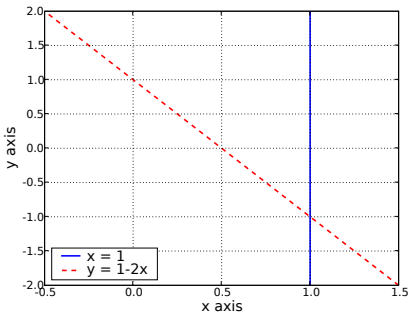


FIGURE 6.4.1. Two lines in the plane $\mathbb{R}^2$.

We now make an important observation: A line can either be specified by its coefficients $\mathbf{l} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$ or the set of points $\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$ satisfying(6.1) $ax + by + c = 0$. Note for instance that the lines Figure are uniquely determined by $\mathbf{l} = \begin{bmatrix} 2 \\ 1 \\ -1 \end{bmatrix}$ and $\mathbf{l}' = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$ respectively. But we now note an awkward asymmetry: it does not change the line if we multiply its coefficients with a non-zero constant, i.e. $k\mathbf{l} = \begin{bmatrix} ka \\ kb \\ kc \end{bmatrix}$, with $k \neq 0$ is exactly the same line as $\mathbf{l} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$, since x and y satisfy (6.1) if and only if they satisfy

$$kax + kby + kc = 0, \qquad k \neq 0.$$

For this reason we consider the vectors $\begin{bmatrix} a \\ b \\ c \end{bmatrix}$ and $k \begin{bmatrix} a \\ b \\ c \end{bmatrix}$ $(k \neq 0)$ to be equivalent. This however is not the case with the point representation of a line, $k\mathbf{x}$ is not the same point in the plane as $\mathbf{x}$. A complete symmetry is obtained if we note that the line (6.1) can be written as the inner product,

$$\begin{bmatrix} x & y & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = 0. \tag{6.3}$$

Here the point $\begin{bmatrix} x \\ y \end{bmatrix}$ in $\mathbb{R}^2$ is represented by a 3-vector by adding a third coordinate of 1. The important observation is that, for any $k \neq 0$, we have

$$\begin{bmatrix} kx & ky & k \end{bmatrix} \begin{bmatrix} a \\ b \\ c \end{bmatrix} = 0$$

if and only if (6.3) holds. It is therefore natural to consider the vector $\begin{bmatrix} kx \\ ky \\ k \end{bmatrix}$ $k \neq 0$ equivalent to $\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$.

An equivalence class of vectors under this equivalence relationship is known as a *homogeneous vector*, where any particular vector $\begin{bmatrix} a \\ b \\ c \end{bmatrix}$ is a representative of a whole class of equivalent vectors. The set of equivalence classes in $\mathbb{R}^3 \setminus \{\mathbf{0}\}$ (the vector space $\mathbb{R}^3$ with the zero vector $\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$ removed) forms the *projective space* $\mathbb{P}^2$.

6.4.2. Homogeneous representation of points. A point $\begin{bmatrix} x \\ y \end{bmatrix}$ lies on the line $\mathbf{l} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$ if and only if

$$ax + by + c = 0,$$

which can be written as the inner product (6.3) where the point $\begin{bmatrix} x \\ y \end{bmatrix} \in \mathbb{R}^2$ is represented by a 3-vector by adding a third coordinate of 1 and an arbitrary homogeneous vector $\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \in \mathbb{P}^2$ represents the point $\begin{bmatrix} x_1/x_3 \\ x_2/x_3 \end{bmatrix}$ in $\mathbb{R}^2$.

THEOREM 1. *The point*

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

lies on the line

$$\mathbf{l} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

if and only if

$$\mathbf{l}^T\mathbf{x} = 0. \tag{6.4}$$

In order to unique specify a point in $\mathbb{R}^2$, we need to specify two ratios (x_1/x_3 and x_1/x_3) in $\mathbb{P}^2$or two values (an x-coordinate and an y-coordinate) in $\mathbb{R}^2$. In a similar manner, a line is specified by two parameters and so has two degrees of freedom. For example, in an inhomogeneous representation of a line, we could uniquely specify a line by its gradient and y-intercept.

Homogeneous vector representations of lines and points allow us to express some previously clumsy concepts, for example the intersection of two lines, in an elegant way.

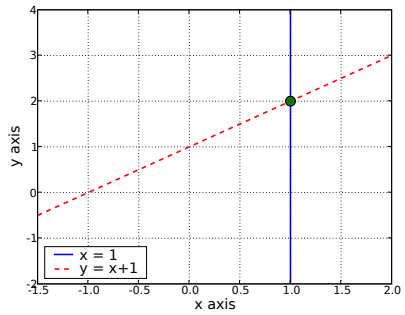


FIGURE 6.4.2. Two intersecting lines in the plane $\mathbb{R}^2$.

THEOREM 2. *The intersection of two lines* $\mathbf{l}$ *and* $\mathbf{l}'$ *is given by the point*

$$\mathbf{x} = \mathbf{l} \times \mathbf{l}'$$

where $\times$ *denotes the vector cross product.*

EXERCISE 3. Prove Theorem 2.

EXAMPLE 4. Determine the intersection point of the lines $x = 1$ and $y = x + 1$.

The line $x = 1$ is written as $-x + 0y + 1 = 0$, and thus has the homogeneous representation $\mathbf{l} = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}$, while the line $y = x + 1$ is written as$-x + y - 1$, and thus has the homogeneous representation $\mathbf{l}' = \begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix}$. Then, from Theorem 2, we calculate the intersection point as

$$\mathbf{x} = \mathbf{l} \times \mathbf{l}' = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ -1 & 0 & 1 \\ -1 & 1 & -1 \end{vmatrix} = \begin{bmatrix} -1 \\ -2 \\ -1 \end{bmatrix},$$

which is the homogeneous representation of the point $\begin{bmatrix} 1 \\ 2 \end{bmatrix}$, consistent with the intersection obtained in Figure 6.4.1}.

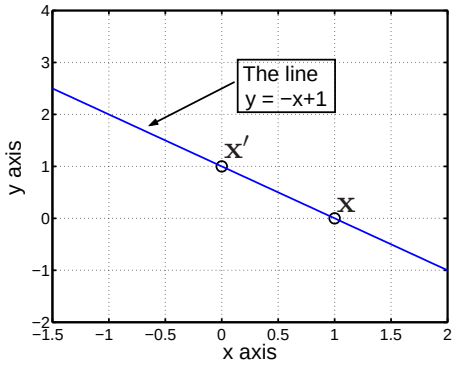


FIGURE 6.4.3. A line joining two points in the plane $\mathbb{R}^2$.

THEOREM 5. *The line* $\mathbf{l}$ *joining two points* $\mathbf{x}$ *and* $\mathbf{x}'$ *is given by the cross product*

$$\mathbf{l} = \mathbf{x} \times \mathbf{x}'.$$

EXERCISE 6. Prove Theorem 5.

EXAMPLE 7. Determine the line joining the points $\mathbf{x} = [0, 1]^T$ and $\mathbf{x}' = [1, 0]^T$.

The homogeneous representation of $\mathbf{x}$ and $\mathbf{x}'$ are $\begin{bmatrix} 0 & 1 & 1 \end{bmatrix}^T$ and $\begin{bmatrix} 1 & 0 & 1 \end{bmatrix}^T$ respectively. The homogeneous representation of the line joining the two points is therefore given by

$$\mathbf{l} = \mathbf{x} \times \mathbf{x}' = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ 0 & 1 & 1 \\ 1 & 0 & -1 \end{vmatrix} = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix},$$

or equivalently, $x_1 + x_2 - x_3 = 0$. The inhomogeneous representation is then $x + y - 1 = 0$, or equivalently, $y = -x + 1$, see Figure 6.4.3.

The homogeneous presentation of a line allows a particularly elegant expression for the intersection of parallel lines,

$$ax + by + c = 0 \quad \text{and} \quad ax + by + c' = 0.$$

These two lines are represented by the vectors $\mathbf{l} = \begin{bmatrix} a & b & c \end{bmatrix}^T$ and $\mathbf{l}' = \begin{bmatrix} a & b & c' \end{bmatrix}^T$ for which the first two coordinates are the same. Although these lines are parallel,

we can compute their intersection $\mathbf{x}$ using Theorem 2,

$$\mathbf{x} = \mathbf{l} \times \mathbf{l}' = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ a & b & c \\ a & b & c' \end{vmatrix} = \begin{bmatrix} bc' - bc \\ ac - ac' \\ 0 \end{bmatrix} = (c' - c) \begin{bmatrix} b \\ -a \\ 0 \end{bmatrix}.$$

Since we are working in homogeneous coordinates, we consider any nonzero multiple of a vector to be equivalent to the original vector, i.e. the point of intersection is

$$\mathbf{x} = \begin{bmatrix} b \\ -a \\ 0 \end{bmatrix}.$$

This is a perfectly well-defined point in the projective plane. If however, we try to find its Euclidean equivalent we get $\begin{bmatrix} \frac{b}{0} \\ -\frac{a}{0} \end{bmatrix}$. This is not defined, however, it does suggest that two parallel lines intersect at infinity. In addition, the coordinates $\begin{bmatrix} a & b \end{bmatrix}^T$ point in the direction of the parallel lines, i.e. in the direction of the point of intersection at infinity.

EXAMPLE 8. Consider the parallel lines $x = 1$ and $x = 2$ with homogeneous representations $\mathbf{l} = \begin{bmatrix} -1 & 0 & 1 \end{bmatrix}^T$ and $\mathbf{l}' = \begin{bmatrix} -1 & 0 & 2 \end{bmatrix}^T$. These two lines intersect at

$$\mathbf{x} = \mathbf{l} \times \mathbf{l}' = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{vmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix},$$

which is the point at infinity in the direction of the y-axis.

6.4.3. Ideal points and the line at infinity. The projective space $\mathbb{P}^2$ consists of all homogeneous vectors $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$, where $\mathbf{x}$ represents a finite point in $\mathbb{R}^2$ if $x_3 \neq 0$. If $x_3 = 0$ we say $\mathbf{x}$ is an *ideal point*. These points have no equivalent in Euclidean space, although it is useful to think of them as points at infinity.

Since the set of all ideal points consists of all points of the form $\begin{bmatrix} x_1 \\ x_2 \\ 0 \end{bmatrix}$ it follows that this set lies on the line $\mathbf{l}^T = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$, since

$$\begin{bmatrix} x_1 & x_2 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = 0.$$

We call this line the *line at infinity* and denote it by $\mathbf{l}_\infty = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$.

EXAMPLE 9. The line at $\mathbf{l}^T = \begin{bmatrix} a & b & c \end{bmatrix}$ intersects the line at infinity $\mathbf{l}_\infty^T = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix}$ at

$$\mathbf{x} = \mathbf{l} \times \mathbf{l}_\infty = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ a & b & c \\ 0 & 0 & 1 \end{vmatrix} = \begin{bmatrix} b \\ -a \\ 0 \end{bmatrix},$$

which is an ideal point. The line $\mathbf{l}'^T = \begin{bmatrix} a & b & c' \end{bmatrix}$, parallel to $\mathbf{l}$ intersects $\mathbf{l}_\infty$ at the same point.

In Euclidean coordinates, the vector $\begin{bmatrix} b \\ -a \end{bmatrix}$ is tangent to the line $ax+by+c=0$ or, equivalently, orthogonal to its normal, and so represents the line's direction. If we were to vary the line's direction, the ideal point $\begin{bmatrix} b \\ -a \\ 0 \end{bmatrix}$ would also vary over $\mathbf{I}_\infty$.

Thus the line at infinity can be thought of as a set of directions of lines in the plane.

6.4.4. Duality. The reader might have observed that there is a close connection between points and lines in $\mathbb{P}^2$—one either think of a line as a collection of points or in terms of its coefficients. In fact a much stronger statement is possible as formulated in the following duality theorem (stated without proof):

THEOREM 10. *For any theorem of 2-dimensional projective geometry there is a dual theorem, which may be derived by interchanging the roles of points and lines in the original theorem.*

This theorem ensures that for every statement involving points and lines, there is a dual statement where lines and points are interchanged.

6.4.5. A useful way to think of $\mathbb{P}^2$. The study of the geometry of $\mathbb{P}^2$ is known as projective geometry. A useful way of thinking of $\mathbb{P}^2$ is as a set of rays in $\mathbb{R}^3$, see Figure 6.4.4. The set of vectors $k \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$ (as k varies) forms a ray through the origin. Such a ray can be thought of representing a single point in $\mathbb{P}^2$.

The corresponding point in $\mathbb{R}^2$ may be obtained by intersecting a particular ray with the plane $x_3 = 1$, i.e., there is a one-to-one correspondence between $\mathbb{R}^2 \setminus \begin{bmatrix} 0 \\ 0 \end{bmatrix}$ and $\mathbb{P}^2$:

$$\text{if } \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \in \mathbb{R}^2 \Rightarrow \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix} \in \mathbb{P}^2$$

and

$$\text{if } \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \in \mathbb{P}^2 \Rightarrow \begin{bmatrix} \frac{x_1}{x_3} \\ \frac{x_2}{x_3} \end{bmatrix} \in \mathbb{R}^2.$$

6.5. Projective transformations.

6.5.1. Projectivities. The most useful transformation $h : \mathbb{P}^2 \longrightarrow \mathbb{P}^2$of the projective plane to itself is one that is invertible and maps straight lines to straight lines. More specifically, if the three points $\mathbf{u}$, $\mathbf{v}$ and $\mathbf{w}$ lie on a straight line then we require that $h(\mathbf{u})$, $h(\mathbf{v})$ and $h(\mathbf{w})$ also lie on a straight line. Such a transformation is known as a projective transformation or a projectivity, also sometimes referred as a collineation or homography.

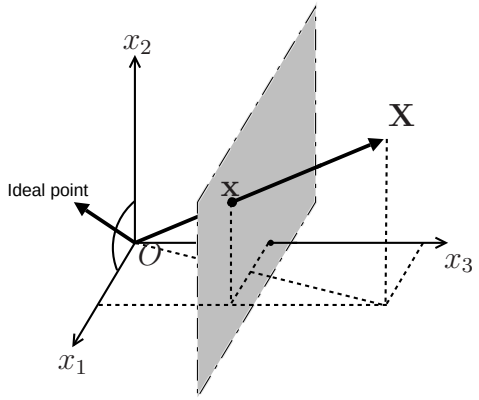


FIGURE 6.4.4. A model of the projective plane.

In the statement above we have emphasized that straight lines go to straight lines. Since we almost exclusively work with straight lines, we use the convention that when we say *lines* we mean *straight lines,* unless otherwise stated.

Fortunately there is a particularly simple representation of projective transformations, as the following theorem states:

THEOREM 11. *A mapping $h : \mathbb{P}^2 \longrightarrow \mathbb{P}^2$ is a projectivity if and only if there exists a non-singular, homogeneous 3×3 matrix H such that, for any point in $\mathbf{x} \in \mathbb{P}^2$ it is true that*

$$h(\mathbf{x}) = H\mathbf{x}.$$

EXERCISE 12. Show that if H is a 3×3, non-singular, homogeneous matrix, then the map $\mathbf{y} = H\mathbf{x}$ is a projectivity. The converse of the theorem is harder and can be accepted without proof.

This theorem enables to give an alternative definition of a projectivity:

DEFINITION 13. A projective transformation is a linear transformation $h : \mathbb{P}^2 \longrightarrow \mathbb{P}^2$ defined by a non-singular, homogeneous, 3×3 matrix

$$\begin{bmatrix} x_1' \\ x_2' \\ x_3' \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \tag{6.1}$$

or more briefly, $\mathbf{x}' = H\mathbf{x}$.

Note that, since the vectors $\mathbf{x}$ and $\mathbf{x}'$ are homogeneous, the projective transformations $\mathbf{x}' = kH\mathbf{x},\ k \neq 0$, and $\mathbf{x}' = H\mathbf{x}$, are equivalent. Thus our emphasis throughout that H is a *homogeneous* matrix—any non-zero multiple of H is equivalent to H.

There are 8 independent elements in the homogeneous matrix H, implying that there are 8 degrees of freedom in a projective transformation. This means that we we need 4 point correspondences to calculate H, as explained in the next section.

6.5.2. A Hierarchy of transformations. In this subsection, we consider special cases of the projective transformation, starting with the most restricted case, an isometry and ending with the most general projective transformation.

6.5.2.1. *Isometric transformations.* An *isometric* transformation (or isometry) of the plane $\mathbb{R}^2$ preserves Euclidean distance (*iso* = same, *metric* = measure), e.g. a rotation. The most general isometry is represented by the matrix equation

$$\begin{bmatrix} x_1' \\ x_2' \\ 1 \end{bmatrix} = \begin{bmatrix} \epsilon\cos\theta & -\sin\theta & t_x \\ \epsilon\sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}.$$

where $\epsilon = \pm 1$. If $\epsilon = 1$, then the isometry is *orientation-preserving* and is a Euclidean transformation (a composition of a rotation and a translation). If $\epsilon = -1$ the isometry reverses orientation, for example, a composition of a reflection and a translation. Here we focus on Euclidean transformations, as they are predominant in the applications.

We can write a Euclidean transformation in block form

$$\mathbf{x}' = H_E\mathbf{x} = \begin{bmatrix} R & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \mathbf{x},$$

where R is a 2×2 rotation matrix (an unitary matrix; a matrix satisfying $R^TR = RR^T = I$), $\mathbf{t}$ is a translation 2-vector and $\mathbf{0} = \begin{bmatrix} 0 & 0 \end{bmatrix}^T$.

Degrees of freedom. A Euclidean transformation has 3 degrees of freedom: 1 for rotation and 2 for translation. Thus 3 parameters must be specified to define the transformation. Accordingly, we can compute the transformation from 2 point correspondences.

Invariants

- length
- angles
- area
- parallel lines

Example. For the Euclidean transformation where the matrix R is a anticlockwise rotation through $\frac{\pi}{3}$ and a translation $\mathbf{t} = \begin{bmatrix} -\frac{1}{2} \\ 1 \end{bmatrix}$, the transformation matrix H_E is given by

$$H_E = \begin{bmatrix} \cos(\frac{\pi}{3}) & -\sin(\frac{\pi}{3}) & -\frac{1}{2} \\ \sin(\frac{\pi}{3}) & \cos(\frac{\pi}{3}) & 1 \\ 0 & 0 & 1 \end{bmatrix}.$$

Below is the PYTHON code used to apply this transformation to the unit square and unit circle. Note that we need to convert the xy-coordinates of both objects into homogeneous coordinates by adding a row of ones.

```
import pylab as P

import numpy as N



t = N.linspace(-N.pi,N.pi,200)

x = N.cos(t)

y = N.sin(t)

circle = N.vstack((x,y,N.ones(N.shape(x)))) # make homogeneous coordinates



P.plot(circle[0],circle[1],'b-')
```

```
a = N.array([1.0, 1.0, -1.0, -1.0, 1.0])

b = N.array([-1.0, 1.0, 1.0, -1.0, -1.0])

square = N.vstack((a,b,N.ones(N.shape(a))))  # make homogeneous coordinates



P.plot(square[0],square[1],'r-')



t = N.array([-0.5, 1],ndmin=2).transpose()

R = N.array([N.cos(N.pi/3), -N.sin(N.pi/3), N.sin(N.pi/3), N.cos(N.pi/3)])

R = R.reshape(2,2)

H = N.hstack((R,t))

H = N.vstack((H,[0,0,1.0]))



Hcircle = N.dot(H,circle)

Hsquare = N.dot(H,square)

Hcircle = Hcircle/Hcircle[2]  # transform from homogeneous coords
```

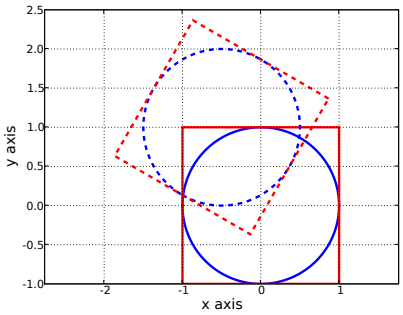


FIGURE 6.5.1. Example of a Euclidean transformation.

```
Hsquare = Hsquare/Hsquare[2]  # transform from homogeneous coords



P.plot(Hcircle[0],Hcircle[1],'b--')

P.plot(Hsquare[0],Hsquare[1],'r--')



P.show()
```

above commands, to convince yourself of the invariants of the Euclidean transformation.

6.5.2.2. *Similarity transformations.* A *similarity* transformation (or a similarity) is an isometry together with an isotropic scaling. In the case of a Euclidean transformation composed with an isotropic scaling, the form of the similarity is given by

$$\begin{bmatrix} x_1' \\ x_2' \\ 1 \end{bmatrix} = \begin{bmatrix} s\,\cos\theta & -s\,\sin\theta & t_x \\ s\,\sin\theta & s\,\cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}.$$

This can be written more concisely in block form as

$$\mathbf{x}' = H_S\mathbf{x} = \begin{bmatrix} s\,R & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \mathbf{x},$$

where s is a scalar that represents the scaling and the other symbols are the same as for an isometry.

Degrees of freedom The similarity transformation has four degrees of freedom: the same three as the isometry plus one for the scaling. We can therefore compute a similarity from two point correspondences.

Invariants

- ratio of two lengths
- angles
- ratio of areas
- parallel lines

Example. For the similarity transformation with scaling factor $s = \frac{1}{2}$, and where the matrix R and the translation vector $\mathbf{t}$ are the same as the Euclidean transformation example, we have the transformation matrix H_S given by

$$H_S = \begin{bmatrix} \frac{1}{2}\cos(\frac{\pi}{3}) & -\frac{1}{2}\sin(\frac{\pi}{3}) & -\frac{1}{2} \\ \frac{1}{2}\sin(\frac{\pi}{3}) & \frac{1}{2}\cos(\frac{\pi}{3}) & 1 \\ 0 & 0 & 1 \end{bmatrix}.$$

Below is the MATLAB code used to generate the transformation matrix H_S and to apply this transformation to the unit square and unit circle.

```
>> H = [0.5*R t; 0 0 1]
H =
    0.2500   -0.4330   -0.5000
    0.4330    0.2500    1.0000
         0         0    1.0000

>> Hcircle = H*circle; Hsquare = H*square
Hsquare =
    0.1830   -0.6830   -1.1830   -0.3170    0.1830
```

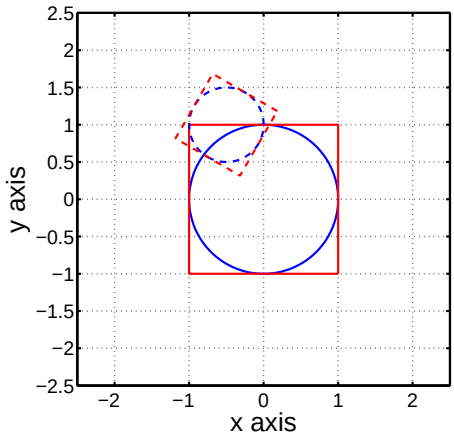


FIGURE 6.5.2. Example of a similarity transformation.

```
1.1830    1.6830    0.8170    0.3170    1.1830
1.0000    1.0000    1.0000    1.0000    1.0000
```

```
>> plot(Hcircle(1,:),Hcircle(2,:),'b--');plot(Hsquare(1,:),Hsquare(2,:),'r--')
>> axis([-2.5 2.5 -2.5 2.5]);axis square
```

transformation given in this example, to convince yourself of the invariants of the similarity transformation.

6.5.2.3. *Affine transformations.* An *affine* transformation (or an affinity) is a non-singular linear transformation followed by a translation. It is represented, in matrix form, by

$$\begin{bmatrix} x_1' \\ x_2' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix},$$

or more compactly in block form, by

$$\mathbf{x}' = H_A\mathbf{x} = \begin{bmatrix} A & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \mathbf{x},$$

where A is a non-singular 2×2, and the other symbols are as before.

DEGREES OF FREEDOM An affinity has six degrees of freedom, and so we need three point correspondences to compute H_A, the matrix of the affinity.

Invariants

- ratio of areas

  Area is scaled by $\det A = \sigma_1\sigma_2$, i.e. the ratios of areas is preserved.

- parallel lines

  Two parallel lines intersect at $\begin{bmatrix} x_1 & x_2 & 0 \end{bmatrix}^T$ (an ideal point). Under an affinity this ideal point is mapped to

$$\mathbf{x}' = \begin{bmatrix} A & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ 0 \end{bmatrix} = \begin{bmatrix} A\mathbf{y} \\ 0 \end{bmatrix},$$

  where $\mathbf{y} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$. Since the last element of $\mathbf{x}'$ is zero, it is also an ideal point. Said differently, an affinity maps parallel lines to parallel lines. Note that the transformed parallel lines do not necessarily have the same direction as the original parallel lines.

- ratio of lengths of parallel lines

  Example. For the affinity with matrix $A = \begin{bmatrix} 1 & 2 \\ -1 & \frac{1}{2} \end{bmatrix}$ and translation vector $\mathbf{t}$ the same as the previous two examples, the transformation matrix is given by

$$H_A = \begin{bmatrix} 1 & 2 & -\frac{1}{2} \\ -1 & \frac{1}{2} & 1 \\ 0 & 0 & 1 \end{bmatrix}.$$

  Below is the MATLAB code used to apply this transformation to the unit square and unit circle.

```
>> A = [1 2; -1 .5];
>> H = [A t; 0 0 1];
>> Hcircle = H*circle; Hsquare = H*square
Hsquare =
   -1.5000    2.5000    0.5000   -3.5000   -1.5000
   -0.5000    0.5000    2.5000    1.5000   -0.5000
    1.0000    1.0000    1.0000    1.0000    1.0000
```

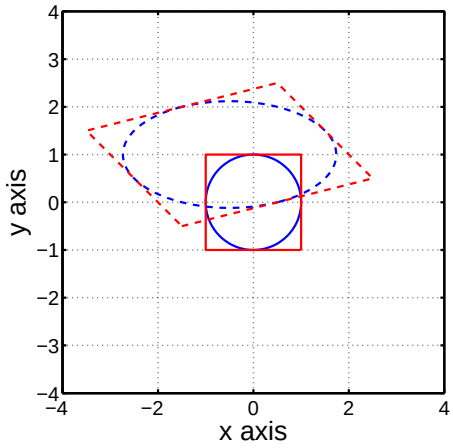


FIGURE 6.5.3. Example of a affinity transformation.

```
>> plot(Hcircle(1,:),Hcircle(2,:),'b--');plot(Hsquare(1,:),Hsquare(2,:),'r--')
>> axis([-4 4 -4 4]);axis square
```

transformation given in this example, to convince yourself of its invariants.

6.5.2.4. *Projective transformations.* A *projective* transformation (or projectivity) as defined in (6.1), is a general non-singular linear transformation of homogeneous coordinates, and generalizes an affinity, which is a non-singular linear transformation of inhomogeneous coordinates and a translation. Given in block form as

$$\mathbf{x}' = H_P\mathbf{x} = \begin{bmatrix} A & \mathbf{t} \\ \mathbf{v}^T & v \end{bmatrix} \mathbf{x},$$

where both $\mathbf{v}$ and v are not necessarily nonzero, and the other symbols are as before. Note that since v could be zero, we cannot always scale the matrix H_A in such a way that $v = 1$.

DEGREES OF FREEDOM The matrix H_A has nine elements, but since it is a homogeneous matrix, any nonzero multiple of H_A, say kH_A is equivalent to H_A, we therefore only have eight degrees of freedom. We can compute the matrix H_A from a four point correspondence (with no three points being collinear).

Invariants

– cross ratio of four points (ratio of ratios of lengths on a line)

Example. Consider the projective transformation with matrix

$$H_P = \begin{bmatrix} -1 & 2 & 1 \\ 1 & -1 & \frac{1}{2} \\ -\frac{1}{2} & 2 & -3 \end{bmatrix}.$$

Below is the MATLAB code used to apply this transformation to the unit square and unit circle.

```
>> H = [-1 2 1; 1 -1 .5; -.5 2 -3]
H =
   -1.0000    2.0000    1.0000
    1.0000   -1.0000    0.5000
   -0.5000    2.0000   -3.0000

>> Hcircle = H*circle; Hsquare = H*square
Hsquare =
   -2.0000    2.0000    4.0000         0   -2.0000
    2.5000    0.5000   -1.5000    0.5000    2.5000
   -5.5000   -1.5000   -0.5000   -4.5000   -5.5000

>> Hcircle(1,:) = Hcircle(1,:)./Hcircle(3,:);
>> Hcircle(2,:) = Hcircle(2,:)./Hcircle(3,:);
>> Hcircle(3,:) = Hcircle(3,:)./Hcircle(3,:);
>> Hsquare(1,:) = Hsquare(1,:)./Hsquare(3,:);
>> Hsquare(2,:) = Hsquare(2,:)./Hsquare(3,:);
>> Hsquare(3,:) = Hsquare(3,:)./Hsquare(3,:)
Hsquare =
    0.3636   -1.3333   -8.0000         0    0.3636
   -0.4545   -0.3333    3.0000   -0.1111   -0.4545
    1.0000    1.0000    1.0000    1.0000    1.0000
```

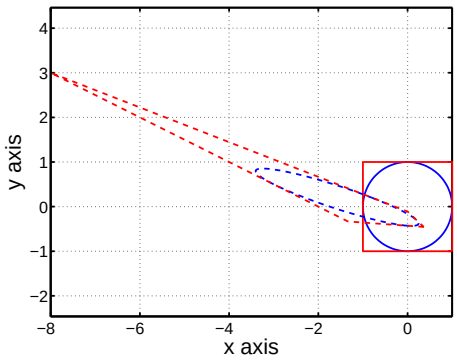


FIGURE 6.5.4. Example of a projective transformation.

```
>> plot(Hcircle(1,:),Hcircle(2,:),'b--');plot(Hsquare(1,:),Hsquare(2,:),'r--')
```

`Hcircle` and `Hsquare` in order to plot the transformed objects in the plane. This rescaling was not necessary in the previous example since the last row of the transformed objects' consisted of ones.

Use Figure 6.5.4, an illustration of the projective transformation given in this example, to convince yourself of that the invariants listed for the previous transformations (for example parallel lines) are not invariants for this transformation.

REMARK. If a projective transformation preserves parallel lines, it is an affinity. To prove this, suppose a projective transformation is such that parallel lines are mapped to parallel lines, then

$$\begin{bmatrix} A & \mathbf{t} \\ \mathbf{v}^T & v \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ 0 \end{bmatrix} = \begin{bmatrix} A\mathbf{x} \\ \mathbf{v}^T\mathbf{x} \end{bmatrix} = \begin{bmatrix} x_1' \\ x_2' \\ 0 \end{bmatrix}.$$

Then it must hold that $\mathbf{v}^T\mathbf{x} = 0$ for all $\mathbf{x}$, so that $\mathbf{v}$ must necessarily be equal to the zero vector. Since the matrix must be non-singular, it must then also hold that $v \neq 0$, so that the transformation matrix is of the form

$$\begin{bmatrix} A & \mathbf{t} \\ \mathbf{v}^T & v \end{bmatrix}$$

for a nonzero constant v. The transformation is therefore an affinity.

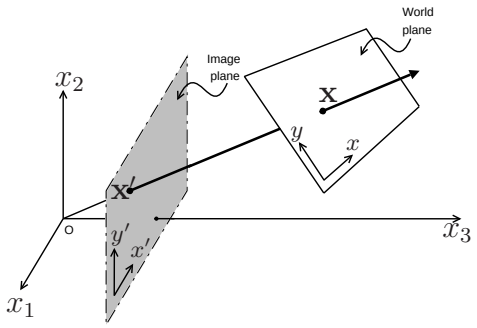


FIGURE 6.5.5. Transform from world plane to image plane.

6.5.3. Removing perspective distortion from a plane. Consider the *central projection* in Figure 6.5.5. Projection along rays though a common point (the center of the projection) defines a mapping from one plane to another, say the world plane to the image plane. It maps lines to lines, and so is a projective transformation. If a coordinate system is defined in each of the planes, say world coordinates and image coordinates, and points are represented in homogeneous coordinates, then the central projection mapping may be expressed as $\mathbf{x}' = H\mathbf{x}$, where H is a non-singular 3×3 matrix. This transformation is called a *perspective transformation* and is actually more restricted than a general projective transformation, in that it only has six degrees of freedom.

Shape is distorted under such a perspective transformation, but since the image plane is related to the world plane via a projective transformation, we can undo this distortion. This is done by computing the inverse projective transformation and applying it to the image, resulting in an image where the objects have their correct geometric shape.

In order to compute this inverse transformation, we choose local inhomogeneous coordinates $\mathbf{x} = (x, y)$ and $\mathbf{x}' = (x', y')$—both measured directly from the world and the image plane, respectively. Then, the projective

transformation

$$\begin{bmatrix} x_1' \\ x_2' \\ x_3' \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix},$$

can be written in inhomogeneous form as

$$x' = \frac{x_1'}{x_3'} = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}}, \qquad y' = \frac{x_2'}{x_3'} = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}}.$$

Each point correspondence leads to two linear equations of the form

$$\begin{aligned} x'\big(h_{31}x + h_{32}y + h_{33}\big) &= h_{11}x + h_{12}y + h_{13} \\ y'\big(h_{31}x + h_{32}y + h_{33}\big) &= h_{21}x + h_{22}y + h_{23} \end{aligned}$$

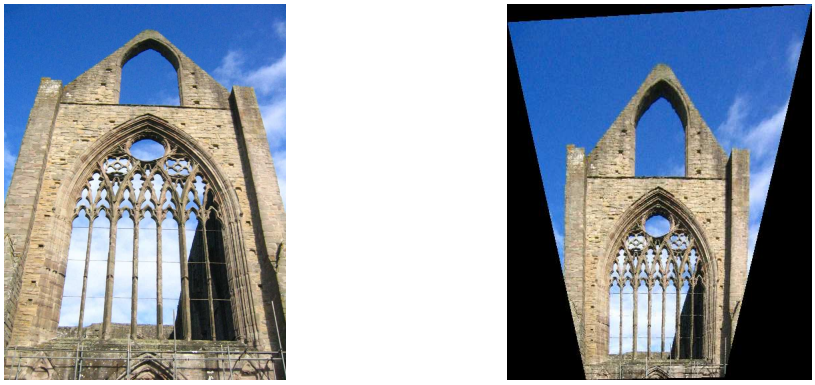
or equivalently,

$$\begin{bmatrix} x & y & 1 & 0 & 0 & 0 & -x'x & -x'y & -x' \\ 0 & 0 & 0 & x & y & 1 & -y'x & -y'y & -y' \end{bmatrix} \begin{bmatrix} h_{11} \\ h_{12} \\ \vdots \\ h_{33} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

Since H has eight degrees of freedom and each point correspondence results in two linear equations, we need four point correspondences to solve for H. Of course, if one has more than four point correspondences it is even better. The resulting overdetermined system for the null space can then be solved in a least-squares sense. Since measurement errors corrupt the system, the rank of the coefficient matrix will now be 9, but with a small 9th singular value. The 9th singular vector is therefore used as an approximation for the null space.

For four point correspondences, we get the system of equations $A\mathbf{h} = \mathbf{0}$, where A is an 8×9 matrix and $\mathbf{h}^T = \begin{bmatrix} h_{11} & h_{12} & \cdots & h_{33} \end{bmatrix}$. Choose points so that A has full rank, i.e. A is of rank 8, this means that the point correspondences should be chosen such that no three points are collinear.

The vector $\mathbf{h}$ is then a vector in the null space of the matrix A, or any nonzero multiple of it. One way of obtaining $\mathbf{h}$ is to compute the SVD of



(a) Distorted image. (b) Corrected image.

FIGURE 6.5.6. Removal of perspective distortion.

the matrix A, i.e. $A = U\Sigma V^T$. Then the last column of V will be a basis for the null space of A, so that we can choose $\mathbf{h}$ as the last column of V.

Applying this to a perspective distorted image, we can correct the distortion, for example in Figure 6.5.6(a) we recognize that the wall of the ruin forms a rectangle in world coordinates. We then choose the points $\mathbf{x}_i = \begin{bmatrix} x_i & y_i, \ i = 1, \ldots, 4 \end{bmatrix}$ as four points on the ruin and map them to the four corners of a rectangle to get the projective transformation H. Once we have H, we apply its inverse to the entire distorted image, to obtain the rectified image Figure 6.5.6(b).

REMARK. The computation of the matrix H does not require any knowledge of the camera's parameters, for example the focal length, or the orientation of the plane

6.6. The pinhole camera.

6.6.1. Basic pinhole model. Here we consider the central projection of a point $\mathbf{X} = \begin{bmatrix} X & Y & Z \end{bmatrix}^T$ in the camera coordinate system, with origin at the camera center $\mathbf{C}$, onto the image plane. The image plane is located at $Z = f$ in the camera coordinate system where f is known as the focal length of the camera. The point where the Z-axis pierces the image plane is known as the principal point and the Z-axis as the principal axis. The origin of the image coordinate system is for now, chosen as the principal point and its x- and y axes are aligned with the X- and Y axes

of the camera coordinate system. All of this is illustrated in Figure 6.6.1. If a point $\mathbf{X} \in \mathbb{R}^3$ has coordinates $\begin{bmatrix} X & Y & Z \end{bmatrix}^T$ relative to the camera coordinate system, $\mathbf{X}$ projects onto the point $\mathbf{x}$ on the image plane, with $\mathbf{C}$ the center of the projection, see Figure 6.6.1. Using homogeneous coordinates this projection is described by a matrix P. We'll variously refer to this matrix as the camera matrix or the projection matrix, depending on which aspect we wish to emphasize.

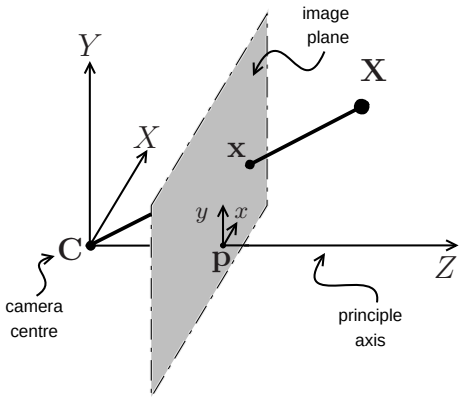


FIGURE 6.6.1. Pinhole camera geometry.

Using similar triangles, it follows immediately that $\mathbf{x} = \begin{bmatrix} f\frac{X}{Z} & f\frac{Y}{Z} \end{bmatrix}^T$ (in Euclidean coordinates). This is a nonlinear map, however, that becomes linear if homogeneous coordinates are used.

6.6.2. Central projection using homogeneous coordinates. Realising that

$$\begin{bmatrix} f\frac{X}{Z} \\ f\frac{Y}{Z} \end{bmatrix}$$

is written in homogeneous coordinates as

$$\begin{bmatrix} fX \\ fY \\ Z \end{bmatrix}$$

the map from homogeneous camera coordinates to homogeneous image coordinates is given by

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}.$$

The matrix in this expression can be written as $\mathrm{diag}(f, f, 1)\big[I \mid \mathbf{0}\big]$ where

$$\mathrm{diag}(f, f, 1) = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

i.e. the diagonal matrix with $(f, f, 1)$ on the diagonal and

$$[I \mid \mathbf{0}] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

Using this notation, we can describe the central projection from $\mathbf{X}$ to $\mathbf{x}$ as

$$\mathbf{x} = P\mathbf{X}$$

where P is the 3×4 homogeneous *camera projection matrix*. This defines the camera matrix for the central projection as

$$P = \mathrm{diag}(f, f, 1)\,[I \mid \mathbf{0}]\,.$$

6.6.3. Principal point offset. The camera matrix derived above assumes that the origin of the image coordinate system is at the principal point $\mathbf{p}$, however, this is not usually the case in practice. If the coordinates of the principal point $\mathbf{p}$ are (p_x, p_y) in the image coordinate system, see Figure 6.6.2, then the mapping of $\mathbf{X}$ to $\mathbf{x}$ is given by

$$P : \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \longrightarrow \begin{bmatrix} f\frac{X}{Z} + p_x \\ f\frac{Y}{Z} + p_y \end{bmatrix},$$

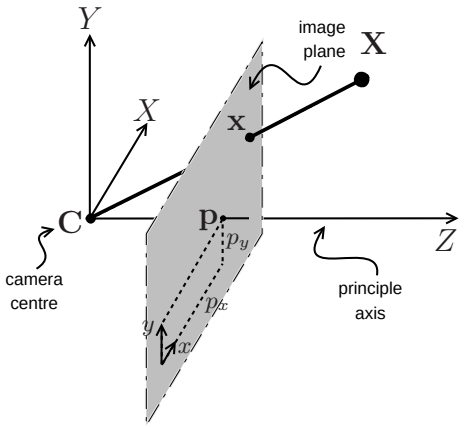


FIGURE 6.6.2. Pinhole camera geometry with offset image coordinates.

or equivalently in homogeneous coordinates by

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{bmatrix} = \begin{bmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}.$$

Introducing the *calibration matrix*

$$K = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix}$$

the camera matrix P is given by

$$P = K\,[I \mid \mathbf{0}]\,.$$

Emphasizing the fact that we are projecting features described in terms of the camera coordinate system, we rewrite the projection as

$$\mathbf{x} = K\,[I \mid \mathbf{0}]\,\mathbf{X}_{\text{cam}}. \tag{6.1}$$

The next step is to introduce the world coordinate system and relate it to the camera coordinate system.

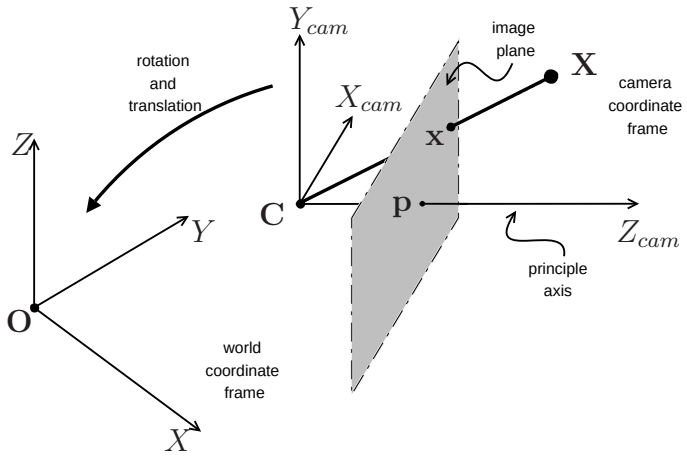


FIGURE 6.6.3. Pinhole camera with general geometry.

6.6.4. The world coordinate system. In general, 3D objects are described in terms of coordinate systems fixed to the objects as shown in Figure 6.6.3. In homogeneous coordinates it is given by

$$\mathbf{X} = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}.$$

Since we already know how to project a feature in the camera coordinate system onto the image coordinate system, we only need to relate the world– and camera coordinate systems, i.e. $\mathbf{X}$ and $\mathbf{X}_{\text{cam}}$. Since the two coordinate systems are related by a rotation and a translation, see Figure 6.6.3 we may write $\widetilde{\mathbf{X}}_{\text{cam}} = R\left(\widetilde{\mathbf{X}} - \widetilde{\mathbf{C}}\right) = R\widetilde{\mathbf{X}} - \mathbf{t}$, where the tilde denotes Euclidean coordinates, $\mathbf{X} = \begin{bmatrix} \widetilde{\mathbf{X}} \\ 1 \end{bmatrix}$ for example, $\widetilde{\mathbf{C}}$ is the coordinates of the camera center in the world coordinate system, and R is a 3×3 rotation matrix describing the rotation of the world coordinate system relative to the camera coordinate system. Also note that $\widetilde{\mathbf{X}}_{\text{cam}} = \mathbf{0}$ if $\widetilde{\mathbf{X}} = \widetilde{\mathbf{C}}$, i.e. the camera coordinate is zero at the camera center, as expected. If we use $\widetilde{\mathbf{X}}_{\text{cam}}$ and $\mathbf{X}$ to denote the homogeneous representations of $\widetilde{\mathbf{X}}_{\text{cam}}$ and $\widetilde{\mathbf{X}}$ respectively, then we

have

$$\widetilde{\mathbf{X}}_{\text{cam}} = \begin{bmatrix} R & -R\widetilde{\mathbf{C}} \\ \mathbf{0}^T & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} R & -R\widetilde{\mathbf{C}} \\ \mathbf{0}^T & 1 \end{bmatrix} \mathbf{X}. \tag{6.2}$$

Combining this with (6.1), we get

$$\mathbf{x} = KR \left[I \mid -\widetilde{\mathbf{C}} \right] \widetilde{\mathbf{X}},$$

where $\widetilde{\mathbf{X}}$ is now given in the world coordinate system.

Degrees of freedom. We note that the general pinhole camera has nine degrees of freedom:

- three for the calibration matrix K (the elements f, p_x and p_y);
- three for the rotation matrix R and
- three for the camera center $\widetilde{\mathbf{C}}$.

Note that all the parameters that refer to the specific type of camera are contained in K; these parameters are referred to as the intrinsic parameters, and K is referred to as the calibration matrix. R and $\widetilde{\mathbf{C}}$ describe the external orientation of the world coordinate system to the camera coordinate system and are therefore referred to as the extrinsic parameters.

6.6.5. CCD cameras. In the models above we assume that the image coordinate frame is Euclidean with equal scales in both axial directions, which is not always true. In the case of CCD cameras there is the additional possibility of having rectangular pixels. In particular, if the number of pixels per unit distance in image coordinates are m_x and m_y in the x and y directions, respectively, then the calibration matrix becomes

$$K = \begin{bmatrix} m_x & 0 & 0 \\ 0 & m_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \alpha_x & 0 & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}$$

where $\alpha_x = fm_x$ and $\alpha_y = fm_y$ represent the focal length of the camera in terms of pixel coordinates in the x and y directions, respectively. Similarly, (x_0, y_0) is the

principal point in terms of pixel coordinates with $x_0 = m_x p_x$ and $y_0 = m_y p_y$. A CCD camera thus has ten degrees of freedom.

6.6.6. Finite projective camera. For added generality, we use a calibration matrix of the form

$$K = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix},$$

where the added parameter s is referred to as the *skew* parameter. The skew parameter will be zero for most normal cameras, but has some important applications.

With this calibration matrix K, we call the camera

$$P = KR \begin{bmatrix} I \mid -\widetilde{\mathbf{C}} \end{bmatrix}$$

a *finite projective camera*. A finite projective camera has eleven degrees of freedom: one more than the CCD camera to make provision for the skew parameter. This also reflects the fact that P is a 3×4 homogeneous matrix.

Note that the 3×3 sub-matrix KR is non-singular (det $P \neq 0$). Conversely, any 3×4 matrix for which the left hand 3×3 sub-matrix is non-singular, is a finite camera matrix.

EXERCISE 14. Given a finite projective camera, i.e. a 3×4 homogeneous matrix $P = \begin{bmatrix} M & \mathbf{t} \end{bmatrix}$ with non-singular 3×3sub-matrix M, find an algorithm based on the QR factorization to compute the decomposition of P into K, R and $\widetilde{\mathbf{C}}$.

6.6.7. General projective camera. The final step in this hierarchy of projective cameras is to remove the condition of non-singularity on the left hand 3×3 sub-matrix. A *general projective* camera is one represented by an arbitrary homogeneous 3×4 matrix of rank 3. It has eleven degrees of freedom. We need a rank 3 matrix, since if the rank is 1 or 2, the range of the projective transformation is a line or point and not the whole plane, i.e. a 3D scene will be projected onto a point or a line. The orthographic camera that will be encountered in Chapter 32 is an example of an infinite camera. We now briefly discuss the difference.

Since the camera matrix P has rank 3, it means that it has a one dimensional null space spanned by $\mathbf{y}$ where

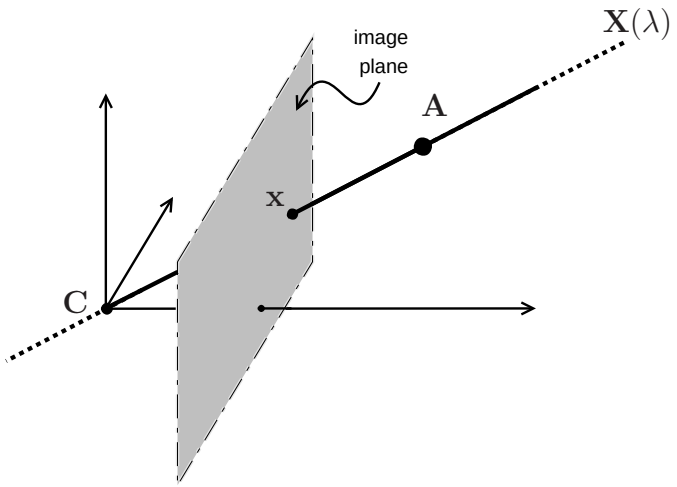


FIGURE 6.6.4. The ray through **A** and the camera center.

$$P\mathbf{y} = \mathbf{0}.$$

We can also say that the null space of P consists of a single homogeneous vector. It turns out that this homogeneous vector has special meaning—it is the camera center in the world coordinate system, expressed in homogeneous coordinates. To show this, suppose that $\mathbf{a}$ is an arbitrary point in 3-space, and $\mathbf{y}$ is in the null space of P. Then the line in 3-space through $\mathbf{a}$ and $\mathbf{y}$ is given (in homogeneous coordinates) by

$$\mathbf{X}(\lambda) = \lambda\mathbf{a} + (1-\lambda)\mathbf{y}.$$

Each point $\mathbf{X}(\lambda)$ on this line maps to

$$\mathbf{x}(\lambda) = P\mathbf{X}(\lambda) = P\left(\lambda\mathbf{a} + (1-\lambda)\mathbf{y}\right) = \lambda P\mathbf{a} + (1-\lambda)P\mathbf{y} = \lambda P\mathbf{a},$$

which is the same as $P\mathbf{a}$ in homogeneous coordinates. Thus the whole ray through $\mathbf{y}$ and $\mathbf{a}$ maps to a single point $P\mathbf{a}$. Thus $\mathbf{X}(\lambda)$ is a ray through the *camera center* $\mathbf{C}$. Since this holds for any choice of $\mathbf{a}$, it follows that $\mathbf{y} = \mathbf{C}$ the camera center, see Figure 6.6.4.

Remarks:

(1) It is not surprising that a vector in the null space of P is the camera center of P, since the image point $\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = P\mathbf{C}$ is not defined, and the camera center is the unique point for which the image is not defined.

(2) For finite cameras one can calculate the null space of P directly. Moreover, since the homogeneous representation of the camera center is $\mathbf{C} = \begin{bmatrix} \widetilde{\mathbf{C}} \\ 1 \end{bmatrix}$ it follows that
$$P\mathbf{C} = KR\left[I \mid -\widetilde{\mathbf{C}}\right]\begin{bmatrix} \widetilde{\mathbf{C}} \\ 1 \end{bmatrix} = \mathbf{0},$$
which shows that the camera center in homogeneous coordinates, is indeed the null space of P.

(3) If P is not a finite camera we write it as $P = [M \mid \mathbf{p}_4]$, where M is a singular 3×3 matrix of rank 2, and $\mathbf{p}_4$ denotes the fourth column of P. Since M is singular of rank 2, there exists a unique homogeneous vector $\mathbf{d}$ such that $M\mathbf{d} = \mathbf{0}$, and for $\mathbf{C} = \begin{bmatrix} \mathbf{d}^T & 0 \end{bmatrix}^T$ it follows that $P\mathbf{C} = \mathbf{0}$. Thus
$$\mathbf{C} = \begin{bmatrix} \mathbf{d} \\ 0 \end{bmatrix}$$
is the unique camera center of P, proving that if M is singular the camera center is at infinity.

6.7. Camera calibration.

In this section we discuss two aspects of calculating the camera matrix P: firstly, given a number of point correspondences, we use our linear model to derive a set of equations that generate the matrix P, then we consider the nonlinear effect of radial distortion.

6.7.1. Calibration objects. Suppose we have a real-life object and we know the $3D$ coordinates of n points on the object in some world coordinate system. If for example you are asked to reconstruct a big industrial plant, you may attach a

number of markers on the plant as shown in Figure 6.7.1, you might hire a surveyor to precisely measure the $3D$ coordinates of your markers.

FIGURE 6.7.1. Industrial plant with markers attached.

In other applications you may construct a calibration object to precise specifications. Basically a calibration object is an object such as a cube with easily identifiable markers in known positions. The calibration object used for the patient positioning system is shown in Figure 6.7.2.

Apart from knowing the exact $3D$ positions of the markers, one also wants to detect them automatically and reliably. Very often calibration objects with a chess board pattern is used in which case one has to automatically detect the corners on the chess board for which something like the Harris corner detector is quite useful. In case you need to build your own calibration object, we suggest you use Lego bricks as shown in Figure 6.7.3.

Lego bricks are made to precise specifications and measure $32 \times 16 \times 9.6$mm. The only problem is that the corners get worn down with use and it becomes a bit more of an issue to locate them accurately. Nevertheless, for most purposes it works fine.

Note that the calibration object needs to span $3D$space. A planar calibration object causes leads to singularities since it does not contain sufficient information.

Is it necessary t
give more info
mation?

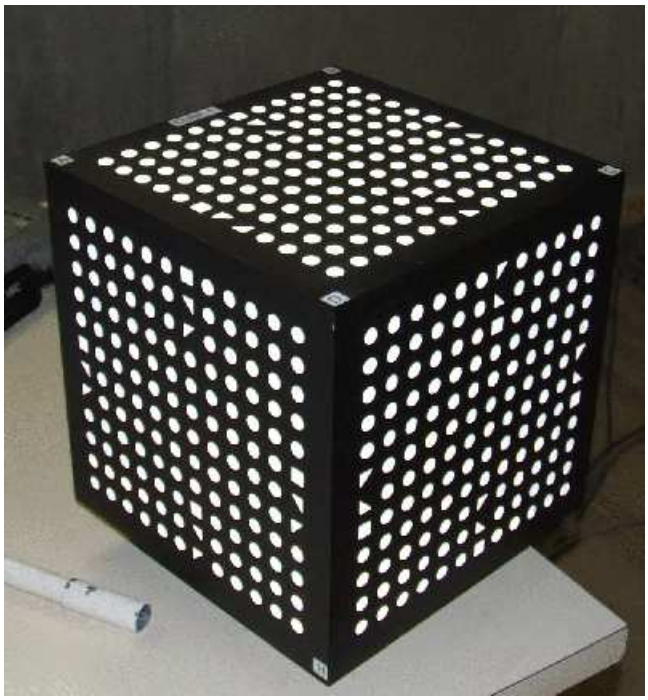


FIGURE 6.7.2. Calibration object for patient positioning.

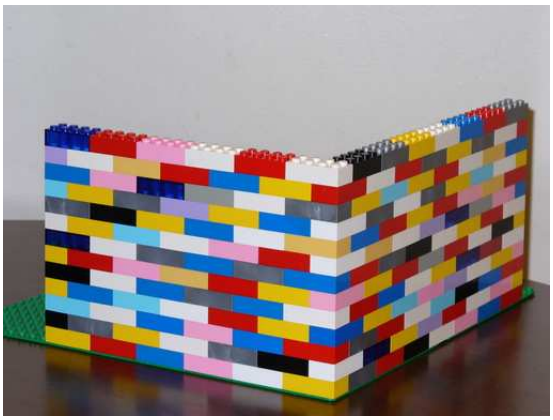


FIGURE 6.7.3. Lego calibration object.

6.7.2. Calculating the camera matrix. We can now assume that we have available a calibration object and that we know the $3D$ coordinates $\mathbf{X}_j,\ j = 1, \ldots, n$ of n features in a world coordinate system. Note that the world coordinate system

is important. It is the coordinate system in which we eventually reconstruct our $3D$ object. The camera matrix P is still unknown but it projects the calibration features to images features $\mathbf{x}_j,\ j = 1, \ldots, n$, that we can identify in the image. (The calibration object was constructed for exactly this purpose.) Thus we know that the known calibration features and their known projections onto the image plane are related through,

$$\mathbf{x}_j = P\mathbf{X}_j,\ j = 1, \ldots, n, \tag{6.1}$$

where all the quantities are expressed in homogeneous coordinates. Since P is an homogeneous matrix, defined only up to an arbitrary scale, it has only 11 degrees of freedom. We'll shortly find that each point correspondence gives us two equations. Thus at least $5\frac{1}{2}$ point correspondences are needed to calculate P. Setting up the necessary equations to solve for P is a little tricky only because we are working in homogeneous coordinates. The correspondences given by (6.1) are not equalities in the usual sense. Since we work with homogeneous coordinates they only mean that $\mathbf{x}_i$ and $P\mathbf{X}_i$ point in the same direction. This fact can be simply expressed in terms of the vector cross product as $\mathbf{x}_i \times P\mathbf{X}_i = \mathbf{0}$. This allows us to derive a simple linear solution for the entries of the camera matrix P. [1]

If we write the camera matrix in terms of its rows, as

$$P = \begin{bmatrix} \mathbf{r}_1^T \\ \mathbf{r}_2^T \\ \mathbf{r}_3^T \end{bmatrix} \text{ and } \mathbf{x}_j = \begin{bmatrix} x_j \\ y_j \\ 1 \end{bmatrix},$$

then

$$P\mathbf{X}_i = \begin{bmatrix} \mathbf{r}_1^T \\ \mathbf{r}_2^T \\ \mathbf{r}_3^T \end{bmatrix} \mathbf{X}_i = \begin{bmatrix} \mathbf{r}_1^T\mathbf{X}_i \\ \mathbf{r}_2^T\mathbf{X}_i \\ \mathbf{r}_3^T\mathbf{X}_i \end{bmatrix},$$

[1] The similarity of this problem with that of computing the 2D projective transformation in Project 3, is evident.

and the vector cross product becomes

$$\begin{bmatrix} \mathbf{0}^T & -\mathbf{X}_i^T & y_i\mathbf{X}_i^T \\ \mathbf{X} & \mathbf{0}^T & -x_i\mathbf{X}_i^T \\ -y_i\mathbf{X}_i^T & x_i\mathbf{X}_i^T & \mathbf{0}^T \end{bmatrix} \begin{bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \\ \mathbf{r}_3 \end{bmatrix} = \mathbf{0}. \tag{6.2}$$

This equation has the form $A_i\mathbf{r} = \mathbf{0}$, where A_i is a 3×12 matrix, and $\mathbf{r}$ is a 12 vector made up of the entries of the camera matrix P.

(1) The equation $A_i\mathbf{r} = \mathbf{0}$ is a linear equation in the unknown $\mathbf{r}$.
(2) Although there are three equation in (6.2), only the two of them are linearly independent, so that (6.2) reduces to

$$\begin{bmatrix} \mathbf{0}^T & -w_i\mathbf{X}_i^T & y_i\mathbf{X}_i^T \\ w_i\mathbf{X}_i^T & \mathbf{0}^T & -x_i\mathbf{X}_i^T \end{bmatrix} \begin{bmatrix} \mathbf{r}_1 \\ \mathbf{r}_2 \\ \mathbf{r}_3 \end{bmatrix} = \mathbf{0}.$$

From a set of n correspondences we obtain $2n$ equations, and a $2n \times 12$ matrix A, by stacking up these $2n$ equations.
(3) The camera matrix P has 11 degrees of freedom, so that we need at least five and a half point correspondences to solve for $\mathbf{r}$ in $A\mathbf{r} = \mathbf{0}$.
(4) In practice, the rule of thumb for good estimation of a camera matrix is that the number of constraints should exceed the number of unknowns by a factor 5, so that in this case we would use 28 constraints (14 point correspondences) and use the SVD to find the best approximation to the null space of A.

Exercise 15. Show that the three equations of (6.2) are linearly dependent.

Problem 16. How would you go about solving for the null space if you use more than the minimum number ($5\frac{1}{2}$) of corresponding points? It is a really good idea to use many more, well separated over the whole image.

6.7.3. Radial distortion. Until now, we have assumed that our cameras are linear, i.e. that the relationship between world– and image coordinates satisfy the linear relationship (32.4). For real lenses this is not true since the lenses introduce nonlinear distortions. The most important nonlinear effect is that of radial distortion. This is where a camera lens looses accuracy towards its edges, which causes straight lines, especially close to the edges of the image, to bend. An example of severe radial

(a) Distorted image. (b) Corrected image.

FIGURE 6.7.4. Removal of radial distortion.

distortion is shown in the left hand image of Figure 6.7.4. Note for example, the distortion of the line where the wall meets the ceiling. In the right hand image of the Figure the radial distorting has been corrected and the line now appears as being straight.

The idea is that one should first correct an image for radial distortion and then apply the theory derived in the previous lectures.

The correction for radial distortion takes place on the image plane. Suppose that the corrected, undistorted coordinates are given in pixel coordinates by $\begin{bmatrix} \widehat{x} \\ \widehat{y} \end{bmatrix}$, and the measured, distorted coordinates by $\begin{bmatrix} x \\ y \end{bmatrix}$. Assuming only radial distortion, we write the relationship between the distorted and undistorted coordinates as

$$\begin{bmatrix} \widehat{x} \\ \widehat{y} \end{bmatrix} = \begin{bmatrix} x_c \\ y_c \end{bmatrix} + L(r) \begin{bmatrix} x - x_c \\ y - y_c \end{bmatrix}, \tag{6.3}$$

where $r = \sqrt{(x - x_c)^2 + (y - y_c)^2}$ is the radial distance from the center $\begin{bmatrix} x_c \\ y_c \end{bmatrix}$ of the radial distortion. Note that the distortion factor $L(r)$ is a function of radius r only. Also note that if the aspect ratio (pixel size in x and y direction) is not unitary, it has to be taken into account.

The distortion factor $L(r)$ is only defined for positive values of r and satisfies $L(0) = 1$. For an arbitrary function $L(r)$, we may use a Taylor expansion

$$L(r) = 1 + \kappa_1 r + \kappa_2 r^2 + \kappa_3 r^3 + \cdots ,$$

to approximate the distortion factor. An optimization procedure is used to estimate the distortion center $\begin{bmatrix} x_c \\ y_c \end{bmatrix}$ as well as the expansion coefficients. The distortion parameters,

$$\mathbf{r}_p = \{\kappa_1, \kappa_2, \kappa_3, \ldots, x_c, y_c\} \tag{6.4}$$

are then considered part of the internal calibration of the camera.

In order to remove radial distortion in practice one has to identify one, preferably more, straight lines in the scene. Or rather lines that are supposed to be straight but are radially distorted. The radial distortion is corrected by the transformation (6.3), and we need to calculate its parameters (6.4). Choose a point $\begin{bmatrix} x & y \end{bmatrix}^T$ on the distorted curve in the image. Use an optimization procedure ones minimizes the distance between $\begin{bmatrix} \widehat{x} & \widehat{y} \end{bmatrix}^T$ in (6.3) and the straight line connecting the end points of the line. As initial guess set $\kappa_j = 0$ and the the distortion center equal to the center of the image. How many distortion parameters one use depends on the accuracy requirements of the application. Using only κ_1 and setting the rest equal to zero already gives pretty good results.

:

6.8. Triangulation.

Given two calibrated cameras P and P' as well as a pair of corresponding points $\mathbf{x}$ and $\mathbf{x}'$ the idea is to calculate the $3D$ feature $\mathbf{X}$ where

$$\begin{aligned} \mathbf{x} &= P\mathbf{X}, \\ \mathbf{x}' &= P'\mathbf{X}. \end{aligned}$$

As before these equations cannot be solved directly for $\mathbf{X}$ since we are working in homogeneous coordinates, i.e. the equality signs merely indicate that the two sides of the equality sign point in the same direction. Taking cross products as usual in

this situation, e.g. $\mathbf{x} \times P\mathbf{X} = \mathbf{0}$, we get

$$\begin{aligned} y\mathbf{p_3^T}\mathbf{X} - z\mathbf{p_2^T}\mathbf{X} &= 0 \\ z\mathbf{p_1^T}\mathbf{X} - x\mathbf{p_3^T}\mathbf{X} &= 0 \\ x\mathbf{p_2^T}\mathbf{X} - y\mathbf{p_1^T}\mathbf{X} &= 0, \end{aligned}$$

where the $\mathbf{p}_i^T$ are the rows of P. Taking the first two equations of each of the two camera systems, we get a system of the form

$$A\mathbf{X} = \mathbf{0}, \tag{6.1}$$

where

$$A = \begin{bmatrix} y\mathbf{p}_3^T - z\mathbf{p}_2^T \\ z\mathbf{p}_1^T - x\mathbf{p}_3^T \\ y'\mathbf{p}_3'^T - z'\mathbf{p}_2'^T \\ z'\mathbf{p}_1'^T - x'\mathbf{p}_3'^T \end{bmatrix}.$$

Note that we measure the image coordinates directly in Euclidean (pixel) coordinates. Converting to homogeneous coordinates simply means that $z = 1 = z'$.

Again we calculate the best approximation of the null space of A using the SVD.

These are the basics for the reconstruction of $3D$ objects from stereo images. A few remarks however, are in order.

(1) We have glossed over the problem of automatically identifying corresponding points in the stereo image pair. One possibility is to use something like the Scale-invariant Feature Transform (SIFT) algorithm developed by David Lowe in 1999. Another possibility is to use an elegant construction based on the so-called epipolar geometry of the stereo setup that reduces the problem to searching for the corresponding point along a line, the epipolar line.

(2) Solving for the system (6.1) amounts to finding the intersection of the rays through the camera center and the image feature. Unfortunately, as Figure ? suggests, two lines in $3D$ space do not in general intersect. Because of measurement errors our system (6.1) has no solution! That is why we have been rather careful to say we are looking for the best approximation to the null space as provided by the SVD. The problem is that there is not a natural way to say what we mean by 'best' approximation. In projective geometry

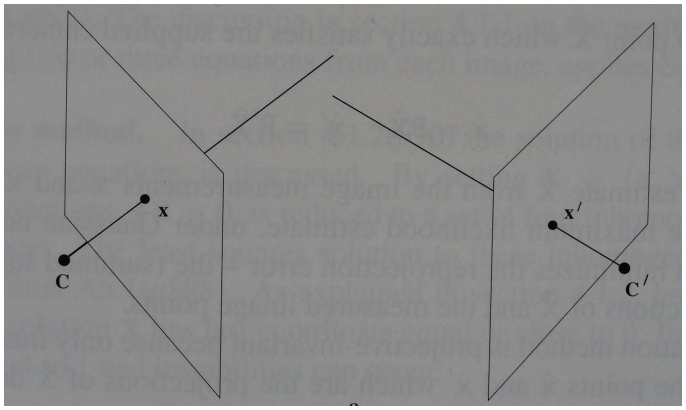


FIGURE 6.8.1. Lines in $3D$ do not in general intersect.

distance is not an invariant and cannot be used to define 'best' approximations. Hartley and Zisserman uses something they call the geometric error. In short it amounts to minimizing the error in pixel coordinates. Since this is measured in an Euclidean coordinate system, it is well defined. In order to pull it off one has again to resort to epipolar geometry.

(3) Epipolar geometry is quite straightforward and the interested reader should have no problem to learn it from Hartley and Zisserman (our personal favorite reference on everything related to stereo vision).

Part 2

ANALYTICAL TECHNIQUES

CHAPTER 7

FOURIER SERIES/TRANSFORMS

7.1. Introduction.

We will discuss four main versions of Fourier series / transforms—all closely related to each other. The four are sketched out in Figure 7.1.1:

	Type	**Domain**	**Description**
i.	Continuous	$[-\infty, \infty]$	Fourier transform
ii.	Continuous	$[-\pi, \pi]$	Fourier series
iii.	Discrete	$[-\pi, \pi]$	DFT—discrete Fourier transform
iv.	Discrete	$[0, N]$	DFT—case handled by FFT algorithm

Of these, we have used case i. in the theory for CT (computerized tomography). Effective numerical implementation required case iv and the FFT algorithm. The order of the four cases in Figure 7.1.1 reflect that, in some sense, i. can be seen as the most fundamental situation, of which ii., iii., and iv. are successively more restricted special cases. The conceptually simplest case is probably case ii.—the Fourier series of a continuous function. Hence, that case is described first, in Section 7.2. The subsequent Sections 7.3 and 7.4 cover the remaining cases. Generalizations to 2-D are discussed in Section 7.5.

7.2. Fourier series.

With very mild restrictions, any periodic function can be represented as a superposition (sum) of sines and cosines. This result, due to Joseph Fourier (1768-1830) is one of the most fundamental in all of mathematics. In the case of $[-\pi, \pi]$-periodicity, the key result can be written as

TYPE OF DATA **In all cases, assumed complex**	FORM OF FOURIER REPRESENTATION	TRANSFORM **Formulas for Fourier coefficients**
CONTINUOUS, $(-\infty, \infty)$ 	$u(x) = \int_{-\infty}^{\infty} \hat{u}(\omega)\, e^{i\omega x}\, d\omega$ **FOURIER TRANSFORM** *EVERY mode* $e^{i\omega x}$, $-\infty < \omega < \infty$ *possible*	$\hat{u}(\omega) = \frac{1}{2\pi} \int_{-\infty}^{\infty} u(x)\, e^{-i\omega x}\, dx$
CONTINUOUS, $(-\boldsymbol{\pi}, \boldsymbol{\pi})$ 	$u(x) = \sum_{k=-\infty}^{\infty} \hat{u}_k\, e^{ikx}$ **FOURIER SERIES** *ONLY modes that are* $2\pi -$ *periodic are possible*	$\hat{u}_k = \frac{1}{2\pi} \int_{-\pi}^{\pi} u(x)\, e^{-ikx}\, dx$
DISCRETE, $(-\boldsymbol{\pi}, \boldsymbol{\pi})$, **N points** 	$u(x) = \sum_{k=0}^{N-1} \hat{u}_k\, e^{ikx}$ **DISCRETE FOURIER TRANSFORM** *ONLY N modes present*	$\hat{u}_k = \frac{1}{N} \sum_{j=0}^{N-1} u(x_j)\, e^{-ikx_j}$ (*Grid points* $x_j = -\pi + 2\pi j/N$, $j = 0, 1, ..., N-1$)
DISCRETE, $(\mathbf{0}, \mathbf{N})$, **N points** 	$u_j = \sum_{k=0}^{N-1} \hat{u}_k\, e^{2\pi ikj/N}$ **DISCRETE FOURIER TRANSFORM** *N modes (case handled by FFT algorithm)*	$\hat{u}_k = \frac{1}{N} \sum_{j=0}^{N-1} u_j\, e^{-2\pi ikj/N}$ (*case handled by FFT algorithm*)

FIGURE 7.1.1. Comparisons between different types of Fourier expansions.

$$u(x) = a_0 + \sum_{k=1}^{\infty} a_k \cos kx + \sum_{k=1}^{\infty} b_k \sin kx \quad . \tag{7.1}$$

Supposing this result is true, it is easy to find the coefficients from the function

$$\left.\begin{array}{l} a_0 = \frac{1}{2\pi}\int_{-\pi}^{\pi} u(x)\ dx \\ a_k = \frac{1}{\pi}\int_{-\pi}^{\pi} u(x)\cos kx\ dx \\ b_k = \frac{1}{\pi}\int_{-\pi}^{\pi} u(x)\sin kx\ dx \end{array}\right\} \quad , \quad k = 1, 2,\ \ldots \tag{7.2}$$

For example, to find b_n for some integer n, we consider

$$\int_{-\pi}^{\pi} u(x)\sin nx\ dx = a_0 \sin nx\ dx \quad + \quad \sum_{k=1}^{\infty} a_k \int_{-\pi}^{\pi} u(x)\cos kx\ \sin nx\ dx + \tag{7.3}$$

$$+ \quad \sum_{k=1}^{\infty} b_k \int_{-\pi}^{\pi} u(x)\sin kx\ \sin nx\ dx \tag{7.4}$$

Because of

$$\begin{aligned} \cos kx\ \sin nx &= \frac{1}{2}\ \sin(k+n)x\ - \frac{1}{2}\ \sin(k-n)x\ , \quad \text{and} \\ \sin\ kx\ \sin nx &= -\frac{1}{2}\ \cos(k+n)x + \frac{1}{2}\cos(k-n)x, \end{aligned}$$

every term in the RHS of (7.4) integrates to zero apart from the one term which corresponds to $\frac{1}{2}\cos(k-n)x \equiv \frac{1}{2}$ if $k = n$. Hence $\int_{-\pi}^{\pi} u(x)\sin nx\ dx = b_k \cdot \pi$, giving the value of b_k.

It is often inconvenient to have three types of terms in (7.1), and with that, the three different formulas (7.2) for the coefficients. Also, the function $u(x)$ may be complex valued. For these reasons, one often replaces (7.1) with a complex expansion

$$u(x) = \sum_{k=-\infty}^{\infty} \hat{u}_k e^{ikx} \tag{7.5}$$

and (7.2) by the single formula

$$\hat{u}_k = \frac{1}{2\pi}\int_{-\pi}^{\pi} u(x)e^{-ikx}dx \quad , \quad k = -\infty, \ldots, -1, 0, 1, \ldots, +\infty. \tag{7.6}$$

Equations (7.5) and (7.6) correspond to the second case 'CONTINUOUS, $(-\pi, \pi)$' in Figure 7.1.1. In the common situation that $u(x)$ is real, $\hat{u}_0$ is real and it follows directly from (7.6) that $\hat{u}_{-k} = \hat{u}_k^*, \ k = 1, 2, \ldots, \infty$ (with '$*$' denoting complex conjugation).

If $u(x)$ has a jump discontinuity, the Fourier series will at that location converge to the average value from the two sides. Near the jump, there will be a *Gibbs' phenomenon*. We wish to distinguish between two different version of Gibbs' phenomenon:

- Use the exact Fourier coefficients but truncate the Fourier expansion to a finite number of terms.
- Use the Fourier polynomial that interpolates the discontinuous function at N points.

These two cases are discussed next.

(1) **Truncated Fourier expansion.** The partial sums $\sum_{k=-N}^{N} \hat{u}_k e^{ikx}$ 'overshoots' the jump at each side by about 9%. As $N \to \infty$, the region in which this occurs gets narrower, but the height persists. In this case, the error in the partial sum (in the maximum norm) will be $O(1)$ near the jump and $O(1/N)$ away from it.

(2) **Fourier interpolation.** Another version of Gibbs' phenomenon occurs if one performs equispaced interpolation with trigonometric functions (the interpolation formula is given by equation (7.12) in Section 7.4). In this case, the overshoot approaches 14% of the height of the jump as the density of the interpolation points tend to infinity. Figure 7.2.1 illustrates Gibbs' phenomenon for the Fourier interpolation formula. Note how the interpolation function passes through the interpolation points in the top part of the Figure. For comparison the overshoot in the case of a truncated Fourier series is also indicated.

If $u(x)$ is smoother, the convergence rates become faster, and the Gibbs overshoot disappears:

Order of max − norm errors in partial Fourier sums caused by irregularities of a function

	Max-norm of errors (order)	
Function	Near irregularity	Away from irregularity
f discont.	1	$1/N$
f' discont.	$1/N$	$1/N^2$
f'' discont.	$1/N^2$	$1/N^3$
$\vdots$	$\vdots$	$\vdots$
f analytic (periodic)	e^{-cN}, $c > 0$	

The condition that a Fourier series converges if $u(x)$ is piecewise differentiable can be weakened somewhat, but attempting to do this leads to surprisingly tricky mathematics.

> In spite of the fact that a truncated Fourier series $\sum_{k=-N}^{N} \hat{u}_k e^{ikx}$ always gives the best approximation to $u(x)$ of all possible sums $\sum_{k=-N}^{N} d_k e^{ikx}$ in the sense of least squares—($\int_{-\infty}^{\infty}(u(x)-\sum_{k=-N}^{N} d_k e^{ikx})^2 dx$ is minimized—the Fourier series of a continuous function $u(x)$ can still diverge to infinity at one (or more) points as $N \to \infty$. However, such mathematical subtleties play no role in the practical application of Fourier series.

It is always permissible to integrate term-by-term both sides of a Fourier expansion. Taking derivatives also works, but on condition that the resulting series converges uniformly and absolutely.

There are numerous uses of Fourier expansions in analysis/modeling/engineering and it is impossible to provide a complete list. In this book, they enter in the contexts of CT image reconstruction and the analytic solution of some wave- and heat equations. Other applications include,

- Signal and image processing (the widely used JPEG algorithm for image compression is Fourier based),

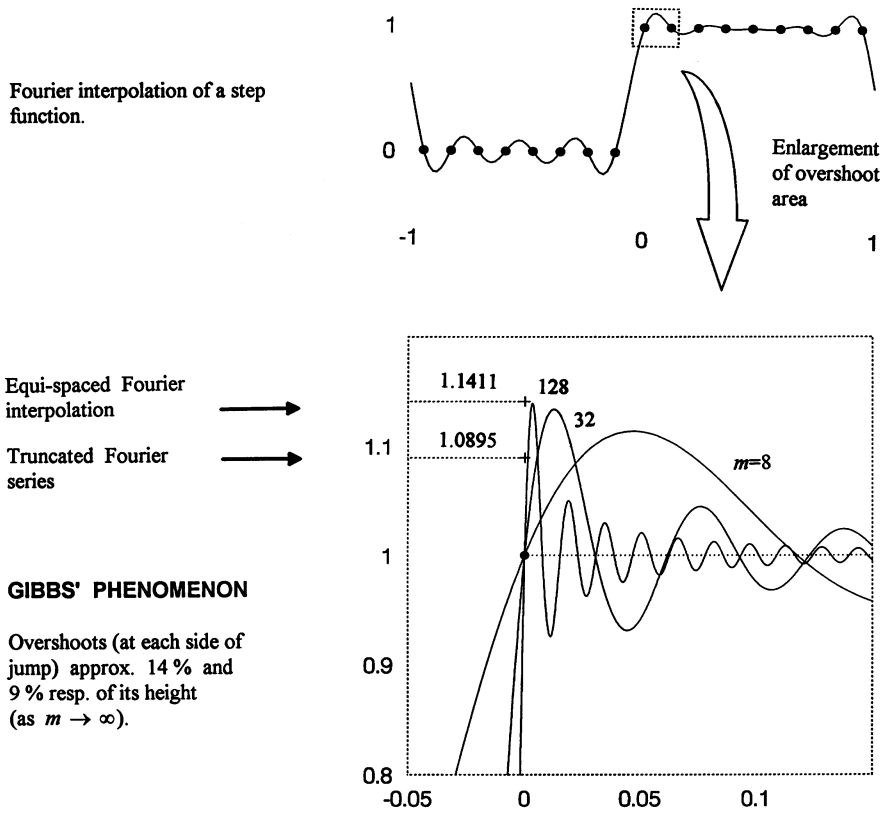


FIGURE 7.2.1. Gibbs' phenomenon for trigonometric interpolation and for truncated Fourier series.

- Numerical algorithms (e.g. the fast multiplication of two very large numbers),
- The Numerical solution of partial differential equations (e.g. the Fourier spectral or pseudospectral schemes), etc.

7.3. Fourier transform.

To keep the notation simple, we introduced in the last section Fourier series assuming $[-\pi, \pi]$ periodicity. In case of a more general period of $[-L\pi, L\pi]$ (L need not be an integer), the equations (7.3) and (7.6) becomes

$$u(x) = \sum_{k=-\infty}^{\infty} \hat{u}_k e^{i\frac{k}{L}x} \qquad \text{where} \qquad \hat{u}_k = \tfrac{1}{2\pi L}\int_{-L\pi}^{L\pi} u(x)e^{-i\frac{k}{L}x}dx \ .$$

Writing $\hat{u}_k = \frac{1}{L}\hat{u}(\frac{k}{L})$ gives

$$u(x) = \tfrac{1}{L}\sum_{k=-\infty}^{\infty} \hat{u}\left(\tfrac{k}{L}\right) e^{i\frac{k}{L}x} \qquad \text{and} \qquad \hat{u}\left(\tfrac{k}{L}\right) = \tfrac{1}{2\pi}\int_{-L\pi}^{L\pi} u(x)e^{-i\frac{k}{L}x}dx \ .$$

We now call $\frac{k}{L} = \omega$ and let $L \to \infty$. In the limit, the sum above becomes an integral, and we get the Fourier transform, as listed on the top line of Figure 7.1.1:

$$u(x) = \int_{-\infty}^{\infty} \hat{u}(\omega)e^{i\omega x}d\omega \qquad \text{and} \qquad \hat{u}(\omega) = \tfrac{1}{2\pi}\int_{-\infty}^{\infty} u(x)e^{-i\omega x}dx \ . \tag{7.1}$$

Explained in words:

- When making the period wider, we need to use an ever denser set of frequencies (no longer just e^{ikx} but, for $[-L\pi, L\pi]$-periodicity, $e^{i\frac{k}{L}x}$ where k runs through all the integers),
- As $L \to \infty$, a continuum of frequencies is needed. Instead of summing the frequency components to represent our function, we need to integrate over them.

Convergence of a Fourier transform is assured if $\int_{-\infty}^{\infty} |u(x)|\, dx < \infty$. As in the case of a Fourier series, many important uses of the Fourier transform follow from differentiating or integrating (7.1). Table 1 summarizes a number of key properties of the Fourier transform. Table 2 gives some examples of the transforms of some simple functions.

Unfortunately there is considerable confusion between different texts about how to best define Fourier transforms. Our convention was stated in equation (7.1)—the function u(x) is a straightforward superposition of the basic modes $e^{i\omega x}$. The transform using this choice is denoted $\hat{u}_1(\omega)$ below, and compared against two other

Function	Transform	Comment
$\frac{du}{dx}$	$i\omega\hat{u}(\omega)$	Turns space derivatives into algebraic factors; key to solving many ODEs and turning some PDEs into ODEs. Derivation: $\widehat{\left(\frac{du}{dx}\right)} = \frac{1}{2\pi}\int_{-\infty}^{\infty}\frac{du}{dx}e^{-i\omega x}dx = \left[\frac{u\,e^{-i\omega x}}{2\pi}\right]_{-\infty}^{\infty} + \frac{i\omega}{2\pi}\int_{-\infty}^{\infty} u\,e^{-i\omega x}dx = i\omega\hat{u}$
$\frac{d^2u}{dx^2}$	$-\omega^2\hat{u}(\omega)$	Immediate from case above
$x\,u(x)$	$i\,\frac{d\hat{u}(\omega)}{d\omega}$	Can for example be used to obtain integral representations of variable coefficient ODEs - a key to generating many asymptotic estimates. Ex: The Fourier transform to Airy's equation $y'' - xy = 0$ satisfies $-\omega^2\hat{y} - i\hat{y}' = 0 \Rightarrow \hat{y} = A\,e^{i\,\omega^3/3};\ y(x) = = \int_{-\infty}^{\infty} A\,e^{i\,\omega^3/3}e^{i\,\omega\,x}d\omega = A\int_{-\infty}^{\infty}\cos\left(\frac{\omega^3}{3} + \omega x\right)d\omega$
$u(x-\alpha)$	$e^{-i\omega\alpha}\hat{u}(\omega)$	Translation of a function introduces an exponential factor in the transform
$\frac{1}{2\pi}\int_{-\infty}^{\infty} f(\xi)g(x-\xi)d\xi$	$\hat{f}(\omega)\cdot\hat{g}(\omega)$	Convolution theorem: Large number of applications - some described in connection with DFT-FFT. Derivation: Given $f(x),\ g(x)$, then which function $h(x)$ has the transform $\hat{h}(\omega) = \hat{f}(\omega)\cdot\hat{g}(\omega)$? $h(x) = \int_{-\infty}^{\infty}\hat{f}(\omega)\cdot\hat{g}(\omega)\,e^{i\omega x}d\omega = = \int_{-\infty}^{\infty}\left[\frac{1}{2\pi}\int_{-\infty}^{\infty} f(\xi)\,e^{-i\omega\xi}d\xi\right]\hat{g}(\omega)e^{i\omega x}d\omega = = \frac{1}{2\pi}\int_{-\infty}^{\infty} f(\xi)\left[\int_{-\infty}^{\infty}\hat{g}(\omega)\,e^{i\omega(x-\xi)}d\omega\right]d\xi = = \frac{1}{2\pi}\int_{-\infty}^{\infty} f(\xi)g(x-\xi)d\xi$
$\frac{1}{2\pi}\int_{-\infty}^{\infty}\lvert u(\xi)\rvert^2\,d\xi = \int_{-\infty}^{\infty 2} d\omega\,\lvert\hat{u}(\omega)\rvert^2$		Parseval's relation: Setting $x = 0$ in the derivation of the convolution theorem above gives: $h(0) = \int_{-\infty}^{\infty}\hat{f}(\omega)\cdot\hat{g}(\omega)d\omega = \frac{1}{2\pi}\int_{-\infty}^{\infty} f(\xi)g(-\xi)d\xi$; Result follows from setting $f(x) = u(x),\ g(x) = \overline{u(-x)}$

TABLE 1. Some properties of the Fourier transform.

versions:

Convention here	$u(x) = \int_{-\infty}^{\infty}\hat{u}_1(\omega)e^{i\omega x}d\omega$	$\hat{u}_1(\omega) = \frac{1}{2\pi}\int_{-\infty}^{\infty} u(x)e^{-i\omega x}dx$
Change of sign in exponents	$u(x) = \int_{-\infty}^{\infty}\hat{u}_2(\omega)e^{-i\omega x}d\omega$	$\hat{u}_2(\omega) = \frac{1}{2\pi}\int_{-\infty}^{\infty} u(x)e^{i\omega x}dx$
Scaling of frequencies by factor 2π	$u(x) = \int_{-\infty}^{\infty}\hat{u}_3(\omega)e^{2\pi i\omega x}d\omega$	$\hat{u}_3(\omega) = \int_{-\infty}^{\infty} u(x)e^{-2\pi i\omega x}dx$

Some examples of Fourier transforms

Function	Transform	Comment (in all cases Re$\alpha > 0$)
$\delta(x - x_0)$	$\frac{1}{2\pi} e^{-i\omega x_0}$	Transform of a delta function is a single trig. mode
$e^{i\omega_0 x}$	$\delta(\omega - \omega_0)$	Transform of a single trig. mode is a delta function
$e^{-\alpha x^2}$	$\frac{1}{\sqrt{4\pi\alpha}} e^{-\omega^2/(4\alpha)}$	Some functions have transforms which are simply scaled copies of themselves. The transform of a real and even function is again real and even.
$\sqrt{\frac{\pi}{\alpha}} e^{-x^2/(4\alpha)}$	$e^{-\alpha\omega^2}$	Different scaling of case above
$\frac{1}{\cosh \alpha x}$	$\frac{1}{2\alpha \cosh\left(\frac{\pi\omega}{2\alpha}\right)}$	
$\frac{1}{\cosh^2 \alpha x}$	$\frac{\omega}{2\alpha^2 \sinh\left(\frac{\pi\omega}{2\alpha}\right)}$	
$\frac{1}{\sqrt{\lvert x\rvert}}$	$\frac{1}{\sqrt{2\pi\lvert\omega\rvert}}$	Note: singular at the origin
$x\, e^{-\alpha x^2}$	$-\frac{i\omega}{4\alpha\sqrt{\pi\alpha}} e^{-\omega^2/(4\alpha)}$	Derivative of Gaussian - transform of a real odd function is again odd, but purely imaginary
$\begin{cases} 0 & \lvert x\rvert > \alpha \\ 1 & \lvert x\rvert \le \alpha \end{cases}$	$\frac{1}{\pi} \frac{\sin \alpha\omega}{\omega}$	Transform of a step function decays for large ω like $O(1/\omega)$
$e^{-\alpha\lvert x\rvert}$	$\frac{\alpha}{\pi(\omega^2+\alpha^2)}$	If the derivative is discontinuous, the decay rate becomes $O(1/\omega^2)$ (like for a Fourier series)

TABLE 2. Some examples of Fourier transforms

		TO		
		$\hat{u}_1$	$\hat{u}_2$	$\hat{u}_3$
FROM	$\hat{u}_1$	-	$\omega \to -\omega$	$\omega \to 2\pi\omega$ Multiply by 2π
	$\hat{u}_2$	$\omega \to -\omega$	-	$\omega \to -2\pi\omega$ Multiply by 2π
	$\hat{u}_3$	$\omega \to \omega/(2\pi)$ Divide by 2π	$\omega \to \omega/(2\pi)$ Divide by 2π	-

TABLE 3. Conversions between the different definitions of the Fourier transform.

If we know a Fourier transform in one of these systems, we can quickly convert to any of the other as illustrated in Table 3.

7.4. Discrete Fourier transform (DFT).

For the discrete case, we consider first a function $u(x)$ which is 2π-periodic, and defined only at the N *equispaced* node points $x_j = j\frac{2\pi}{N},\ j = 0, 1, \ldots, N-1$. The modes are then limited to

$$e^{ikx} \quad , \quad k = 0, 1, \ldots, N-1.$$

This can be contrasted to $\{e^{i\omega x},\ \omega \text{ real}\}$ for the Fourier transform, and $\{e^{ikx},\ k$ any integer$\}$ for the Fourier series of a 2π-periodic function.

Both the input data and the output results for the DFT consists of N complex numbers. It is natural to index both of these sets by the integers $0, 1, \ldots, N-1$. We then write the discrete transform pair as follows:

$$u_j = \textstyle\sum_{k=0}^{N-1} \hat{u}_k e^{2\pi ikj/N} \quad , \quad j = 0, 1, \ldots, N-1 \tag{7.1}$$

and

$$\hat{u}_k = \tfrac{1}{N} \textstyle\sum_{j=0}^{N-1} u_j e^{-2\pi ikj/N} \quad , \quad k = 0, 1, \ldots, N-1. \tag{7.2}$$

These transforms are often best expressed in matrix form; among other things it will tell us that these two equations are equivalent—one follows from the other. The matrix form of (7.1) is given by,

$$\begin{bmatrix} 1 & 1 & 1 & 1 & \ldots & 1 \\ 1 & \omega & \omega^2 & \omega^3 & \ldots & \omega^{N-1} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \ldots & \omega^{2N-2} \\ \vdots & \vdots & & & & \vdots \\ \vdots & \vdots & & & & \vdots \\ 1 & \omega^{N-1} & \ldots & \ldots & \ldots & \omega^{(N-1)^2} \end{bmatrix} \begin{bmatrix} \hat{u}_0 \\ \hat{u}_1 \\ \hat{u}_2 \\ \vdots \\ \vdots \\ \hat{u}_{N-1} \end{bmatrix} = \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \\ \vdots \\ u_{N-1} \end{bmatrix} \tag{7.3}$$

where ω is the N-th root of unity, i.e. $\omega = e^{2\pi i/N}$. Similarly for the inverse transform (7.2):

$$\frac{1}{N}\begin{bmatrix} 1 & 1 & 1 & 1 & \dots & 1 \\ 1 & \omega^{-1} & \omega^{-2} & \omega^{-3} & \dots & \omega^{-N+1} \\ 1 & \omega^{-2} & \omega^{-4} & \omega^{-6} & \dots & \omega^{-2N+2} \\ \vdots & \vdots & & & & \vdots \\ \vdots & \vdots & & & & \vdots \\ 1 & \omega^{-N+1} & \dots & \dots & \dots & \omega^{-(N-1)^2} \end{bmatrix} \begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \\ \vdots \\ u_{N-1} \end{bmatrix} = \begin{bmatrix} \hat{u}_0 \\ \hat{u}_1 \\ \hat{u}_2 \\ \vdots \\ \vdots \\ \hat{u}_{N-1} \end{bmatrix} \tag{7.4}$$

We now verify that the product of the two matrices above is N times the identity matrix. It should be clear that all the diagonal elements of the product equal N, it remains to show that all the off-diagonal elements vanish. A little thought (and a small specific example is always helpful) should convince the reader that the off-diagonal elements are all of the form

$$\sum_{j=0}^{N-1} \omega^{kj}$$

where k is an integer with $1 \le k \le N-1$. This sums to $(1-\omega^{kN})/(1-\omega^k)$. Since ω is the Nth root of unity, the numerator is zero and because of the restrictions on k the denominator is not zero. This confirms that (7.1), (7.2) indeed constitutes a transform and its inverse.

In Section *not written*, we find that the fast Fourier transform (FFT) amounts to exploiting the matrix in (7.3) (or similarly in (7.4)) can be factored into a product of very sparse matrices. The matrix×vector product in (7.3) can then instead be performed by multiplying the vector in turn with these sparse matrices.

7.4.1. Discrete convolution theorem. One of the most important applications of the FFT algorithm is that it allows periodic discrete convolutions to be calculated rapidly. If three vectors

$$[x_0, x_1, \dots, x_{N-1}]\,,\ [y_0, y_1, \dots, y_{N-1}]\,,\ [z_0, z_1, \dots, z_{N-1}] \tag{7.5}$$

satisfy

$$\begin{bmatrix} z_0 & z_{N-1} & z_{N-2} & \ddots & z_1 \\ z_1 & z_0 & z_{N-1} & \ddots & z_2 \\ \ddots & \ddots & \ddots & \ddots & \ddots \\ \ddots & \ddots & \ddots & \ddots & \ddots \\ z_{N-1} & z_{N-2} & \ddots & z_1 & z_0 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ \vdots \\ x_{N-1} \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ \vdots \\ \vdots \\ y_{N-1} \end{bmatrix}, \tag{7.6}$$

then their DFT coefficients satisfy

$$\begin{bmatrix} \hat{z}_0 & & & & \\ & \hat{z}_1 & & & \\ & & \ddots & & \\ & & & \ddots & \\ & & & & \hat{z}_{N-1} \end{bmatrix} \begin{bmatrix} \hat{x}_0 \\ \hat{x}_1 \\ \vdots \\ \vdots \\ \hat{x}_{N-1} \end{bmatrix} = \frac{1}{N} \begin{bmatrix} \hat{y}_0 \\ \hat{y}_1 \\ \vdots \\ \vdots \\ \hat{y}_{N-1} \end{bmatrix}. \tag{7.7}$$

Since this equation amounts to a simple multiplication of the DFT coefficients,

$$\hat{z}_n \hat{x}_n = \hat{y}_n,$$

this result, together with the FFT algorithm, offers a very fast way to calculate any one of the vectors in (7.5) if the other two are provided.

The following is an illustration of how the discrete convolution theorem can be derived in the case of $N = 4$ generalizes immediately to any size. We multiply (7.7) from the left by the matrix in (7.3) and in-between the matrix and vector in (7.7), we insert the matrices from (7.4) and (7.3):

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 \\ 1 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega^9 \end{bmatrix} \begin{bmatrix} \hat{z}_0 & & & \\ & \hat{z}_1 & & \\ & & \hat{z}_2 & \\ & & & \hat{z}_3 \end{bmatrix} \frac{1}{N} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega^{-1} & \omega^{-2} & \omega^{-3} \\ 1 & \omega^{-2} & \omega^{-4} & \omega^{-6} \\ 1 & \omega^{-3} & \omega^{-6} & \omega^{-9} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 \\ 1 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega^9 \end{bmatrix} \begin{bmatrix} \hat{x}_0 \\ \hat{x}_1 \\ \hat{x}_2 \\ \hat{x}_3 \end{bmatrix} =$$

$$= \frac{1}{N} \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & \omega & \omega^2 & \omega^3 \\ 1 & \omega^2 & \omega^4 & \omega^6 \\ 1 & \omega^3 & \omega^6 & \omega^9 \end{bmatrix} \begin{bmatrix} \hat{y}_0 \\ \hat{y}_1 \\ \hat{y}_2 \\ \hat{y}_3 \end{bmatrix}$$

We now cancel the $1/N$, multiply together the three leftmost matrices, and apply (7.1) to the two matrix×vector products:

$$\begin{bmatrix} \hat{z}_0 + \hat{z}_1 + \hat{z}_2 + \hat{z}_3 & \hat{z}_0 + \hat{z}_1\omega^3 + \hat{z}_2\omega^6 + \hat{z}_3\omega^9 & \hat{z}_0 + \hat{z}_1\omega^2 + \hat{z}_2\omega^4 + \hat{z}_3\omega^6 & \hat{z}_0 + \hat{z}_1\omega^1 + \hat{z}_2\omega^2 + \hat{z}_3\omega^3 \\ \hat{z}_0 + \hat{z}_1\omega^1 + \hat{z}_2\omega^2 + \hat{z}_3\omega^3 & \hat{z}_0 + \hat{z}_1 + \hat{z}_2 + \hat{z}_3 & \hat{z}_0 + \hat{z}_1\omega^3 + \hat{z}_2\omega^6 + \hat{z}_3\omega^9 & \hat{z}_0 + \hat{z}_1\omega^2 + \hat{z}_2\omega^4 + \hat{z}_3\omega^6 \\ \hat{z}_0 + \hat{z}_1\omega^2 + \hat{z}_2\omega^4 + \hat{z}_3\omega^6 & \hat{z}_0 + \hat{z}_1\omega^1 + \hat{z}_2\omega^2 + \hat{z}_3\omega^3 & \hat{z}_0 + \hat{z}_1 + \hat{z}_2 + \hat{z}_3 & \hat{z}_0 + \hat{z}_1\omega^3 + \hat{z}_2\omega^6 + \hat{z}_3\omega^9 \\ \hat{z}_0 + \hat{z}_1\omega^3 + \hat{z}_2\omega^6 + \hat{z}_3\omega^9 & \hat{z}_0 + \hat{z}_1\omega^2 + \hat{z}_2\omega^4 + \hat{z}_3\omega^6 & \hat{z}_0 + \hat{z}_1\omega^1 + \hat{z}_2\omega^2 + \hat{z}_3\omega^3 & \hat{z}_0 + \hat{z}_1 + \hat{z}_2 + \hat{z}_3 \end{bmatrix} \times$$

$$\times \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix}$$

From (7.3) follows that the elements in the matrix above agree with the ones in the matrix (7.6).

It is sometimes useful to write the convolution as

$$y_n = (z * x)_n := \sum_{s=0}^{N-1} z_{n-s} x_s$$

The convolution now follows by a direct application of the DFT pair, (7.1) and 7.2.

7.4.2. Aliasing. It follows directly from (7.2) that

$$\hat{u}_{k+pN} = \frac{1}{N} \sum_{j=0}^{N-1} u_j e^{-2\pi i(k+pN)j/N} = \hat{u}_k$$

for any integer p. Thus, modes for which k-values differ by N (or by any multiple of N) become indistinguishable at the node points, as seen in Figure 7.11. Consequently, consider a $[-\pi, \pi]$-periodic function $u(x)$, with the Fourier series

$$u(x) = \sum_{k=-\infty}^{\infty} \hat{u}_k e^{ikx}. \tag{7.8}$$

Sampling this function at the discrete points $x_j = j\frac{2\pi}{N}$, the DFT coefficients (here denoted by $\hat{U}_k$, to distinguish from the Fourier series coefficients above) can be calculated from (7.2). Substituting (7.8) into the expression for $\hat{U}_k$ we obtain the following,

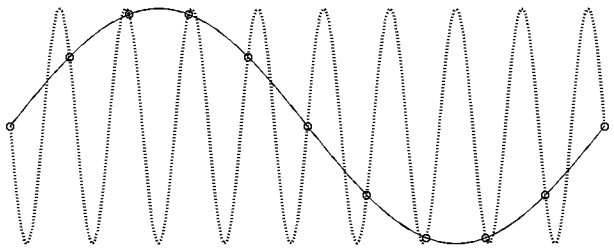


FIGURE 7.4.1. The functions $\sin x$ and $-\sin 9x$ are indistinguishable on the grid $x_j = \frac{2\pi j}{10},\ j = 0, 1, \ldots, 10$.

$$\hat{U}_k = \frac{1}{N}\sum_{j=0}^{N-1} u_j e^{-2\pi ikj/N} \tag{7.9}$$

$$= \frac{1}{N}\sum_{s=-\infty}^{\infty}\sum_{j=0}^{N-1} \hat{u}_s e^{2\pi ij(s-k)/N} \tag{7.10}$$

$$= \sum_{n=-\infty}^{\infty} \hat{u}_{k+nN}. \tag{7.11}$$

The effect of sampling $u(x)$ at discrete points therefore leads to an aliasing error—all the modes that cannot be resolved by the grid are folded back (aliased) onto those that are resolved by the grid.

7.4.3. Numbering of modes and trigonometric interpolation. Let us for now assume that N is odd (this is not important, but it slightly simplifies the algebra). Suppose we are given $u_0,\ u_1, \ldots, u_{N-1}$, all real numbers, and that we have

calculated the complex DFT coefficients $\hat{u}_0,\ \hat{u}_1, \ldots, \hat{u}_{N-1}$ using (7.2). Then one can form the Fourier interpolant,

$$U(x) := \sum_{k=0}^{N-1} \hat{u}_k e^{ikx}.$$

One would expect this to give interpolated values of $u(x)$ in-between the original data points. However, this leads to a nasty surprise: Even when all the original data is real, the interpolant becomes complex-valued between the node points ($j = 0, 1, \ldots, N-1$). To resolve this, we need to make use of aliasing:

Fourier modes	$\hat{u}_0$	$\hat{u}_1$	$\ldots$	$\hat{u}_{\frac{N-1}{2}}$	$\hat{u}_{\frac{N+1}{2}}$	$\ldots$	$\hat{u}_{N-2}$	$\hat{u}_{N-1}$
Corresponding to wave numbers	0	1	$\ldots$	$\frac{N-1}{2}$	$\frac{N+1}{2}$	$\ldots$	N-2	N-1
By aliasing, can instead be viewed as wave numbers	0	1	$\ldots$	$\frac{N-1}{2}$	$-\frac{N-1}{2}$	$\ldots$	-2	-1

This is of course exactly the aliasing result derived in the previous section,

$$\hat{u}_{-k} = \hat{u}_{-k+N}.$$

Thus the discrete transform (7.1) can be written as

$$u_j = \sum_{k=-\frac{1}{2}(N-1)}^{\frac{1}{2}(N-1)} \hat{u}_k e^{2\pi ijk/N}.$$

This suggests the following definition for the Fourier interpolant,

$$U(x) = \sum_{k=-\frac{1}{2}(N-1)}^{\frac{1}{2}(N-1)} \hat{u}_k e^{ikx}. \tag{7.12}$$

If the original function u_j is real then $\hat{u}_0$ is real and $\hat{u}_{-k} = \hat{u}_k^* t$ with the result that $U(x) = U(x)^*$, i.e. $U(x)$ is a real function as it should be.

If N is even, an additional slight complication arises. From the aliasing result (7.11) follows that

$$\hat{u}_{-\frac{1}{2}N} = \hat{u}_{\frac{1}{2}N}.$$

This is known as the *reflection* or *Nyquist frequency* and requires the following modification in the interpolation formula,

$$U(x) = \sum_{k=-\frac{1}{2}N+1}^{\frac{1}{2}N-1} \hat{u}_k e^{ikx} + \hat{u}_{\pm\frac{1}{2}N} \cos(\frac{1}{2}Nx),$$

where $\hat{u}_{\pm\frac{1}{2}N} := \hat{u}_{+\frac{1}{2}N} = \hat{u}_{-\frac{1}{2}N}$. In many cases, we can simply set $\hat{u}_{\pm\frac{1}{2}N}$ to zero (or just ignore its presence, depending on the application).

7.4.4. Character of transform for different cases of input data. If the N input numbers to a DFT are complex, so are the output ones. This is seen illustrated as the top case in Figure 7.4.2. If the input is a real vector (next case in Figure 7.4.2), the output will have the particular structure illustrated to the right. To avoid unnecessary computations, specialized codes for this case use as input and deliver as output only the entries inside the boxes shown with thick boundary lines. The last two cases—cosine and sine transforms—arise also often in applications. The real input data is here itself symmetric or anti-symmetric, and the transforms take on a similar structure. These examples illustrate again that it is necessary to interpret the modes in the order [0, 1, 2, ..., -2, -1], as just described (rather than [0, 1, 2, ..., N-2, N-1]).

In this Figure 7.4.2, N has been assumed to be even. This is the most common case in computations because the FFT algorithm is particularly simple and effective for sizes—values of N—that are powers of two. One therefore has to keep in mind that the mode $N/2$ then does not quite 'fit the pattern', as pointed out above.

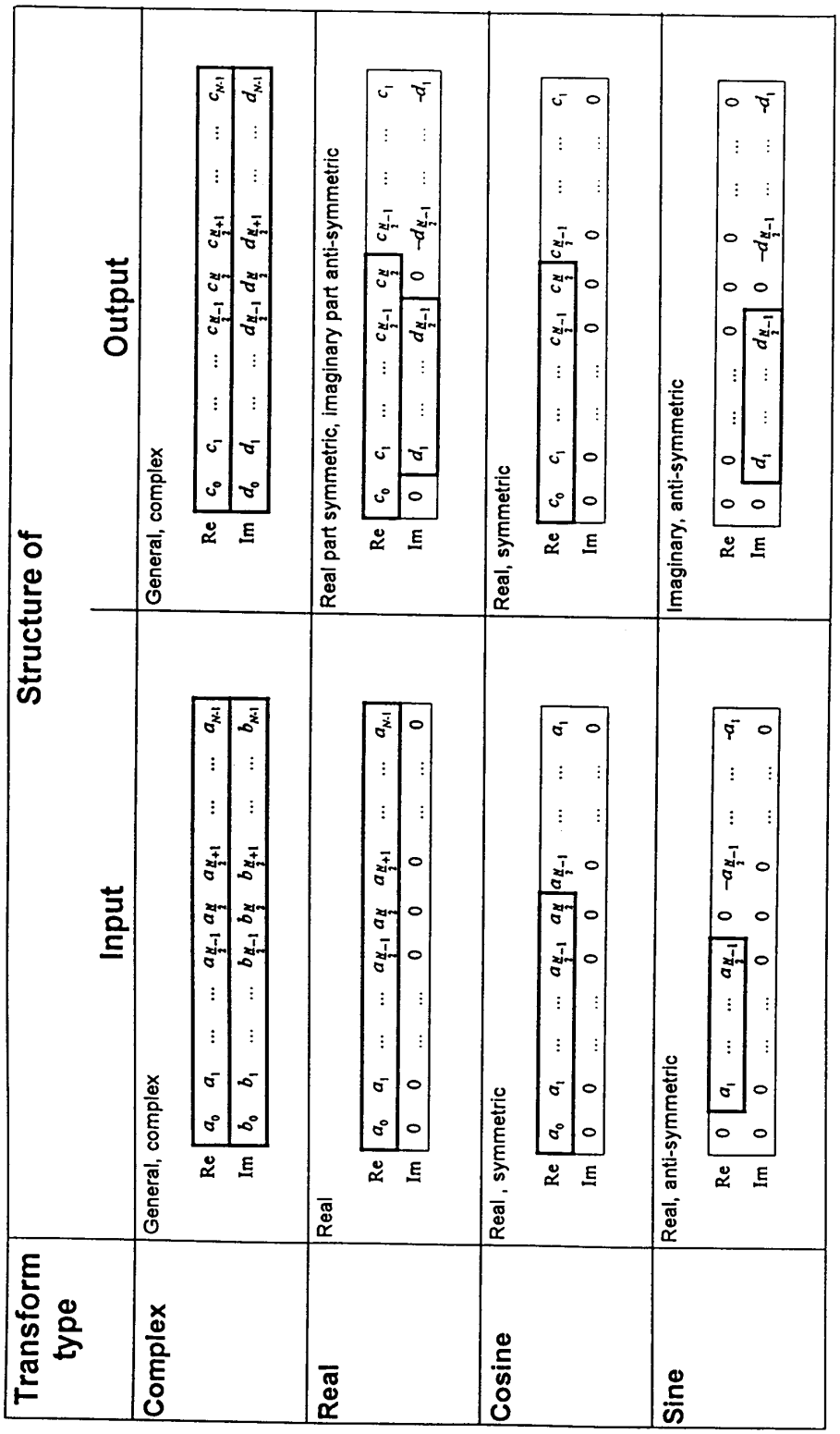


FIGURE 7.4.2. Comparison between a general complex DFT and some special cases of it. The heavy lines surround the data vectors which specialized routines for the cases typically use as input and output (other table entries are superfluous).

7.5. 2-D Fourier transform.

We can graphically display the paths (both ways) between $u(x)$ and its 1-D Fourier transform $\hat{u}(\omega)$ as

$$\boxed{u(x)} \quad \begin{array}{c} \nearrow \ \frac{1}{2\pi}\int_{-\infty}^{\infty} u(x)e^{-i\omega x}dx \ \searrow \\ \nwarrow \ \int_{-\infty}^{\infty} \hat{u}(\omega)e^{i\omega x}d\omega \ \swarrow \end{array} \quad \boxed{\hat{u}(\omega)}$$

In 2-D, the Fourier transform pair is defined as

$$u(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \hat{u}(\omega_x,\omega_y)e^{i(\omega_x x+\omega_y y)}d\omega_x\, d\omega_y \tag{7.1}$$

where

$$\hat{u}(\omega_x,\omega_y) = \frac{1}{(2\pi)^2}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} u(x,y)e^{-i(\omega_x x+\omega_y y)}dx\, dy \tag{7.2}$$

This can be graphically displayed as a 2-step process using only 1-D Fourier transforms:

$$\boxed{u(x,y)} \quad \begin{array}{c} \nearrow \ \frac{1}{2\pi}\int_{-\infty}^{\infty} u(x,y)e^{-i\omega_x x}dx \ \searrow \\ \nwarrow \ \int_{-\infty}^{\infty} \breve{u}(\omega_x,y)e^{i\omega_x x}d\omega_x \ \swarrow \end{array} \quad \boxed{\breve{u}(\omega_x,\omega_y)} \quad \begin{array}{c} \nearrow \ \frac{1}{2\pi}\int_{-\infty}^{\infty} \breve{u}(\omega_x,y)e^{-i\omega_y y}dy \ \searrow \\ \nwarrow \ \int_{-\infty}^{\infty} \hat{u}(\omega_x,\omega_y)e^{i\omega_y y}d\omega_y \ \swarrow \end{array} \quad \boxed{\hat{u}(\omega_x,\omega_y)}$$

The path left-to-right clearly illustrates (7.2) and the return path (7.1).We can note

- It is of no consequence that we carried out the integrations x first, then y; reversed order gives the same result,
- The fact that the equations (7.1), (7.2) form a transform pair (one inverse of the other) follows from this same property of the 1-D transforms,
- There is no need to have special 2-D transform codes - successive 1-D transforms can be used instead.

One of the most noteworthy (and, for the FT method for tomography, critically important) properties of the 2-D Fourier transform is the following:

Theorem: If a function is rotated around the origin in (x, y)-space, its Fourier transform becomes rotated the same angle in (ω_x, ω_y)-space (and undergoes otherwise no change).

Proof: If the original function $u(x, y)$ has the Fourier transform

$$\hat{u}(\omega_x, \omega_y) = \frac{1}{(2\pi)^2} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} u(x, y) e^{-i(\omega_x x + \omega_y y)} dx\ dy \ \ ,$$

then the rotated function $u(x\cos\theta + y\sin\theta, -x\sin\theta + y\cos\theta)$ has the transform

$$\tfrac{1}{(2\pi)^2} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} u(x\cos\theta + y\sin\theta, -x\sin\theta + y\cos\theta) e^{-i(\omega_x x + \omega_y y)} dx dy =$$

$$\text{call } \left\{ \begin{array}{l} x' = x\cos\theta + y\sin\theta \\ y' = -x\sin\theta + y\cos\theta \end{array} \right\}, \text{ then } \left\{ \begin{array}{l} x = x'\cos\theta - y'\sin\theta \\ y = x'\sin\theta + y'\cos\theta \end{array} \right\}$$

$$= \tfrac{1}{(2\pi)^2} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} u(x', y') e^{-i[\omega_x(x'\cos\theta - y'\sin\theta) + \omega_y(x'\sin\theta + y'\cos\theta)]} dx dy =$$

reorder terms in exponent; change integration variables from x, y to x', y' (no extra factor since Jacobian $= 1$)

$$= \tfrac{1}{(2\pi)^2} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} u(x', y') e^{-i[(\omega_x\cos\theta - \omega_y\sin\theta)x' + (-\omega_x\sin\theta + \omega_y\cos\theta)y']} dx' dy' =$$

irrelevant what the dummy integration variable is called, so

$$= \hat{u}(\omega_x\cos\theta + \omega_y\sin\theta, -\omega_x\sin\theta + \omega_y\cos\theta) \ .$$

Hence, the Fourier transform has become rotated the same angle around the origin in (ω_x, ω_y)-space as the function was in (x, y)-space.

The result generalizes immediately to 3-D. For an easy proof in this case, it suffices to note that any rotation in 3-space can be carried out as successive rotations around the three axes. For each of these cases, the proof above applies.

CHAPTER 8

DERIVATION AND ANALYSIS OF WAVE EQUATIONS

8.1. Introduction.

Wave phenomena are ubiquitous in nature. Examples include water waves, sound waves, electromagnetic waves (radio waves, light, X-rays, gamma rays etc.), the waves that in quantum mechanics are found to be an alternative (and often better) description of particles, etc. Some features are common for most waves, e.g. that they in cases of small amplitude can usually be well approximated by a simple trigonometric wave function (Section 8.2) Other features differ. In some cases, all waves travel with the same speed (e.g. sound waves, or light in vacuum) whereas in other cases, the speed depends strongly on the wave length (e.g. water waves, or quantum mechanical particle waves). In most cases, one can start from basic physical principles and from these derive partial differential equations (PDEs) that govern the waves. In Section 8.3 we will do this for transverse waves on a tight string, and for Maxwell's equations describing electromagnetic waves in 3-D. In both of these cases, we obtain linear PDEs that are quite easy to solve numerically. In other cases, such as water waves, discussed in Section 8.4, the full governing equations are too complex to give here, and we need to restrict ourselves to a number of general observations. In still other cases, such as the Schrödinger equation for quantum wave functions, the set of PDEs are well known and extremely accurate (often said to describe *all* of chemistry!) but these equations are prohibitively difficult to solve in all but the simplest special cases. We note in Section 8.5 that some important nonlinear wave equations can be formulated as systems of first order PDEs. Not only are these systems usually very well suited for numerical solution, they also allow a quite simple analysis regarding various features, such as types of waves they support, and their speeds (also when spatial derivatives have been replaced by finite differences). In the case

of the acoustic wave equation, discussed further in Section 8.6, we note some closed-form analytic solutions. We arrive in Section 8.7 at Hamilton's equations. These are fundamental in many applications, such as mechanical and dynamical systems, and the study of chaotic motions. In the context of this book, their key application is to provide the governing equations for the freak wave phenomenon that is discussed in Chapter 5.

8.2. Wave Function.

A progressive wave may at some instant in time look like what is shown in Figure 8.2.1

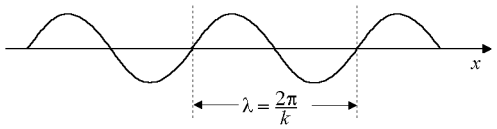


FIGURE 8.2.1. Snapshot of a progressive sinusoidal wave.

The variable that is displayed here need however not correspond to sideways deflections. For sound waves, it can represent pressure, for light it can represent the strength of electric or magnetic fields, etc. At least when the amplitude is small, such a progressive wave can be well approximated by a single trigonometric mode

$$\phi(x,t) = \phi_0 \, \cos(k\, x - \omega\, t) \; . \tag{8.1}$$

In terms of the *wave number* k and *angular time frequency* ω, we find

$$\begin{array}{rrcl} wave\, length & \lambda & = & 2\pi/k \\ frequency & \nu & = & \omega/(2\pi) \end{array}$$

If the time t is increased by one and the spatial position x by ω/k, the argument of the cosine in (8.1) is unchanged. Hence, the wave in (8.1) travels with

$$\text{phase speed} \qquad c_p \;=\; \omega/k \quad . \tag{8.2}$$

We will later come across another speed, *group speed*, c_g. If only 'wave speed' is mentioned, or no subscript for c is given, the phase speed c_p is assumed.

For almost all waves, ω is not a constant quantity, but a function of k. In some cases, this relation takes the form $\omega = c \cdot k$, e.g. for sound waves and for light in vacuum. In such cases the wave speed (according to (8.2)), becomes independent of the wave number. In other cases, the *dispersion relation* $\omega = \omega(k)$ takes different forms. For waves on deep water, the leading order approximation (when the wave amplitude is small) can be shown [??] to be

$$\omega = \sqrt{gk} \tag{8.3}$$

where g denotes the acceleration of gravity.

It is often very convenient to write the wave function as

$$\phi(x,t) = \phi_0 \text{ Re } \ e^{i\,(k\,x - \omega\,t)},$$

which is mathematically equivalent to (8.1). Many trigonometric manipulations become very much easier if one uses the complex exponential function rather than manipulate sines and cosines directly. It is even more convenient not to write "*Re*" all the time, but instead let that be implicitly understood. Hence, we will in the following usually write the 1-D wave function simply as

$$\phi(x,t) = \phi_0 \ e^{i\,(k\,x - \omega\,t)}. \tag{8.4}$$

Not all wave forms are sinusoidal. However, by Fourier analysis (cf. Chapter 7), any other shape can be viewed as a superposition of sinusoidal waves of different wave numbers k. Together with knowledge of the dispersion relation $\omega = \omega(k)$, we can then (for a linear wave equation) analyze how an initial wave form evolves in time.

The 2-D counterpart to (8.4) is

$$\phi(\underline{x},t) = \phi_0 \ e^{i\,(\underline{k}\cdot\underline{x} - \omega\,t)} \tag{8.5}$$

where $\mathbf{x} = (x_1, x_2)$ and $\mathbf{k} = \ (k_1, k_2)$ are two-component vectors.

The wave $\phi(\underline{x},t)$ given by (8.5) clearly reduces to (8.4) in case we introduce a (scalar) x-direction parallel to the $\mathbf{k}$-vector. We can also note that $\phi(\underline{x},t)$ is unchanged if $\mathbf{x}$ moves along any direction orthogonal to $\mathbf{k}$. From the fist observation follows that the wave length $\lambda = 2\pi/\left|\mathbf{k}\right|$ and the phase speed $c_p = \omega/\left|\mathbf{k}\right|$; from

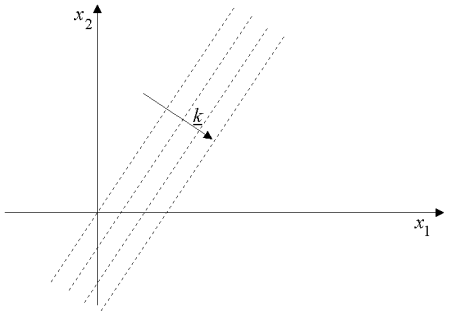


FIGURE 8.2.2. 2-D progressing wave. Wave crests are marked with dotted lines; waves progress in the direction of the **k**-vector.

the second observation follows that wave fronts are orthogonal to their direction of travel.

8.3. Examples of Derivations of Wave Equations.

The two cases we will consider are waves traveling along a string under tension, and Maxwell's equations for electromagnetic waves in 3-D. The methods of derivation that we will employ are rather different, and they illustrate two of the main approaches for obtaining governing equations in many other situations as well. In Section 8.7, we will come across a third approach via Hamilton's equations.

8.3.1. Transverse waves in a string under tension. Two kinds of waves will travel along a string under tension - transverse and longitudinal. Their speeds are typically very different. Transverse (sideways) oscillations are usually fairly slow (and visible), whereas longitudinal (lengthwise) waves causing no visible deflections while traveling with the speed of sound in the material (maybe of the order of 1 km/s). In a loosely stretched 'slinky', both wave types can be seen traveling at about 10 m/s. The transverse waves in a string is the simplest case to obtain an equation for, and we will do that next.

A string, with density ρ per unit length, is stretched in the x-direction with a tension force T (cf. Figure 8.3.1).

At any time, the vertical forces on the small string segment must balance. They are

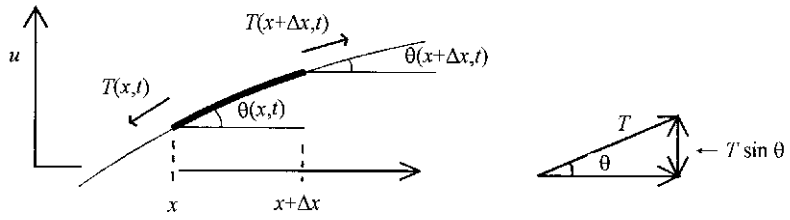


FIGURE 8.3.1. Illustration of an infinitesimal section of a transversely vibrating string.

$$\underbrace{\rho \cdot \Delta x}_{\text{mass}} \times \underbrace{\frac{\partial^2 u}{\partial\, t^2}}_{\text{acceleration}} = \underbrace{T(x+\Delta x,t)\ \sin\theta(x+\Delta x,t) - T(x,t)\ \sin\theta(x,t)}_{\text{difference between vertical tension forces at the two ends}}$$

Assuming the deflection angles $\theta(x,t)$ are small, $\sin\theta \approx \tan\theta = \frac{\partial u}{\partial x}$. Dividing both sides by Δx and letting $\Delta x \to 0$ then gives

$$\rho(x)\ \frac{\partial^2 u}{\partial t^2} = \frac{\partial}{\partial x}\left(T\ \frac{\partial u}{\partial x}\right).$$

Still assuming that the deflection is small, the tension T is approximately a constant, and can be factored out. Introducing $c^2 = T/\rho$, we arrive at the 1-D wave equation in its standard form

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2} \tag{8.1}$$

We will soon see that this equation supports waves traveling with the velocity c to either left or right.

8.3.2. Maxwell's equations in 3-D. ******.. To be filled in ....Will start by basic laws of electromagnetics, and then use vector calculus ****** to obtain:

$$\left\{\begin{array}{ll} \frac{\partial E_x}{\partial t} & = \frac{1}{\varepsilon}\left(\frac{\partial H_z}{\partial y} - \frac{\partial H_y}{\partial z}\right) \\ \frac{\partial E_y}{\partial t} & = \frac{1}{\varepsilon}\left(\frac{\partial H_x}{\partial z} - \frac{\partial H_z}{\partial x}\right) \\ \frac{\partial E_z}{\partial t} & = \frac{1}{\varepsilon}\left(\frac{\partial H_y}{\partial x} - \frac{\partial H_x}{\partial y}\right) \end{array}\right. , \quad \left\{\begin{array}{ll} \frac{\partial H_x}{\partial t} & = -\frac{1}{\mu}\left(\frac{\partial E_z}{\partial y} - \frac{\partial E_y}{\partial z}\right) \\ \frac{\partial H_y}{\partial t} & = -\frac{1}{\mu}\left(\frac{\partial E_x}{\partial z} - \frac{\partial E_z}{\partial x}\right) \\ \frac{\partial H_z}{\partial t} & = -\frac{1}{\mu}\left(\frac{\partial E_y}{\partial x} - \frac{\partial E_x}{\partial y}\right) \end{array}\right. \tag{8.2}$$

Here

$E_x,\ E_y,\ E_z$	the components of the electric field
$H_x,\ H_y,\ H_z$	the components of the magnetic field
μ	permeability
ε	permettivity

For lossy media, we need to subtract $\sigma E_x,\ \sigma E_y,\ \sigma E_z,\ \rho H_x,\ \rho H_y,\ \rho H_z$, resp. from the six RHSs (with σ and ρ denoting conductivity and magnetic resistivity respectively).

8.4. Water Waves.

A great variety of different wave phenomena occur in and on bodies of water. Table 1 summarizes some that can be found in lakes and oceans.

Wave type	Cause	Period	Velocity
Sound	Sea life, ships	$10^{-1} - 10^{-5}$ s	1.52 km/s
Capillary ripples	Wind	$< 10^{-1}$s	0.2 - 0.5 m/s
Gravity waves	Wind	1 - 25 s	2 - 40 m/s
Sieches	Earthquakes, storms	minutes to hours	standing waves
Storm surges	Low pressure areas	1 - 10 h	$\sim$ 100 m/s
Tsunami	Earthquakes, slides	10 min - 2 h	$<$ 800 km/h
Internal waves	Stratification instabilities, tides	2 min - 10 h	$<$ 5 m/s
Tides	Moon and sun	12 - 24 h	$<$ 1700 km/h
Planetary waves	Earth rotation	$\sim$ 100 days	1 - 10 km/h

TABLE 1. Types of water waves in lakes and oceans

We will here describe only the most obvious case - gravity waves (so called because gravity is the restoring force which strives to keep the surface level). To get started with some analysis of steady translating waves, we make a number of simplifying assumptions:

- no surface tension or wind forces,
- infinitely deep water,
- very small wave amplitude a.

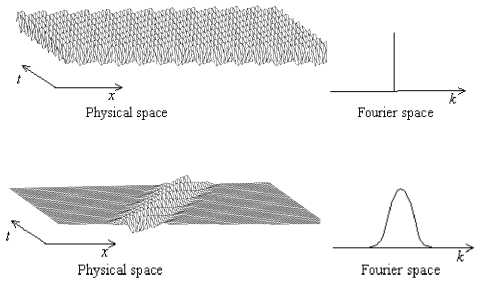


FIGURE 8.4.1. Physical and Fourier space representation of a uniform wave train and of a wave packet.

For waves on a string, we could write down a PDE which describes how any initial state evolves forward in time. In contrast to this, for deep water gravity waves, there is no single PDE which describes the evolution of a surface disturbance.

8.4.1. Dispersion relation for deep water. To make some progress, we need to make still one more simplifying assumption, namely that the surface elevation $\phi(x,t)$ for small amplitude becomes approximately sinusoidal:

$$\phi(x,t) = a\ \cos(kx - \omega t) \tag{8.1}$$

As noted earlier (8.3), the dispersion relation then becomes

$$\omega = \sqrt{gk}, \tag{8.2}$$

where the acceleration of gravity $g \approx 9.8\,\mathrm{m/s^2}$. Denoting the wave length by $\lambda = 2\pi/k$, the velocity of this wave (cf. (**??**)) becomes

$$c_p = \frac{\omega}{k} = \sqrt{\frac{g}{k}} = \sqrt{\frac{g\lambda}{2\pi}}, \tag{8.3}$$

and the time period $T = 2\pi/\omega$. The fact that c_p grows proportionally with $\sqrt{\lambda}$ leads to many notable features of water waves (such as the extremely high speed of tsunamis). Some of these are mentioned in Section ....

8.4.2. Group speed vs. phase speed. The upper part of Figure 8.4.1 illustrates a wave train, described by (8.1), when just one single frequency k is present.

In Fourier space, the wave becomes a delta function. In many situations, waves instead travel in the form of wave packets, as sketched in the lower part of the figure. In Fourier space, the wave packet is a superposition of waves with very similar frequencies. If the peak (in Fourier space) is getting narrower, the packet becomes wider in physical space. We can clearly see two different velocities associated with the wave packet, both of which can be expressed in terms of the dispersion relation:

phase speed	speed of individual crests	$c_p = \frac{\omega(k)}{k}$
group speed	speed of the whole group	$c_g = \frac{d\omega(k)}{dk}$

Derivation of the formula for c_g : The wave function for a single wave number k_0 can be written

$$\phi(x,t) = \widehat{\phi}_0 \, e^{i(k_0 x - \omega(k_0)\, t)} \tag{8.4}$$

A wave packet with the same main wave number is similarly a superposition of different waves

$$\phi(x,t) = \int_{-\infty}^{\infty} \widehat{\phi}_0 \, e^{i(k\, x - \omega(k)\, t)} \, dk$$

where $\widehat{\phi}(k)$ is very nearly zero everywhere but has a sharp peak at $k = k_0$. In that small neighborhood of k_0 we have (by Taylor expansion)

$$\omega(k) = \omega(k_0) + (k - k_0)\, \alpha \qquad \text{where} \quad \alpha = \left.\frac{d\omega}{dk}\right|_{k=k_0} .$$

Hence

$$\begin{aligned} \phi(x,t) &\approx \int_{-\infty}^{\infty} \widehat{\phi}(k) \, e^{i(k\, x - (\omega(k_0) + (k-k_0)\alpha)\, t)} \, dk \\ &= \underbrace{e^{i(k_0\, x - \omega(k_0) t)}}_{\text{Pure harmonic of wave number } k_0} \times \underbrace{\int_{-\infty}^{\infty} \widehat{\phi}(k) \, e^{i(k-k_0)(x-\alpha t)} \, dk}_{\text{Factor providing the envelope of the wave packet}} \end{aligned}$$

We notice that in the second factor, the variables x and t appear only in the combination $x - \alpha t$, showing that this expression translates

Type of wave	Dispersion relation $\omega =$	$c_p = \omega/k$	$c_g = \partial\omega/\partial k$	c_g/c_p	Comment
Gravity wave, deep water	$\sqrt{gk}$	$\sqrt{\frac{g}{k}}$	$\frac{1}{2}\sqrt{\frac{g}{k}}$	$\frac{1}{2}$	g = gravitational acceleration
Gravity wave, shallow water	$\sqrt{gk\tanh kh}$	$\sqrt{\frac{g}{k}\tanh kh}$	$c_p \cdot (c_g/c_p)$	$\frac{1}{2} + \frac{kh}{\sinh(2hk)}$	h = water depth
Capillary wave	$\sqrt{Tk^3}$	$\sqrt{Tk}$	$\frac{3\sqrt{Tk}}{2}$	$\frac{3}{2}$	T = surface tension
Quantum mechanical particle wave	$\frac{hk^2}{4\pi m}$	$\frac{hk}{4\pi m}$	$\frac{hk}{2\pi m}$	2	h = Planck's constant m = particle mass c_g = particle velocity
Light in vacuum	ck	c	c	1	c = 299,792,458 m/s
Light in a transparent medium	$\frac{ck}{n(k)}$	$\frac{c}{n(k)}$	$c_p\left(1 - \frac{kn'(k)}{n(k)}\right)$	$1 - \frac{kn'(k)}{n(k)}$	$n(k)$ = index of refraction

TABLE 2. Dispersion relations and wave speeds for some different types of waves.

with the speed $\alpha = \frac{d\omega}{dk}\big|_{k=k_0}$, i.e. this quantity is equal to the group speed.

Table 2 summarizes the dispersion relations and the two speeds c_p and c_g for some different types of waves (in the case of surface waves, the liquid is assumed to be water, with density $\rho = 1$).

The results in this table have some notable implications:

- Since $c_p > c_g$ for gravity waves, a surfer gets the longest ride is he/she can catch a wave at the end of a wave packet
- Compared to the wave lengths of tsunami waves (hundreds of kilometers), all oceans are shallow. Measurements of the travel times for such waves offered the first means (in the 19th century) of estimating average ocean depths.
- If a twig sticks up in a stream, the phase angle of waves gets locked at it. Since capillary waves have $c_g > c_p$, they can be observed as small stationary ripples upstream of the twig. Gravity waves will instead appear in some relatively narrow sector downstream of the twig.

- The ratio $c_g/c_p = \frac{1}{2}$ for gravity waves can be shown to imply that the distinct V-shaped wake left behind a ship will always form the angle $\arcsin \frac{1}{3} \approx 19.5^o$ to each side of the center line of he wake *independently of the speed of the ship* (intuition might wrongly suggest that the wake should get narrower with increased speeds, like the case is for shocks generated by a fast-moving object).
- In quantum mechanics, a particle's position is undetermined within the width Δx of its wave packet. Fourier analysis will show that Δx and its spread in Fourier space are related by $\Delta x \cdot \Delta k > 1$ (or $\Delta x \cdot \Delta k >$ constant; there is some arbitrariness in how wide one regards a Gaussian pulse to be). De Broglie's relation $k = 2\pi m\nu/h$ relates wave number k to Planck's constant h; here m is the particle mass and ν $(= c_g)$ its velocity. From this, we obtain Heisenberg's uncertainty relation $\Delta x \cdot \Delta \nu > h/(2\pi m)$. In other words, the product of the uncertainties in a particle's position and velocity must always exceed $h/(2\pi m)$.

8.4.3. Stokes' waves. These are periodic translating solutions $\eta(x,t) = \eta(x - ct)$ for the surface elevation in the case of infinite depth and no surface tension; gravity is assumed to be the only restoring force. However, we no longer consider only infinitesimal amplitudes, so the approximation (8.1) amounts only to the first term in an expansion for small a. Including terms up to a^3 can be shown to give

$$\eta(x,t) = a\ \cos(kx - \omega t) + \frac{1}{2}ka^2 \cos 2(kx - \omega t) + \frac{3}{8}k^2a^3 \cos 3(kx - \omega t) + \dots \tag{8.5}$$

where $\omega \approx \left(1 + \frac{1}{2}k^2a^2\right)\sqrt{gk}$ is the next order approximation beyond (8.2) for the dispersion relation. Some observations:

- The governing equations for surface water waves turn out to be nonlinear. Apart for infinitesimal waves, solutions can therefore not be linearly superposed (or multiplied by scalars) to give other solutions.
- In the limit of large amplitude, already Stokes (in 1880) showed that the wave would approach a top angle of 120^o. However, it was noted later that the local wave structure near the top will feature a complicated fine structure.

- The expansion (8.5), if continued to more terms, will diverge before the highest Stokes' wave is reached. This is related to the fact that a Taylor expansion will fail to converge at a radius determined by the nearest singularity. In this case, there will arise unphysical complex singularity points for the real variable a.
- When including further terms in (8.5), the coefficients for the individual modes will no longer be pure powers of a, but will turn into power series expansions in a.
- In real water, high Stokes' waves are never seen. Such a wave is (near the top) unstable to oscillatory disturbances. Another (physically more significant) instability arises already at low amplitudes. The *Benjamin-Feir instability* causes uniform periodic wave trains to loose their periodicity. As a result, wave trains (on deep water) will always exhibit irregularities in amplitude between the individual waves.

8.5. First Order System Formulations for some Linear wave Equations.

It is often practical to rewrite higher order ODEs into systems of first order ones. The main theme of this section is to extend that idea to linear PDEs which describe different types of waves. These first order formulations are usually very well suited for numerical calculations. We will here also use them to analytically determine the different types of translating waves that the equations can feature.

8.5.1. High order ODEs as first order systems. One example suffices to illustrate the general procedure. If we, for example, consider the ODE

$$y''' + \frac{y'' \sin y'}{1 + t^2 y'} - t\, y + 11 \arctan \frac{y}{1+t} = 0 \tag{8.1}$$

we can introduce the new variables

$$\begin{cases} u_1 &= y \\ u_2 &= y' \\ u_3 &= y'' \end{cases} .$$

The ODE then becomes

$$\left\{\begin{array}{lcl} u_1' &=& u_2 \\ u_2' &=& u_3 \\ u_3' &=& -\frac{u_3 \sin u_2}{1+t^2 u_1} + t u_1 - 11 \arctan \frac{u_1}{1+t} \end{array}\right.$$

The initial conditions for $y,\ y'$ and y'' become the initial values for the three variables $u_1,\ u_2$ and u_3. This system, like the more general first order system

$$\left\{\begin{array}{lcl} u_1' &=& f_1(u_1, \ldots, u_n, t) \\ u_2' &=& f_2(u_1, \ldots, u_n, t) \\ \ldots & & \\ u_n' &=& f_n(u_1, \ldots, u_n, t) \end{array}\right. ,$$

can be solved accurately with almost any of the standard numerical techniques for ODEs (such as Runge-Kutta or linear multistep methods).

Analytical solutions of systems of ODEs are rarely available. However, linear systems

$$\frac{d}{dt}\begin{bmatrix} u_1 \\ \vdots \\ u_n \end{bmatrix} = \begin{bmatrix} & \\ & A \\ & \end{bmatrix} \begin{bmatrix} u_1 \\ \vdots \\ u_n \end{bmatrix} + \begin{bmatrix} f_1(t) \\ \vdots \\ f_n(t) \end{bmatrix}$$

where A is a matrix with constant coefficients, form a notable exception. If the f-vector is absent, and A has distinct eigenvalues $\lambda_1, \ldots, \lambda_n$ with corresponding eigenvectors $\underline{v}_1, \ldots, \underline{v}_n$, the general solution becomes

$$\begin{bmatrix} \\ \underline{u} \\ \\ \end{bmatrix} = c_1 e^{\lambda_1 t} \begin{bmatrix} \\ \underline{v}_1 \\ \\ \end{bmatrix} + c_2 e^{\lambda_2 t} \begin{bmatrix} \\ \underline{v}_2 \\ \\ \end{bmatrix} + \ldots + c_n e^{\lambda n t} \begin{bmatrix} \\ \underline{v}_n \\ \\ \end{bmatrix} . \tag{8.2}$$

The coefficients $c_1,\ c_2, \ldots, c_n$ will follow from the initial conditions. Standard ODE text books will discuss the minor modifications that will need to be done to the form of (8.2) in the (usually rare) cases of multiple eigenvalues or missing eigenvectors. If the f-vector is present, *variation of parameters* can be used to obtain a general solution.

8.5.2. High order PDEs as first order systems. A similar approach to the one for ODEs can be used to transform higher order PDEs into lower order ones.

This time, success is not guaranteed, but it still works out in the constant coefficient cases that are of most interest in our present context.

8.5.2.1. *1-D acoustic wave equation.* This equation describes for example the vibrations of a string (8.1) or acoustic waves in 1-D

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2}$$

While keeping u, we can introduce an additional variable v defined by $v_t = c\, u_x$. It then follows that

$$\frac{\partial}{\partial t}\begin{bmatrix} u \\ v \end{bmatrix} = c \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \frac{\partial}{\partial x}\begin{bmatrix} u \\ v \end{bmatrix}. \tag{8.3}$$

8.5.2.2. *2-D acoustic wave equation.* The governing equation in this case is

$$\frac{\partial^2 u}{\partial t^2} = c^2 \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right).$$

This time, we similarly introduce v and w by $v_t = c\, u_x$ and $w_t = c\, u_y$, and obtain

$$\frac{\partial}{\partial t}\begin{bmatrix} u \\ v \\ w \end{bmatrix} = c \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \frac{\partial}{\partial x}\begin{bmatrix} u \\ v \\ w \end{bmatrix} + c \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \frac{\partial}{\partial y}\begin{bmatrix} u \\ v \\ w \end{bmatrix}. \tag{8.4}$$

8.5.2.3. *2-D elastic wave equation.* If a 2-D sheet also possesses elastic properties, the most straightforward derivation of the governing equations (for motions within the $x, y-$plane) leaves them in the form

$$\begin{cases} \frac{\partial u}{\partial t} &= \left(\frac{\partial f}{\partial x} + \frac{\partial g}{\partial y} \right) / \rho \\ \frac{\partial v}{\partial t} &= \left(\frac{\partial g}{\partial x} + \frac{\partial h}{\partial y} \right) / \rho \\ \frac{\partial f}{\partial t} &= (\lambda + 2\mu) \frac{\partial u}{\partial x} + \lambda \frac{\partial v}{\partial y} \\ \frac{\partial g}{\partial t} &= \mu \frac{\partial v}{\partial x} + \mu \frac{\partial u}{\partial y} \\ \frac{\partial h}{\partial t} &= \lambda \frac{\partial u}{\partial x} + (\lambda + 2\mu) \frac{\partial v}{\partial y} \end{cases} . \tag{8.5}$$

Here,

$u,\ v$	local displacements in x-and y-directions
$f,\ g,\ h$	local x-compression, shear, and y-compression respectively
$\rho,\ \lambda,\ \mu$	density, and elastic constants (wrt. compression and shear)

Since the equations (8.5) were directly obtained in first order system form, they are immediately expressible in terms of matrices

$$\frac{\partial}{\partial t}\begin{bmatrix} u \\ v \\ f \\ g \\ h \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1/\rho & 0 & 0 \\ 0 & 0 & 0 & 1/\rho & 0 \\ \lambda+2\mu & 0 & 0 & 0 & 0 \\ 0 & \mu & 0 & 0 & 0 \\ \lambda & 0 & 0 & 0 & 0 \end{bmatrix} \frac{\partial}{\partial x}\begin{bmatrix} u \\ v \\ f \\ g \\ h \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 1/\rho & 0 \\ 0 & 0 & 0 & 0 & 1/\rho \\ 0 & \lambda & 0 & 0 & 0 \\ \mu & 0 & 0 & 0 & 0 \\ 0 & \lambda+2\mu & 0 & 0 & 0 \end{bmatrix} \frac{\partial}{\partial y}\begin{bmatrix} u \\ v \\ f \\ g \\ h \end{bmatrix}.$$

8.5.2.4. *3-D Maxwell's equations.* We obtained also these equations directly in the form of a first order system (8.2). In the case that ε and μ are constants, we can re-arrange the equations into higher order ones for each individual field component (However, this might not be a useful thing to do, since material interfaces, conductors, etc. are often present, in which case the first order formulation is usually preferable. Mainly out of mathematical interest, we will do this re-arrangement in two different ways.

Differentiation of the governing equations. Differentiation of the first and the two last equations in (8.2) with respect to t , z and y respectively gives $(E_x)_{tt} = \frac{1}{\varepsilon}\left((H_z)_{ty} - (H_y)_{tz}\right) = \frac{1}{\varepsilon\mu}\left((E_x)_{yy} + (E_x)_{zz}\right) - \frac{1}{\mu\varepsilon}\left((E_y)_{yx} + (E_z)_{zx}\right)$. Since the electrical field is divergence free $0 = div\ E = (E_x)_x + (E_y)_y + (E_z)_z$, we also have $0 = \frac{\partial}{\partial x} div\ E = (E_x)_{xx} + (E_y)_{yx} + (E_z)_{zx}$. With the last result, the expression for $(E_x)_{tt}$ simplifies to

$$(E_x)_{tt} = \frac{1}{\mu\varepsilon}\left((E_x)_{xx} + (E_x)_{yy} + (E_x)_{zz}\right).$$

Similarly, all the other components of the electric and magnetic fields will each satisfy the 3-D acoustic wave equation (to repeat ourselves, on the very restrictive assumption that ε and μ are constants).

By vector algebra. Writing (8.2) as

$$\frac{\partial}{\partial t}\underline{E} = \frac{1}{\varepsilon}\left(\nabla \times \underline{H}\right) \quad , \quad \frac{\partial}{\partial t}\underline{H} = -\frac{1}{\mu}\left(\nabla \times \underline{E}\right) \tag{8.6}$$

we get

$$\begin{aligned}
\frac{\partial^2 \underline{E}}{\partial t^2} &= \frac{1}{\varepsilon}\left(\nabla \times \frac{\partial \underline{H}}{\partial t}\right) && \text{by (8.6)} \\
&= -\frac{1}{\varepsilon\mu}\left(\nabla \times (\nabla \times \underline{E})\right) && \text{again by (8.6)} \\
&= -\frac{1}{\varepsilon\mu}\left(\nabla\left(\nabla \underline{E}\right) - \nabla^2 \underline{E}\right) && \text{vector identity} \\
&= \frac{1}{\varepsilon\mu}\nabla^2 \underline{E} && \text{since } \nabla \underline{E} = 0
\end{aligned}$$

Similarly,

$$\frac{\partial^2 \underline{H}}{\partial t^2} = -\frac{1}{\varepsilon\mu}\nabla^2 \underline{H},$$

implying again that each component of both fields will satisfy the 3-D acoustic wave equation.

8.5.3. Determination of wave types and speeds. First order formulations of PDEs are often particularly well suited for numerical solution methods. They also often provide a good starting point for analytical work. We next re-visit the examples in the previous section, in order to determine what kinds of waves they support.

8.5.3.1. *1-D acoustic wave equation.* We look for a translating solution to (8.3) of the form

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} u(t-\alpha x) \\ v(t-\alpha x) \end{bmatrix},$$

i.e. for waves with the speed $\sigma = 1/\alpha$. Then $\dfrac{\partial}{\partial t}\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} u \\ v \end{bmatrix}'$ and $\dfrac{\partial}{\partial x}\begin{bmatrix} u \\ v \end{bmatrix} = -\alpha \begin{bmatrix} u \\ v \end{bmatrix}'$. Substitution into (8.3) gives

$$\begin{bmatrix} u \\ v \end{bmatrix}' = -c\alpha \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}', \tag{8.7}$$

which we can write as

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}' = \frac{-1}{c\alpha} \begin{bmatrix} u \\ v \end{bmatrix}'. \tag{8.8}$$

We recognize this as an eigenvalue problem. Since the matrix has eigenvalues ± 1, we get $\alpha = \pm\frac{1}{c}$, and we can conclude that (8.3) admits translating solutions with speeds $\sigma = 1/\alpha_{1,2} = \pm c$

When we next turn to more space dimensions, (8.8) will generalize in a way that no longer allows this immediate interpretation as an eigenvalue problem. That difficulty can be avoided in a couple of ways:

1. Analysis via the determinant,
2. Analysis based on the (complex) wave function.

In this case of equation (8.3):

1. Analysis via the determinant: We can write (8.7) as

$$\left\{ \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + c\alpha \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \right\} \begin{bmatrix} u \\ v \end{bmatrix}' = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

This system has a non-trivial (non-zero) solution if and only if the matrix $\begin{bmatrix} 1 & c\alpha \\ c\alpha & 1 \end{bmatrix}$ is singular. From

$$0 = \det \begin{bmatrix} 1 & c\alpha \\ c\alpha & 1 \end{bmatrix} = 1 - c^2\alpha^2$$

follows again $\alpha_{1,2} = \pm 1/c$, etc.

2. Analysis based on the wave function: Let the wave be

$$\begin{bmatrix} u(x,t) \\ v(x,t) \end{bmatrix} = e^{i(kx-\omega t)} \begin{bmatrix} u_0 \\ v_0 \end{bmatrix}, \tag{8.9}$$

traveling with the velocity $\sigma = \frac{\omega}{k}$. Substituting (8.9) into (8.3) gives, after some simplifications

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} u_0 \\ v_0 \end{bmatrix} = -\frac{\omega}{ck} \begin{bmatrix} u_0 \\ v_0 \end{bmatrix}.$$

From this eigenvalue problem follows $-\frac{\omega}{ck} = \pm 1$, and therefore wave velocities $\sigma = \pm\, c$.We have again found that the waves travel in either direction, with the velocity c.

8.5.3.2. *2-D acoustic wave equation.* 1. Analysis via the determinant: We again to look for translating solutions, now of the form

$$\begin{bmatrix} u(x,y,t) \\ v(x,y,t) \\ w(x,y,t) \end{bmatrix} = \begin{bmatrix} u(t-\alpha x-\beta y) \\ v(t-\alpha x-\beta y) \\ w(t-\alpha x-\beta y) \end{bmatrix}.$$

This solution moves in the $(\alpha,\beta)-$direction with the velocity $\sigma = \frac{1}{\sqrt{\alpha^2+\beta^2}}$. From

$$\frac{\partial}{\partial t}\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} u \\ v \\ w \end{bmatrix}', \quad \frac{\partial}{\partial x}\begin{bmatrix} u \\ v \\ w \end{bmatrix} = -c\alpha\begin{bmatrix} u \\ v \\ w \end{bmatrix}', \quad \frac{\partial}{\partial y}\begin{bmatrix} u \\ v \\ w \end{bmatrix} = -c\beta\begin{bmatrix} u \\ v \\ w \end{bmatrix}'$$

follows

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix}' = -c\alpha\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}\begin{bmatrix} u \\ v \\ w \end{bmatrix}' - c\beta\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}\begin{bmatrix} u \\ v \\ w \end{bmatrix}'.$$

This is no longer in the form of a regular eigenvalue problem, but we can still conclude that it will possess non-trivial solutions if and only if

$$0 = \det\begin{bmatrix} 1 & c\alpha & c\beta \\ c\alpha & 1 & 0 \\ c\beta & 0 & 1 \end{bmatrix} = 1 - c^2(\alpha^2+\beta^2).$$

This shows that there is no 'preferred direction' in the $(x,y)-$plane. There are solutions which translate with speed $\sigma = c$ in any direction.

2. Analysis based on the wave function: Let the wave be

$$\begin{bmatrix} u(x,t) \\ v(x,t) \\ w(x,t) \end{bmatrix} = e^{i(k_x x + k_y y - \omega t)}\begin{bmatrix} u_0 \\ v_0 \\ w_0 \end{bmatrix},$$

traveling with velocity $\sigma = \frac{\omega}{\sqrt{k_x^2+k_y^2}}$. Substitution into (8.4) gives

$$\begin{bmatrix} 0 & k_x & k_y \\ k_x & 0 & 0 \\ k_y & 0 & 0 \end{bmatrix} \begin{bmatrix} u_0 \\ v_0 \\ w_0 \end{bmatrix} = -\frac{\omega}{c} \begin{bmatrix} u_0 \\ v_0 \\ w_0 \end{bmatrix} \tag{8.10}$$

with eigenvalues $-\frac{\omega}{c} = \{\ 0, \sqrt{k_x^2+k_y^2}, -\sqrt{k_x^2+k_y^2}\ \}$, showing that the possible velocities are direction independent: $\sigma = 0,\ +c$, or $-c$. The two choices $+c$ and $-c$ correspond again to the fact that waves can travel in either direction. The zero velocity solution is more surprising. It tells that there, apart from the waves that we are looking for, exists some quantity that does not travel at all.

EXPLAIN TH SPURIOUS VELOCITY FURTHER !!!!

8.5.3.3. *2-D elastic wave equation.* 1. Analysis via the determinant: Analogously to the previous case, looking for translating solutions leads us to consider

$$\begin{aligned} 0 = \det & \begin{bmatrix} 1 & 0 & \alpha/\rho & \beta/\rho & 0 \\ 0 & 1 & 0 & \alpha/\rho & \beta/\rho \\ (\lambda+2\mu)\alpha & \lambda\beta & 1 & 0 & 0 \\ \mu\beta & \mu\alpha & 0 & 1 & 0 \\ \lambda\alpha & (\lambda+2\mu)\beta & 0 & 0 & 1 \end{bmatrix} \\ & = \left[\mu(\alpha^2+\beta^2)-\rho\right] \cdot \left[(\lambda+2\mu)(\alpha^2+\beta^2)-\rho\right]/\rho^2 \ . \end{aligned}$$

There are now two types of possible waves, both with velocities that are direction independent:

$$\begin{array}{ll} P \text{ (pressure or primary) wave:} & c_p = [(\lambda+2\mu)/\rho]^{1/2} \\ S \text{ (shear or secondary) wave:} & c_s = [\mu/\rho]^{1/2} \end{array} \ .$$

In many materials (such as for seismic waves in the earth), the two material constants λ and μ are of similar size. It then follows that $c_p/c_s \approx \sqrt{3}$.

2. Analysis based on the wave function: We are led to the eigenvalue problem

$$\begin{bmatrix} 0 & 0 & k_x/\rho & k_y/\rho & 0 \\ 0 & 0 & 0 & k_x/\rho & k_y/\rho \\ (\lambda+2\mu)k_x & \lambda k_y & 0 & 0 & 0 \\ \mu k_y & \mu k_x & 0 & 0 & 0 \\ \lambda k_x & (\lambda+2\mu)k_y & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} u_0 \\ v_0 \\ f_0 \\ g_0 \\ h_0 \end{bmatrix} = -\omega \begin{bmatrix} u_0 \\ v_0 \\ f_0 \\ g_0 \\ h_0 \end{bmatrix} ,$$

and therefore $-\omega = \{\ 0,\ \pm\sqrt{\frac{\mu}{\rho}}\sqrt{k_x^2+k_y^2},\ \pm\sqrt{\frac{\lambda+2\mu}{\rho}}\sqrt{k_x^2+k_y^2}\}$, representing the same waves as above.

8.5.3.4. *3-D Maxwell's equations.* Omitting the details, the same approaches as above show that there is exactly one type of wave - again direction independent - and that it travels with the velocity $c = 1/\sqrt{\varepsilon\mu}$. Since ε and μ can be found independently by experiments, this relation offers one possibility for determining the speed of light.

8.5.4. Dispersion analysis for FD approximations to wave equations. The first step in finite difference (FD) solutions of wave equations is to replace space derivatives with FD approximations. The velocities of wave propagation will then come to depend on the frequency, the space step, and furthermore, it will no longer be isotropic (same in all directions). Of the two approaches above to determine wave speeds, the second one carries immediately over to the discrete case, and becomes an essential tool for such *dispersion analysis* of FD schemes. We give examples of this analysis below for when space derivatives are replaced by centered second order finite differences (FD2) in the cases of the equations (8.3), (8.4), and (8.2).

8.5.4.1. *Dispersion analysis for the FD2 approximation to the 1-D wave equation.* Let the grid spacing be h, and approximate $\frac{\partial}{\partial x}u(x,t)$ by $\frac{u(x+h,t)-u(x-h,t)}{2h}$ (and similarly for $v(x,t)$). Then

$$\frac{\partial}{\partial x}e^{i(kx-\omega t)} \approx \frac{e^{ikh}-e^{-ikh}}{2h}e^{i(kx-\omega t)} = i\ \frac{\sin kh}{h}e^{i(kx-\omega t)}.$$

Substituting

$$\begin{bmatrix} u(x,t) \\ v(x,t) \end{bmatrix} = e^{i(kx-\omega t)}\begin{bmatrix} u_0 \\ v_0 \end{bmatrix}$$

into the spatially discretized version of (8.3) will therefore lead to

$$-i\omega\begin{bmatrix} u_0 \\ v_0 \end{bmatrix} = c\begin{bmatrix} 0 & i\frac{\sin\ kh}{h} \\ i\frac{\sin\ kh}{h} & 0 \end{bmatrix}\begin{bmatrix} u_0 \\ v_0 \end{bmatrix},$$

i.e.

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}\begin{bmatrix} u_0 \\ v_0 \end{bmatrix} = -\ \frac{\omega h}{c\ \sin kh}\begin{bmatrix} u_0 \\ v_0 \end{bmatrix}.$$

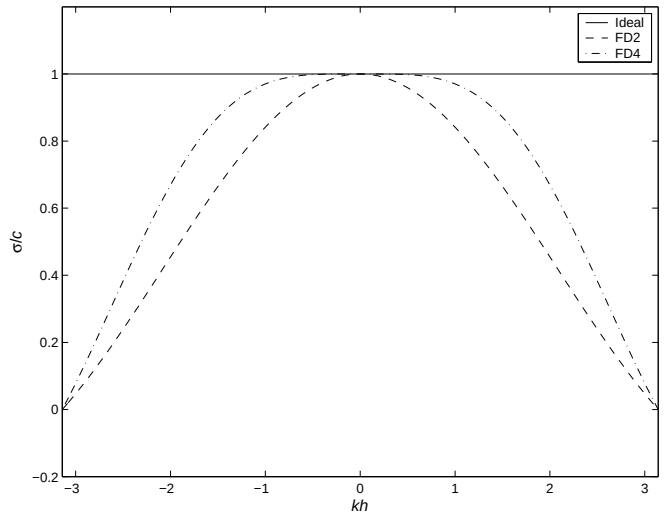


FIGURE 8.5.1. Plot of σ/c (numerical wave velocity relative to its ideal value) displayed as a function of kh for second and fourth order centered finite differences (denoted FD2 and FD4 respectively).

Since the eigenvalues to the matrix are ± 1, it follows that the wave velocities are

$$\sigma = \frac{\omega}{k} = \pm\, c\, \frac{\sin kh}{kh}. \tag{8.11}$$

Fora fixed value of k and letting $h \to 0$ (i.e. letting the spatial grid become increasingly refined), we recover the exact result $\sigma = \pm c$. Choosing the plus sign, Figure 8.5.1 shows σ/c as a function of kh, according to (8.11) (labeled FD2), and also the corresponding result for fourth order centered differences (labeled FD4). Instead of the ideal result ($\sigma/c \equiv 1$), we see a reasonable accuracy in the wave velocity only for very small values of $|kh|$. Since the highest mode k that can be represented on a grid with spacing h satisfies $k = \pm\frac{\pi}{h}$, the domain for kh in the figure is chosen as $[-\pi, \pi]$. The highest modes (with $kh = \pm\pi$) are the 'saw-tooth modes' $e^{\pm\, i\frac{\pi}{h}x}$, which swap sign each time x is incremented by h. For these modes, the FD approximations return a zero value for the derivative at each grid point, in agreement with the fact that the FD curves in Figure 8.5.1 go to zero when $kh = \pm\pi$.

8.5.4.2. *Dispersion analysis for the FD2 approximation to the 2-D wave equation.* The additional effect that now needs to be included is that the computed wave velocities will also depend on their direction of travel in the $x, y-$plane. Let the grid

spacing be h in both the x and y grid directions. We approximate $\frac{\partial}{\partial x}u(x,y,t)$ by $\frac{u(x+h,y,t)-u(x-h,y,t)}{2h}$ and similarly for $\frac{\partial}{\partial y}u(x,y,t)$, i.e. the multiplicative factor caused by the $\frac{\partial}{\partial x}$ and $\frac{\partial}{\partial y}$ - operators, when applied to $e^{i(k_x x+k_y y-\omega t)}$, become no longer ik_x and ik_y but instead $\frac{e^{ik_x h}-e^{-ik_x h}}{2h} = i\frac{\sin\ k_x h}{h}$ and $\frac{e^{ik_y h}-e^{-ik_y h}}{2h} = i\frac{\sin\ k_y h}{h}$, respectively. In place of (8.10), we get

$$\begin{bmatrix} 0 & \frac{\sin\ k_x h}{h} & \frac{\sin\ k_y h}{h} \\ \frac{\sin\ k_x h}{h} & 0 & 0 \\ \frac{\sin\ k_y h}{h} & 0 & 0 \end{bmatrix} \begin{bmatrix} u_0 \\ v_0 \\ w_0 \end{bmatrix} = -\frac{\omega}{c} \begin{bmatrix} u_0 \\ v_0 \\ w_0 \end{bmatrix}$$

and $-\frac{\omega}{c} = \{\ 0,\ \pm\frac{1}{h}\sqrt{(\sin\ k_x h)^2 + (\sin\ k_y h)^2}\}$. Again, waves can travel in any direction, but with velocities

$$\sigma = \frac{\omega}{\sqrt{k_x^2+k_y^2}} = \pm\frac{c}{h}\,\frac{\sqrt{(\sin\ k_x h)^2 + (\sin\ k_y h)^2}}{\sqrt{k_x^2+k_y^2}}.$$

For a wave with wave number k that propagates with the angle θ relative to the $x-$axis, we have $k_x = k\cos\theta$, $k_y = k\sin\theta$, and therefore

$$\sigma = \frac{c}{kh}\sqrt{(\sin\ (kh\ \cos\theta))^2 + (\sin\ (kh\ \sin\theta))^2}.$$

and

$$\begin{aligned} \frac{\sigma}{c} &= \frac{1}{kh}\sqrt{(\sin\ (kh\ \cos\theta))^2 + (\sin\ (kh\ \sin\theta))^2} \\ &\approx 1 - \frac{(kh)^2}{24}(3+\cos 4\theta) + \frac{(kh)^4}{11520}(65+36\cos 4\theta - 5\cos 8\theta) + \dots \end{aligned}$$

Figure 8.5.2 illustrates this relationship, i.e. how the wave velocity in the discrete scheme varies with the wave direction relative to the grid for different values of kh.

8.5.4.3. *Dispersion analysis for the FD2 approximation to the 3-D Maxwell's equations.* ..... TO BE WRITTEN !!! .....

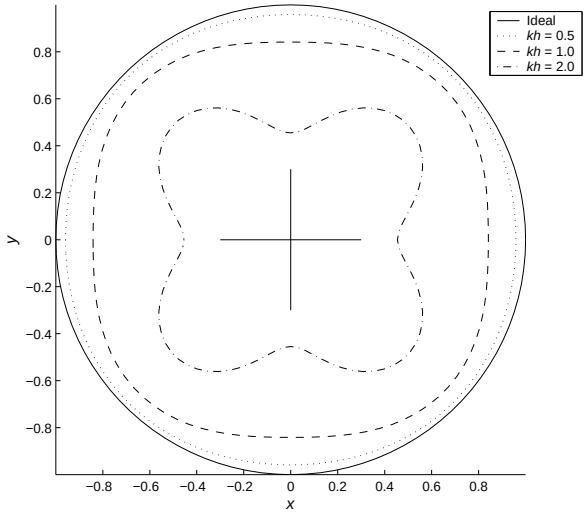


FIGURE 8.5.2. Values of σ/c (numerical wave velocity relative to is ideal value), displayed as radial distance from the origin for different values of kh.

8.6. Analytic solutions of the acoustic wave equation.

We have already come across the acoustic wave equation in 1-D, 2-D and 3-D. In n-D it takes the form

$$\frac{\partial^2 u}{\partial t^2} = c^2 \left(\frac{\partial^2 u}{\partial x_1^2} + \frac{\partial^2 u}{\partial x_2^2} + ... + \frac{\partial^2 u}{\partial x_n^2} \right). \tag{8.1}$$

This can be written more compactly as $\frac{\partial^2 u}{\partial t^2} = c^2 \nabla^2 u$ where $u(\underline{x}, t)$ is a scalar function of $\underline{x} = (x_1, x_2, ..., x_n)$ and, as before, $\nabla = (\frac{\partial}{\partial x_1}, \frac{\partial}{\partial x_2}, ..., \frac{\partial}{\partial x_n})$ i.e. $\nabla^2 = \frac{\partial^2}{\partial x_1^2} + \frac{\partial^2}{\partial x_2^2} + ... + \frac{\partial^2}{\partial x_n^2}$.

Since (8.1) has a second derivative in time, we will need two initial conditions to get a solution started:

$$\begin{cases} u(\underline{x}, 0) & = f(\underline{x}) \\ u_t(\underline{x}, 0) & = g(\underline{x}) \end{cases} . \tag{8.2}$$

In the three subsections below, we discuss briefly the general solution to (8.1), (8.2) in 1-D, in n-D, and finally, we show how (8.1) simplifies in the case of radial symmetry. The analytic solutions all assume constant material properties, and they cannot easily be extended to irregular domains. They can still be useful for obtaining general

insights about the character of wave equations. However, in most practical situations, numerical solutions are needed.

8.6.1. Acoustic wave equation in 1-D; d'Alembert's solution. The general solution to

$$\frac{\partial^2 u}{\partial t^2} = c^2 \frac{\partial^2 u}{\partial x^2} \tag{8.3}$$

can be written down as

$$u(x,t) = F(x-ct) + G(x+ct) \tag{8.4}$$

where F and G are completely arbitrary functions. Verification that (8.4) satisfies (8.3) is straightforward:

$$u_{xx} = \quad F''(x-ct) + \quad G''(x-ct)$$

and

$$u_{tt} = c^2 F''(x-ct) + c^2 G''(x-ct) = c^2 \; u_{xx}$$

Hence (8.3) holds. An easy way to arrive at (8.4) (rather than just verifying it) is by Fourier analysis - cf. the exercises in Part V.

For the common situation when

$$\begin{cases} u(x,0) & = f(x) \\ u_t(x,0) & = g(x) \end{cases} \tag{8.5}$$

are given as initial conditions (rather than $F(x)$ and $G(x)$), we need a way to 'translate' $f(x)$, $g(x)$ over into the functions $F(x)$ and $G(x)$. This is achieved by *d'Alembert's formula*, stating that the unique solution to (8.3), (8.5) is

$$u(x,t) = \frac{1}{2}\left[f(x-ct) + f(x+ct) + \frac{1}{c} \int_{x-ct}^{x+ct} g(\xi)\, d\xi \right]. \tag{8.6}$$

Derivation of (8.6): As just indicated, the idea for the proof is to convert $f(x)$ and $g(x)$ into $F(x)$ and $G(x)$. From (8.5) and (8.4) follow

$$\begin{cases} u(x,0) = f(x) & = \quad F(x) + \quad G(x) \\ u_t(x,0) = g(x) & = -cF'(x) + c\, G'(x) \end{cases} \tag{8.7}$$

After differentiating the top equation (8.7), we can solve the system for $F'(x)$ and $G'(x)$ in terms of $f(x)$ and $g(x)$:

$$\begin{cases} F'(x) &= \frac{1}{2}\left[f\,'(x) - \frac{1}{c}\,g(x)\right] \\ G'(x) &= \frac{1}{2}\left[f\,'(x) + \frac{1}{c}\,g(x)\right] \end{cases}$$

Integrating these give $F(x)$ and $G(x)$

$$\begin{cases} F(x) &= \frac{1}{2}\left[f\,'(x) - \frac{1}{c}\,\int_0^x g(\xi)d\xi\right] + c_1 \\ G(x) &= \frac{1}{2}\left[f\,'(x) + \frac{1}{c}\,\int_0^x g(\xi)d\xi\right] + c_2 \end{cases}$$

and by (8.4)

$$\begin{aligned} u(x,t) &= F(x-ct) + G(x+ct) = \\ &= \frac{1}{2}\left[f(x-ct) - \frac{1}{c}\,\int_0^{x-ct} g(\xi)\,d\xi\right] + \frac{1}{2}\left[f(x-ct) + \frac{1}{c}\,\int_0^{x+ct} g(\xi)\,d\xi\right] + c_3 = \\ &= \frac{1}{2}\left[f(x-ct) + f(x+ct) + \frac{1}{c}\,\int_{x-ct}^{x+ct} g(\xi)\,d\xi\right] + c_3 \end{aligned}$$

It remains only to note that setting $t = 0$ shows that the constant $c_3 = 0$.

8.6.2. Acoustic wave equation in *n*-D.. One can write down formulas in n-D that, completely analogously to d'Alembert's formula in 1-D, give the $u(\underline{x}, t)$ which satisfies (8.1) in terms of the initial conditions (8.2). Rather than doing that, we will here just summarize a few things that these formulas tell us:

1-D: The solution at a point $u(x,t)$ depends on $f(x-ct)$, $f(x+ct)$ and on all the g-values in-between $x-ct$ and $x+ct$. As a consequence, the general solution describes two very different kinds of outgoing wave motions:

(1) Cleanly translating pulses (if started by $f(x) \neq 0$ within some small area only, and $g(x) = 0$ everywhere). Such waves leave a zone of perfect silence behind them,

(2) Disturbances which spread out throughout a complete interval (whenever $g(x) \neq 0$ initially).

Both of these situations can arise on a tight string: Case 1 if the string is locally deformed to feature a small 'hump' but is otherwise straight, and is then released, and Case 2 if a straight string is lightly hit at one point.

2-D: (and also higher *even* dimensions): The solution $u(\underline{x}, t)$ depends both on f and g everywhere within a distance ct from the point $\underline{x}$. Both solution types (with $f(\underline{x}) \neq 0,\ g(\underline{x}) = 0$ and $f(\underline{x}) = 0,\ g(\underline{x}) \neq 0$ resp.) fall in the second category above, i.e. outgoing signals never leave any zone of silence behind.

3-D: (and also higher *odd* dimensions): The solution $u(\underline{x}, t)$ at the point $\underline{x}$ depends on the values of f and g only on *the surface of a sphere* centered at $\underline{x}$ and with radius ct. As a consequence, however we initiate a sound signal at a point in 3-D, it will leave perfect silence behind itself as it travels out. This very remarkable property makes speech possible in 3-D. These outgoing sound signals attenuate with the distance traveled, but undergo no other changes. The conclusion is that 'clean speech' is possible only in odd dimensions from three and up.

8.6.3. Form of the wave equation in radial symmetry. In the special case of signals emerging from one point and propagating out with equal strength in all directions, the wave equation can be simplified from (8.1) to a form that depends on t and $r = \sqrt{x_1^2 + x_2^2 + ... + x_n^2}$ only:

$$\frac{\partial^2 u}{\partial t^2} = c^2 \left(\frac{\partial^2 u}{\partial r^2} + \frac{n-1}{r} \frac{\partial u}{\partial r} \right) \tag{8.8}$$

Derivation of (8.8): The chain rule gives

$\frac{\partial u}{\partial x_i} = \frac{\partial u}{\partial r} \frac{\partial r}{\partial x_j} = \frac{\partial u}{\partial r}\ x_i\ \frac{1}{r}$ (noting that $r^2 = x_1^2 + x_2^2 + ... + x_n^2$ implies $2\ r\ \frac{\partial r}{\partial x_i} = 2x_i$),
$\frac{\partial^2 u}{\partial x_i^2} = \frac{\partial}{\partial x_i} \left(\frac{\partial u}{\partial x_i} \right) = \frac{\partial}{\partial r} \left(\frac{\partial u}{\partial r} \frac{x_i}{r} \right) \frac{x_i}{r} = \frac{\partial^2 u}{\partial r^2} \left(\frac{x_i}{r} \right)^2 + \frac{\partial u}{\partial r} \frac{1}{r} - \frac{\partial u}{\partial r} \frac{x_i^2}{r^3}$

Summation over i now gives $\sum_{i=1}^{n} \frac{\partial^2 u}{\partial x_i^2} = \frac{\partial^2 u}{\partial r^2} + \frac{n-1}{r} \frac{\partial u}{\partial r}$, from which (8.8) follows.

8.7. Hamilton's equations.

8.7.1. Hamiltonian systems from conservation laws. In nature systems often evolve according to some fundamental *conservation law* such as conservation of energy, momentum, etc. Such a fundamental conservation law clearly poses limitations on the ways in which the system are allowed to evolve. Suppose we have a

system with an even number of degrees of freedom denoted by $\mathbf{z}$.In mechanical systems $\mathbf{z}$ can for instance consist of generalized position and momentum coordinates. Since there is a momentum coordinate for each position coordinate, it follows that $\mathbf{z}$ consists of the $2n$ coordinates,

$$z = \begin{bmatrix} q_1 \\ \vdots \\ q_n \\ p_1 \\ \vdots \\ p_n \end{bmatrix},$$

where the q's denote the n position coordinates and p the n corresponding momentum coordinates. Suppose further that the system (it does not need to be a mechanical system) evolves evolves in such a way that a quantity, $H(\mathbf{z})$, remains constant,

$$H(\mathbf{z}(t)) = \text{constant}, \ t \geq 0, \tag{8.1}$$

where we have now explicitly indicated the time dependence of the variables $\mathbf{z}$. The exact value of the constant is determined by its value at any fixed time, more often than not by its initial value. One can think of this conserved quantity as imposing a restriction on the $2n$ coordinates $\mathbf{z}$. In the absence of any conserved quantities, the system is free to evolve in all of the $2n$ dimensional space determined by its $2n$ coordinates. The restriction imposed by the conserved quantity has the effect of limiting the space in which the system is allowed to evolve—instead of having the full $2n$ dimensional space available, motion is now restricted to an $2n-1$ dimensional 'surface'. Let us consider how one might describe the motion on this surface.

From (8.1) it follows that

$$\frac{dH}{dt} = \sum_{j=1}^{2n} \frac{\partial H}{\partial z_j} \dot{z}_j = 0,$$

which can be more conveniently written as

$$\nabla H \cdot \dot{\mathbf{z}} = 0.$$

This expression simply says that the motion is tangent to the normal ∇H of the surface. Unfortunately this this not provide sufficient constraints to determine the motion uniquely—all we really know is that $\dot{\mathbf{z}}$ should always point in the direction of the tangent to the surface, i.e. orthogonal to ∇H. The most general solution to this problem—motivated more fully in the exercises— is therefore,

$$\dot{\mathbf{z}} = J\nabla H \tag{8.2}$$

where J is any non-singular $2n$ dimensional *antisymmetric* matrix, $J = -J^T$. The requirement for J to be non-singular has the effect that the tangent $\dot{\mathbf{z}}$ is well-defined for any nonzero ∇H. Of course there are many matrices meeting these requirements; each choice is a perfectly good candidate to describe the motion on the $2n-1$ dimensional surface. In order to pin this down further and specify J uniquely, one has to know more about the particular system. Systems of this type are known as Hamiltonian systems and they have many remarkable properties that we cannot investigate in full in this book. We'll come back to some of them. For the time being note that one has to specify two things in order to define a Hamiltonian system uniquely—one has to specify the *Hamiltonian* H and the so-called symplectic matrix J. Appropriate choices for Jhave to come from the applications. For the time being we note that arguably the most common choice for J turns out to be

$$J = \begin{bmatrix} 0 & I \\ -I & 0 \end{bmatrix},$$

where I is the n-dimensional identity matrix. With this choice of J, Hamilton's equations assume their so-called *canonical* form. In the next section we see how additional assumptions lead to an appropriate choice of H. Before we proceed, we give a simple example from mechanics.

Example: The figure shows a mass m attached to one end of a light, rigid rod of length ℓ. The other end of the rod is attached to a smooth hinge so that it can swing freely. The total energy of the mass is the sum of its kinetic and potential energies, given by $\frac{1}{2}m\ell^2\dot{\theta}^2$ and $-mg\ell\cos\theta$, respectively. Since the total energy is conserved a good

choice of the Hamiltonian H is

$$H(q,p) = \frac{1}{2m\ell^2}p^2 - mg\ell\cos q,$$

where the generalized position and momentum coordinates are given by

$$\begin{aligned} p &= m\ell^2\dot{\theta} \\ q &= \theta \end{aligned}$$

With the canonical choice of symplectic matrix, Hamilton's equations become,

$$\begin{aligned} \dot{q} &= \frac{\partial H}{\partial p} = \frac{1}{m\ell^2}p, \\ \dot{p} &= -\frac{\partial H}{\partial q} = -mg\ell\sin\theta. \end{aligned}$$

In the original variables this becomes

$$\begin{aligned} \dot{\theta} &= \dot{\theta} \\ \ddot{\theta} &= -\frac{g}{\ell}\sin\theta. \end{aligned}$$

Note how the conservation of energy constrains the system to move along the curves described by,

$$H(q,p) = \frac{1}{2m\ell^2}p^2 - mg\ell\cos q = E,$$

where E is the total energy of the system determined by the initial values of q and p. Figure8.7.1 shows the curves for different values of E; solutions depicted in this way ware known as phase-space solutions. Phase space is therefore just the coordinate space describing the motion of the system. The only essential information that one cannot deduce directly from this Figure, is the exact position and momentum of the pendulum at a given time.

Apart from that, the phase space solution gives us complete *qualitative* information of the motion of the system. The most prominent feature of the Figure 8.7.1 is that it is divided into two distinct parts. For small enough energies (values of E) the

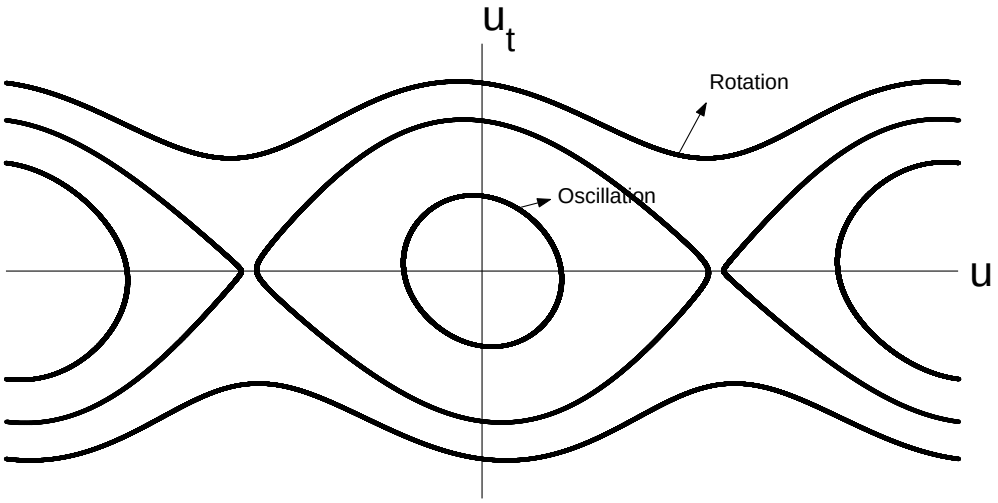


FIGURE 8.7.1. The phase space curves of the pendulum.

orbits form closed loops. This means that the systems periodically returns to its starting position—it oscillates with a fixed period. If, on the other hand, the initial energy is large enough, the pendulum goes 'over the top' and starts rotating. The orbit separating these two motions, the separatrix, is clearly a delicate structure—it is neither oscillating nor rotating. It means that we have to launch the pendulum with the exact amount of energy for it to swing up to the top where it has to come to rest. Just slightly less energy and it performs large amplitude oscillations. A fraction too much energy and it rotates very slowly, each rotation barely making it over the top.

Come to think of it, the representation of Figure 8.7.1 is not entirely satisfactory. We do not really want to make any distinction between angles 2π apart—physically it amounts to the same position. One way of identifying angles 2π apart is to roll up Figure 8.7.1 into a cylinder. As the systems goes round the cylinder it ends up where it started. Thus, even rotation become periodic motion. The situation is illustrated in Figure .

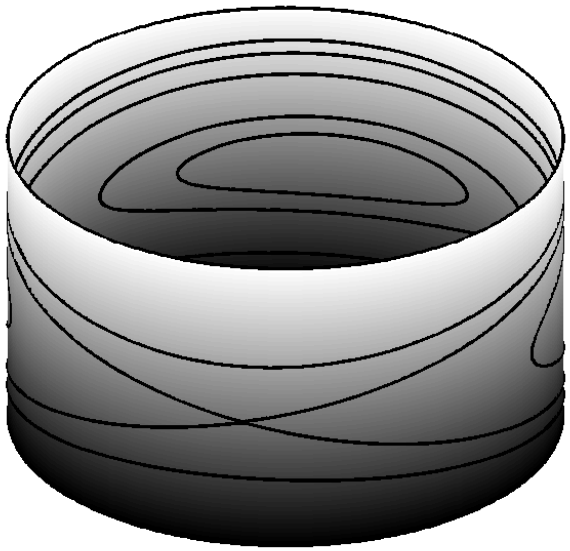


FIGURE 8.7.2. The cylindrical phase space of the pendulum.

Exercise: Let J be any anti-symmetric matrix, $J^T = -J$. Show that $\mathbf{u}^T J \mathbf{u} = 0$ for any vector $\mathbf{u}$.

Exercise: Let J be any *odd* dimensional, anti-symmetric matrix, $J^T = -J$. Show that J is singular.

Exercise: Let $\mathbf{u} = S\mathbf{a}$ for any matrix $\mathbf{a}$. Show that $\mathbf{u}$ is orthogonal to $\mathbf{a}$ if and only if S is an anti-symmetric matrix. Is it possible to determine J uniquely? Why?

8.7.2. Hamilton's equations for a potential system. In this section we show how additional assumptions lead to natural choices for the Hamiltonian H and symplectic matrix J. Let us assume that we are given a real potential function $\theta(\mathbf{q}, t)$

where $\mathbf{q} \in \mathbb{R}^n$. Computing the gradient of θ, we define the following quantities,

$$\omega = -\frac{\partial \theta}{\partial t},\ p_j = \frac{\partial \theta}{\partial q_j},\ j = 1, \ldots, n. \tag{8.3}$$

The only additional assumption is that these quantities are related by an function H,

$$\omega = H(\mathbf{q}, \mathbf{p}, t). \tag{8.4}$$

Let us now investigate the consequences of our assumptions. From (8.3) follows that

$$\frac{\partial p_j}{\partial t} + \frac{\partial \omega}{\partial q_j} = 0,\ j = 1, \ldots, n \tag{8.5}$$

$$\frac{\partial p_i}{\partial q_j} - \frac{\partial p_j}{\partial q_i} = 0,\ i, j = 1, \ldots, n. \tag{8.6}$$

Using the chain rule, the relation (8.4) gives us

$$\frac{\partial \omega}{\partial q_j} = \frac{\partial H}{\partial q_j} + \sum_{i=1}^{n} \frac{\partial H}{\partial p_i} \frac{\partial p_i}{\partial q_j},\ j = 1, \ldots, n.$$

Substituting this into (8.5), gives

$$\frac{\partial p_j}{\partial t} + \frac{\partial H}{\partial q_j} + \sum_{i=1}^{n} \frac{\partial H}{\partial p_i} \frac{\partial p_i}{\partial q_j} = 0,\ j = 1, \ldots, n, \tag{8.7}$$

and using (8.6) gives us

$$\frac{\partial p_j}{\partial t} + \sum_{i=1}^{n} \frac{\partial H}{\partial p_i} \frac{\partial p_j}{\partial q_i} = -\frac{\partial H}{\partial q_j},\ j = 1, \ldots, n.$$

The total change in $\mathbf{p}$ with time is given by

$$\frac{dp_j}{dt} = \frac{\partial p_j}{\partial t} + \sum_{i=1}^{n} \frac{\partial p_j}{\partial q_i} \frac{dq_i}{dt},\ j = 1, \ldots, n,$$

or, making use of (8.7),

$$\begin{aligned}\frac{dp_j}{dt} &= -\frac{\partial H}{\partial q_j} - \sum_{i=1}^{n} \frac{\partial H}{\partial p_i}\frac{\partial p_j}{\partial q_i} + \sum_{i=1}^{n} \frac{\partial p_j}{\partial q_i}\frac{dq_i}{dt}, \\ &= -\frac{\partial H}{\partial q_j} + \sum_{i=1}^{n} \left[-\frac{\partial H}{\partial p_i} + \frac{dq_i}{dt}\right]\frac{\partial p_j}{\partial q_i}, \ j = 1, \ldots, n.\end{aligned}$$

Thus, along the characteristic curves,

$$\frac{dq_j}{dt} = \frac{\partial H}{\partial p_j}, \tag{8.8}$$

we have

$$\frac{dp_j}{dt} = -\frac{\partial H}{\partial q_j}, \tag{8.9}$$

for $j = 1, \ldots, n$. Thus we have arrived at Hamilton's equations in canonical form. It is worth calculating the total time variation of $H(\mathbf{x}, \mathbf{k}, t)$,

$$\frac{dH}{dt} = \sum_{j=1}^{n} \left[\frac{\partial H}{\partial q_j}\frac{dq_j}{dt} + \frac{\partial H}{\partial p_j}\frac{dp_j}{dt}\right] + \frac{\partial H}{\partial t},$$

making use of Hamilton's equations (8.8) and (8.9) it follows that

$$\frac{dH}{dt} = \frac{\partial H}{\partial t}.$$

If H does not depend explicitly on time, i.e. $\frac{\partial H}{\partial t} = 0$, then it does not change with time.

From two assumptions, the existence of a potential function and a relationship between its partial derivatives, we were naturally led to Hamilton's equations in canonical form. These assumptions specified both the Hamiltonian function as well as the symplectic structure J. In the following example we show how these assumptions can be realized in practice.

> **Example:** Imagine a complex wave field where the frequency and wave numbers can change in time and from one location to another. If this happens very abruptly and in a completely arbitrary way, it may not possible to identify any specific pattern. Somehow we need to identify, at least locally, a specific frequency and wave number. A

very general description of a wave is given by

$$\psi(\mathbf{x},t) = A(\mathbf{x},t)\exp i\theta(\mathbf{x},t) \tag{8.10}$$

where $\mathbf{x} \in \mathbb{R}^2$, $A(\mathbf{x},t)$ is a complex amplitude and $\theta(\mathbf{x},t)$ real, represents the phase. We again use the convention that the waves we are interested in are given by the real part of (**??**), without indicating it in our notation. Also note that we can safely assume that $\theta(\mathbf{0},0) = 0$; if not we subtract the constant value from the phase and absorb it into the amplitude. This is a very general description and if $\theta(\mathbf{x},t)$ varies too much, it will not be possible to identify anything that we can associate with a wave. We therefore need to put some restrictions on $\theta(\mathbf{x},t)$. Accordingly we assume that its Taylor expansion is dominated by its first order terms. In that case it is natural to define the local frequency and wave number by,

$$\omega = -\frac{\partial\theta}{\partial t},\ k_j = \frac{\partial\theta}{\partial x_j},\ j = 1,\ldots,n. \tag{8.11}$$

We know that the frequency and wave numbers are related through a dispersion relation,

$$\omega = W(\mathbf{x},\mathbf{k},t), \tag{8.12}$$

which means that the wave numbers evolve along the characteristic curves according to Hamilton's equations,

$$\begin{aligned} \frac{dx_j}{dt} &= \frac{\partial W}{\partial k_j}, \\ \frac{dk_j}{dt} &= -\frac{\partial W}{\partial x_j}, \end{aligned} \tag{8.13}$$

for $j = 1,\ldots,n$ with the dispersion relation acting as the Hamiltonian function.

The next example investigates a specific choice for the dispersion relation.

Example: For deep-water wave, we know that the dispersion relation in the absence of a current is given by

$$W(\mathbf{x}, \mathbf{k}, t) = \sqrt{g\,|\mathbf{k}|}.$$

Since W does not depend on t, it remains constant along the characteristic curves. Since it does not depend on $\mathbf{x}$, the wave number $\mathbf{k}$ also remain constant along the characteristics. The characteristics themselves are straight lines, given by

$$\begin{aligned} \frac{dx_j}{dt} &= \frac{\partial W}{\partial k_j} \\ &= \frac{\frac{1}{2}\sqrt{g}k_j}{|\mathbf{k}|^{3/2}}. \end{aligned}$$

Since $\mathbf{k}$ remains constant along these curves, this integrates to

$$x_j = \frac{\partial W}{\partial k_j} t + \text{const.}$$

In this example, the dispersion relation W does not depend on $\mathbf{x}$. Therefore the frequency ω is the same everywhere, i.e. for all values of $\mathbf{x}$. Moreover, Hamilton's equations then imply that the wave numbers are constant in time. Thus, if one drives a wave field for which the dispersion relation depends on $\mathbf{k}$ but not on $\mathbf{x}$, with a single frequency and wave number, this frequency will be propagated throughout the domain—one observes a monochromatic wave with a single frequency and wave number provided by the forcing.

CHAPTER 9

DIMENSIONAL ANALYSIS

9.1. Introduction.

How does one approach the question as to how dinosaurs ran? Did they stumble about clumsily, or were they agile like monkeys? The problem is that no one has ever seen a live dinosaur, and the exact way they moved cannot be studied by direct observation. On the other hand, it is safe to assume that dinosaurs were subject to the same physical laws as you and I, and all animals moving about today. It is probably for that reason that one already has an intuition that their movement should depend on their size (dinosaurs came in all different shapes and sizes). The bigger the animal, the more clumsy they are. Or is that true? An elephant can easily out-sprint a tortoise and the relatively small cheetah is fastest of them all.

The more general question that we address in this chapter, is to investigate how basic physical laws determine what is possible and what is not. By a careful analysis a seemingly intractable problem may yield surprisingly accurate numerical answers and a complicated problem may become tractable if one can identify the small quantities that can be neglected.

The main physical principle that will be exploited in this chapter is the fact that the laws of nature should not depend on the units that we use to express them in. This is a good thing too, otherwise the physics conducted in the USA where the US Customary Units (slightly different from the old Imperial Units) are still commonly used, might have been different than in countries where the SI units (m, kg, s, A, K, etc) are the units of choice.

9.2. Buckingham's PI-Theorem.

For the discussion in this section we have benefitted from the notes by Harald Hanche-Olsen (2004).

Consider a physical equation in the form

$$A_1 + A_2 + \cdots + A_s = 0 \tag{9.1}$$

where each A_j is a separate term in the equation. As an example a famous equation[1] can be written as

$$E - mc^2 = 0. \tag{9.2}$$

Let us see what happens if we decide to change the units of one of the physical quantities. Let us for example decide to measure the speed of light in km/h instead of m/s. Since E has exactly the same units as mc^2 the fact that we change the units of any physical quantity does not change the physics. The value of the energy E may change, but not the physics. This is a very general observation: the behavior of a physical system should not depend on the chooice of units. If we therefore scale any of the quantities in (9.1) by a factor of say, λ, every term in the equation should scale by the same power of λ so that it can factor out of the equation, leaving the physics intact. By exploiting this simple idea, we'll arrive at the celebrated Buckingham's Theorem.

The physical quantities we are interested in are denoted by $R_1, \ldots, R_n$, measured in a fixed system of units such as the SI system whose basic units are the meter, kilogram, second, ampere and kelvin (m, kg, s, A, K). In general we denote the basic units of our system by $F_{1,} \ldots, F_m$and we write

$$R_j = \rho_j \, [R_j] \tag{9.3}$$

where ρ_j is the value of the physical quantity and $[R_j]$ denotes the units of R_j. Thus we can write

$$[R_j] = \prod_{i=1}^{m} F_i^{a_{ij}}, \; j = 1, \ldots, n. \tag{9.4}$$

[1]It even has its own biography.

It is important to note that the fundamental units are independent, i.e. one unit cannot be expressed in terms of the others. Thus if

$$\prod_{i=1}^{m} F_i^{x_i} = 1$$

it follows that

$$x_1 = x_2 = \cdots = x_m = 0.$$

Let us now change our units to (converting from meter to foot, for example)

$$\widehat{F}_i = x_i^{-1} F_i,\ i = 1, \ldots, m$$

where the x_i are positive numbers that depend on the units. The physical quantities R_j can now be expressed in terms of the new units as

$$R_j = \widehat{\rho}_j \left[\widehat{R_j}\right].$$

We can compute the relationship between the new and old system of units,

$$\begin{aligned} R_j &= \rho_j \prod_{i=1}^{m} F_i^{a_{ij}} \\ &= \underbrace{\rho_j x_1^{a_{1j}} \cdots x_m^{a_{mj}}}_{\widehat{\rho}_j} \widehat{F}_1^{a_{1j}} \cdots \widehat{F}_m^{a_{mj}}. \end{aligned}$$

Thus we find that

$$\widehat{\rho}_j = \rho_j \prod_{i=1}^{m} x_i^{a_{ij}}. \tag{9.5}$$

Example. Let us use this formalism to convert speed measured in m/s to speed measured in km/h. The fundamental units are $F_1 = m$ and $F_2 = s$ and the units of speed, R_1, is given by $[R_1] = F_1^1 F_2^{-1}$ so that $a_{11} = 1$ and $a_{12} = -1$. With $\widehat{F}_1 = \text{km}$ and $\widehat{F}_2 = h$, it follows that $x_1^{-1} = 1000$ and $x_2^{-1} = 3600$. Thus $\widehat{\rho} = \rho x_1^1 x_2^{-1} = 3.6\rho$, and 1m/s becomes 3.6km/h.

The dimension matrix

$$A = \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix} \tag{9.6}$$

of the physical quantities $R_1, \ldots, R_n$ play an important role because it describes how the quantities transform under a change of units.

In physics the so-called dimensionless numbers, i.e. those quantities that do not not change under changes of units, play a particularly important role. Since their magnitudes do not change under changes of units, one might expect that their magnitudes have physical meaning, i.e. the physics changes if a dimensionless quantity changes value. These dimensionless numbers appear as a combination of our physical quantities R_j in the following way,

$$\left[R_1^{\lambda_1} \cdots R_n^{\lambda_n}\right] = \prod_{i=1}^{m} F_i^{a_{i1}\lambda_1 + \cdots + a_{in}\lambda_n}.$$

This combination can only be dimensionless if the right hand side equals one, i.e. if

$$A\boldsymbol{\lambda} = 0.$$

This means that every element in the null space of the dimension matrix defines a dimensionless quantity. The maximum number of independent dimensionless quantities therefore equals the dimension of the null space of A. If the rank of A is r, then the dimension of its null space is $n - r$ (at this point you may want to consult Section 11.5). There is a large number of physically important dimensionless numbers. Table 1 lists just a few.

EXERCISE 17. The value of a dimensionless quantity is given by $\prod_{j=1}^{n} \rho_j^{\lambda_j}$. Make use of (9.5) and show that the value of a dimensionless quantity does not change under a change of units.

We are now in a position to state the main result of this section.

THEOREM 18. *(Buckingham's pi-theorem). Any meaningful relation* $\Phi(R_1, \ldots, R_n) = 0$ *with* $R_j \neq 0$, *is equivalent to a relation of the form* $\Psi(\pi_1, \ldots, \pi_{n-r}) = 0$, *involving a maximal set of independent dimensionless combinations.*

Dimensionless number	Typical form	Description
Bond number, Bo	$Bo = \frac{\rho g L^2}{\tau}$	Expresses the ratio between gravitational forces and surface tension.
Courant number, C	$C = \frac{u\Delta t}{\Delta x}$	Limits the time step, solving advection equations numerically.
Froude number, F	$F = \frac{v}{\sqrt{gh}}$	Describes the resistance of floating bodies.
Mach number, M	$M = \frac{v_0}{v_s}$	Relative speed of an object in a medium to the speed of sound that medium.
Prandl number, P_r	$P_r = \frac{\nu}{\alpha}$	Ration of viscosity to thermal diffusivity.
Rayleigh number, R_a		Describes heat transfer in fluids.
Reynolds number, R_e	$R_e = \frac{v_s L}{\nu}$	The ratio of inertial and viscous forces in fluids.
Richardson number, R_i	$R_i = \frac{gh}{v^2}$	Ratio of potential and kinetic energies.

TABLE 1. Dimensionless numbers.

From our previous discussion this theorem should be intuitively true. The only remaining issue is what we mean by 'meaningful relation'. In short this means that the relation needs to change according to a law similar to (9.5) if we change units. For a more detailed discussion, see Hanche-Olsen (2004).

The power of Buckingham's theorem lies in the fact that the number of physical quantities are reduced by r, where r is the rank of the dimension matrix (9.6).

9.3. Simple Examples.

In this section we apply Buckingham's theorem to a few simple examples as a warm-up to the more interesting examples of the remainder of this chapter.

EXAMPLE. Let us see how one can get an estimate of how long it takes to travel from point A to point B. The physical quantities are, distance d between A and B, the time T, and the average speed, v. The basic units are m and s. According to Buckingham's Theorem (Theorem 18), there is $3 - 2 = 1$ dimensionless quantity.

Since the units of the physical quantities are $[d] = m$, $[T] = t$ and $[v] = ms^{-1}$ the dimension matrix becomes

$$A = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & -1 \end{bmatrix}.$$

The one dimensional null space is spanned by $\begin{bmatrix} -1 & 1 & 1 \end{bmatrix}$ which means means that the dimensionless number is given by

$$\pi = d^{-1}Tv.$$

This can be rewritten as

$$T = \pi d/v$$

giving us the time in terms of the average speed and the distance between the two points. Of course we don't know the value of the dimensionless number π but at least we have reduced to problem to a simple experiment to determine π.

We now decide that the time may also depend on the acceleration a. Let us see how that affects our calculations. It may be convenient to become a little more systematic and order the physical quantities and the units in a table,

Physical quantities	d	T	v	a
Units	m	s	ms^{-1}	ms^{-2}

TABLE 2. Physical quantities and their units.

The dimension matrix therefore becomes

$$A = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & -1 & -2 \end{bmatrix}.$$

In this case we have four physical quantities and two units, Buckingham's theorem therefore gives us two dimensionless numbers that get from calculating the two-dimensional null space of A. Since it is in echelon form, one easily finds the following basis elements, $\begin{bmatrix} -1 & 1 & 1 & 0 \end{bmatrix}$ and $\begin{bmatrix} -1 & 2 & 0 & 1 \end{bmatrix}$. The two dimensionless quantities therefore are

$$\begin{aligned} \pi_1 &= d^{-1}Tv \\ \pi_2 &= d^{-1}T^2a. \end{aligned}$$

Also according to Buckingham's theorem, the physical equation relating all the variables now takes the form, $\Psi(\pi_1, \pi_2) = 0$. This is not uniquely defined and in order to proceed one has to rely on physical intuition. When a is zero we are back at the previous case and we know that $\Psi(\pi_1, 0) = \pi_1 - 1$. At least for small a, it is not unreasonable to assume an additive correction, i.e. to assume that $\Psi(\pi_1, \pi_2) = \kappa_1\pi_1 + \kappa_2\pi_2 - 1 = 0$, where κ_1 and κ_2 are dimensionless constants providing the necessary generality. Thus we end up with

$$d = \kappa_1 vT + \kappa_2 aT^2$$

and it remains to determine the two dimensionless numbers κ_1 and κ_2.

EXAMPLE. In this example we want to calculate the period of a simple linear oscillator swinging in a gravitational field. Let us first set up our physical quantities. We expect that the period T depends on the mass M, the length of the pendulum L, and gravitational acceleration g. The basic units are s, kg and m. Setting up the table of quantities and units we get,

Physical quantities	T	M	L	g
Units	s	kg	m	ms^{-2}

TABLE 3. Physical quantities and their units.

, and the dimension matrix becomes,

$$A = \begin{bmatrix} 1 & 0 & 0 & -2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}.$$

The null space is one dimensional and is spanned by $\begin{bmatrix} 2 & 0 & -1 & 1 \end{bmatrix}$, and the only dimensionless quantity is given by

$$\pi = T^2 L^{-1} g.$$

The model therefore becomes $\Psi(\pi) = 0$. We cannot get much further than this. Assuming however that Ψ has a single zero, denoted by π, we get that

$$T = \kappa\sqrt{L/g}$$

where κ is a dimensionless constant that has to be determined. Interestingly we find that the period does not depend on the mass of the particle.

EXAMPLE. Let us now consider a related problem, but his time we want to calculate the period T of a particle of mass M that is attached to a linear spring with modulus k. Since, from the previous example we might expect the period to depend on gravitational acceleration as well, we set up the following table,

Physical quantities	T	M	k	g
Units	s	kg	$kg\,s^{-2}$	ms^{-2}

TABLE 4. Physical quantities and their units.

with corresponding dimension matrix

$$A = \begin{bmatrix} 1 & 0 & -2 & -2 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Since the one dimensional null space is spanned by $\begin{bmatrix} 2 & -1 & 1 & 0 \end{bmatrix}$ the dimensionless quantity is

$$\pi = T^2 M^{-1} k.$$

Reasoning as before we get the following expression for the period

$$T = \kappa\sqrt{M/k},$$

where again we need to determine the value of the dimensionless number κ. Note that the period does not depend on gravity—the period of the oscillator is the same on the moon and on earth.

EXAMPLE. For our final example we return to water waves, discussed in the previous chapter. As we saw, surface waves are characterized by a wave number $k = 2\pi/\lambda$ where $\lambda$is the wave length, and an angular frequency ω. The dispersion relation expresses the frequency in terms of the wave number, and our goal is to derive it from the general considerations of this chapter. Our first goal is to list the physical quantities that are involved. The depth d of the water should play a role, as

well as the height h of the wave, and gravitational acceleration g. The fluid properties that play a role are the density ρ, and the surface tension τ, for very small waves. We therefore set up our customary table

Physical quantities	ω	k	h	d	ρ	τ	g
Units	s^{-1}	m^{-1}	m	m	$kg\,m^{-3}$	$kg\,s^{-2}$	ms^{-2}

TABLE 5. Physical quantities and their units.

from which we obtain the dimension matrix

$$A = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & -2 & -2 \\ 0 & -1 & 1 & 1 & -3 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}.$$

In this case we expect to find four dimensionless constants, and indeed the null space of A is spanned by the following

$$\begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} -2 \\ 3 \\ 0 \\ 0 \\ -1 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} -2 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}.$$

The dimensionless numbers are therefore,

$$\pi_1 = kh,\ \pi_2 = kd,\ \pi_3 = \frac{k^3\tau}{\omega^2\rho},\ \pi_4 = \frac{gk}{\omega^2}.$$

Rearranging for convenience, we end up with the following,

$$\pi_1 = kh,\ \pi_2 = kd,\ \pi_3 = \frac{\rho g}{\tau k^2},\ \pi_4 = \frac{\omega^2}{gk}.$$

Note that π_3 is written in this way so that it becomes the Bond number, relating gravitational force to surface tension. Since there is now only one term containing the frequency, the relationship involving all these numbers can be written as

$$\omega^2 = gk\Psi(\pi_1, \pi_2, \pi_3). \tag{9.1}$$

In order to simplify further we need to take more physics into account. If the waves are long, i.e. $k << 1$, then the Bond number π_3 is large and one can ignore the surface tension. If in addition, the water is deep with respect to the wave length, then $dk \to \infty$, and if the wave height is small as compared to the wave length, $kh \approx 0$. If all these approximations hold, then Ψ is approximately constant and we find that ω^2 is proportional to gk, and in fact we know from the previous chapter that in this limit

$$\omega^2 = gk.$$

There is one more result that one can derive from this. For very short waves, it seems natural to assume that gravity plays no role and that only surface tension is responsible for the wave motion. In order for gravity to cancel from (9.1) it is necessary for Ψ to be inversely proportional to $\pi_{3.}$. Still assuming that $dk >> 1$ and $kh << 1$, we end up with a dispersion relation of the form

$$\omega^2 = \frac{\tau k^3}{\rho}.$$

The dimensionless constant that is ignored in this expression is in fact equal to 1.

9.4. Shock Waves.

The first atomic bomb was exploded on 16 July 1945 on the barren planes of the Alamogordo Bombing Range, some 210 miles south of Los Alamos where it was developed. One question very much on the minds of the scientists involved was the yield of the gadget, i.e. how much energy it would release. Estimates varied sharply, from the low estimate of 3 000 ton TNT by Oppenheimer to the 30 000 ton TNT of Edward Teller, who always liked big explosions. Waiting for the explosion that was scheduled for 5:30 am was tense with no one really knowing what to expect. The rest is history.

In this section we described the ingeneous argument of GI Taylor, leading to a relatively simple expression that we will use to estimate the yield using the photographs taken of the original explosion.

The energy release by massive explosions drive a shock front that with speed of propagation depending on the amount of energy. The full mathematical model is

truly complicated, and we only give the equations in order to impress the reader—we have no intention of solving them. Assuming that the propagation of the shock front is spherically symmetric, i.e. it only depends on the radius from the explosion center, one can write down the necessary equations: Need a figure?

(1) Conservation of mass, in spherical coordinates

$$\partial_t \rho + \frac{1}{r^2}\partial_r r^2 \rho u = 0,$$

where ρ is the mass density, u the gas velocity, t the time and r the radial distance from the explosion center.

(2) Conservation of momentum

$$\partial_t u + u\partial_r u + \frac{1}{\rho}\partial_r p = 0$$

where p is the gas pressure.

(3) Conservation of energy

$$\partial_t \left(\frac{p}{\rho^\gamma}\right) + u\partial_r \left(\frac{p}{\rho^\gamma}\right) = 0,$$

where γ is the adiabatic index, a constant property of the gas; for air $\gamma = 1.4$.

These are the basic equations and should be supplemented by the boundary conditions at the shock front:

(1) Conservation of mass,

$$\rho_f(u_f - D) = -\rho_0 D$$

where $\rho_f,\ u_f$ are the density and velocity respectively just behind the shock front, $D = \dfrac{dr_f}{dt}$ is the shock speed, r_f is the shock radius, and ρ_0 is the initial gas density of ambient quiescent air.

(2) Conservation of momentum,

$$\rho_f(u_f - D)^2 + p_f = p_0 + \rho_0 D^2$$

where p_f is the gas pressure just behind the shock front and p_0 is the gas pressure of ambient quiescent air.

(3) Conservation of energy,

$$\rho_f(u_f - D)\left[\frac{\gamma}{\gamma-1}\frac{p_f}{\rho_f} + \frac{(u_f-D)^2}{2}\right] = -\rho_0 D\left[\frac{\gamma}{\gamma-1}\frac{p_0}{\rho_0} + \frac{D^2}{2}\right].$$

Initial conditions at the beginning of the shock wave propagation should also be provided,

$$\rho(r,0) = \rho_0;\ p(r,0) = p_0;\ u(r,0) = 0\ (r \geq r_0),$$
$$\rho(r,0) = \rho_i(r);\ p(r,0) = p_i(r);\ u(r,0) = u_i(r)\ (r < r_0),$$

$$4\pi \int_0^{r_0} \rho_i \left[\frac{u_i^2}{2} + \frac{1}{\gamma-1}\frac{p_i}{\rho_i}\right] r^2 dr = E.$$

Here r_0 is the initial radius of the shock wave, p_0 and ρ_0 are constants denoting the initial pressure and density respectively of ambient air. E is the energy, concentrated initially in a sphere of radius r_0. The functions $\rho_i(r),\ p_i(r)$ and $u_i(r)$ give the initial distribution of the density, pressure and velocity inside the shock, respectively.

These equations are complicated, and it is hard to draw any conclusions from them. It was this situation that GI Taylor faced in 1941 when he was studying the blast effects of intense explosions. Let us again write down the physical quantities involved in the propagation of the shock front r_f. In the formulation above they are

$$E,\ \rho_0,\ t,\ r_0,\ p_0,\ \gamma$$

where γ is a dimensionless number (the adiabatic index of a gas). Let us tabulate their units.

Physical quantities	r_f	E	ρ_0	t	r_0	p_0
Units	m	$kg\,m^2\,s^{-2}$	$kg\,m^{-3}$	s	m	$kg\,m^{-1}\,s^{-2}$

TABLE 6. Physical quantities and their units.

Here we have six physical quantities and three basic units. According to Buckingham's theorem this gives us three dimensionless quantities; and expressions that are still hard to make sense of. Taylor therefore needed to simply further. The crucial assumption is that the initial radius $r_0 = 0$. This assumption means that the energy is suddenly released from an infinitely concentrated source. In practice this is reasonable if it is assumed that the shock front r_f is much larger than r_0. Neglecting

r_0 we set up the following dimension matrix

$$A = \begin{bmatrix} 1 & 2 & -3 & 0 & -1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & -2 & 0 & 1 & -2 \end{bmatrix}$$

with null space spanned by $\begin{bmatrix} -5 & 1 & -1 & 2 & 0 \end{bmatrix}^T$ and $\begin{bmatrix} 3 & -1 & 0 & 0 & 1 \end{bmatrix}^T$. This leads to two dimensionless constants,

$$\pi_1 = r_f \left(\frac{\rho_0}{Et^2} \right)^{1/5}$$

and

$$\pi_2 = r_f^3 E^{-1} p_0.$$

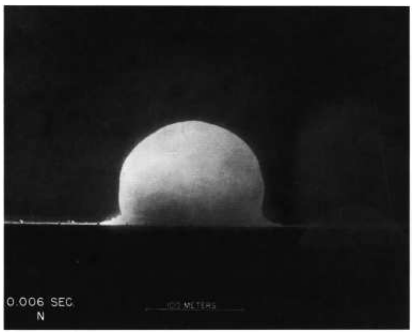
If we clear r_f from the second equation, using the first, we can write the second dimensionless constant in the form

$$\pi_2 = p_0 \left(\frac{t^6}{E^2 \rho_0^3} \right)^{1/5}.$$

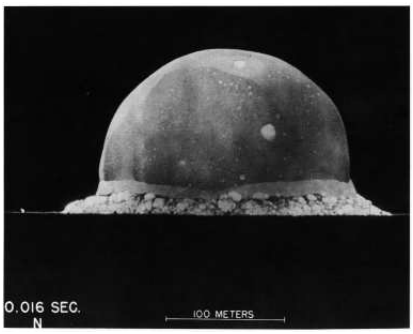
We have now simplified a seemingly intractable problem to an equation of the form, $\Psi(\pi_1, \pi_2) = 0$. Taylor then made another crucial observation, namely that if one can assumes a massive release of energy, it is possible to set $\pi_2 = 0$, or equivalently, neglect the initial pressure p_0 in the ambient gas. Thus we are left with Taylor's celebrated expression for the propagation of a shock front,

$$r_f = \kappa \left(\frac{Et^2}{\rho_0} \right)^{1/5}, \tag{9.1}$$

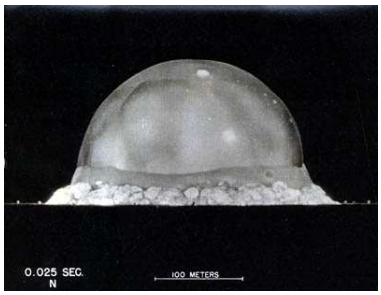
where κ is a dimensionless constant. Further analysis indicates that $\kappa \approx 1$. This has been achieved without having to resort to the problem of solving a complicated system of partial differential equation, incorporating moving boundary conditions. All one now needs to do is to study photographs of the progress of the shock front, in order to estimate the amount of energy released E. Figure 9.4.1 shows the progress of the shock front of the first nuclear explosion mentioned above. These images, together with independent measurements of the strength of the blast, confirmed the accuracy of (9.1).



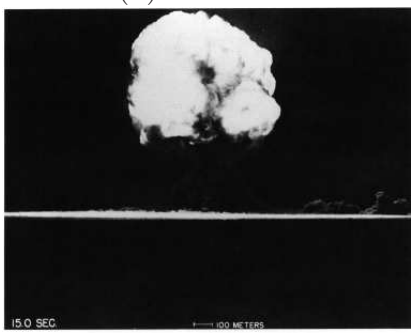
(a) After 0.006 s.



(b) After 0.016 s.



(c) After 0.025 s.



(d) After 15.0 s.

FIGURE 9.4.1. Shock wave of the first atomic bomb.

PROBLEM 19. Use the photographs to estimate the yield of the explosion. Assume that the images are calibrated so that the distance scale reflects the true distances at the site of the explosion. The average ambient density of air is about $\rho_0 = 1\text{kg m}^{-3}$ and one ton of TNT has the energy equivalent of 4184 megajoule. Compare your answer with the best official estimate of 19˜000 ton TNT. Taylor used Figure 9.4.1(c) and measured the radius of the shock front to be 140m. How does your measurement compare with his? (Actually he combined the information from all available photographs in order to reduce the measurement error.)

(Reference:http://www.answers.com/topic/nuclear-weapon-yield).

PROBLEM 20. The famous Italian physicist, Enrico Fermi, was also present at the test, situated in an observation post 10 000 yards from the explosion. Not prepared to wait for the official estimates obtained from direct pressure measurements, he

devised his own experiment. In his own words (R Rhodes. The making of the atomic bomb. Simon and Schuster, New York (1986), p674.),

> About 40 seconds after the explosion the air blast reached me. I tried to estimate its strength by dropping from about six feet small pieces of paper before, during, and after the passage of the blast wave. Since, at the time, there was no wind, I could observe very distinctly and actually measure the displacement of the pieces of paper that were in the process of falling while the blast was passing. The shift was about $2\frac{1}{2}$ meters, which at the time, I estimated to correspond to the blast that would be produced by ten thousand tons of TNT.

It is not unreasonable to assume that Fermi was aware of the Taylor formula. Why did he not use it? Try it! What is the reason for the disappointing estimate? Does this explain why Fermi designed his own experiment?

9.5. Dimensionless Numbers.

Since the dimensionless numbers do not depend on the units in which physical quantities are expressed they often convey universal truths. In this section we explore the truths captured by some of the numbers.

We start with a very simple example and investigate the distance an object might travel in the presence of a gravitation. Let us list the physical quantities that might be involved: distance (height) traveled h, time T, gravitational acceleration g, take-off speed v, and the mass m of the object. Table 7 lists the physical quantities and their units.

Physical quantities	h	T	v	g	m
Units	m	s	ms^{-1}	ms^{-2}	kg

TABLE 7. Physical quantities and their units.

The dimension matrix, with units $m,\ s$, and kg, is given by

$$A = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & -1 & -2 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

Calculating the $3D$ null space shows that the height is independent of the mass (if our physical intuition were stronger we could have omitted it right from the beginning), and the two dimensionless quantities are given by

$$\pi_1 = \frac{Tv}{h},\ \pi_2 = \frac{g}{hT^2}.$$

Solving for T gives us the dimensionless quantity we are after, namely the Froude number, or rather, the square of the Froude number,

$$Fr = \frac{v^2}{hg}. \tag{9.1}$$

The important thing to notice is that it relates the kinetic– and potential energies of a particle moving in a gravitational field. Once a measurement has revealed that $Fr = 2$ for a particle launched in a gravitational field we can confidently use (9.1) to calculate the height obtained by a particle if it is launched with an initial (vertical) velocity v, without knowing any more about the physics.

Noting that the Froude number relates the conversion of kinetic– into potential energy, and back, one might be led to investigate whether it is informative in other situations where there is a transfer between kinetic– and potential energies.

Water waves. In the case of surface waves, gravitation would like to flatten the waves while it kinetic energy is what keeps it moving. The Froude number therefore tells us that waves with higher wavelengths h move faster than waves with shorter wavelengths. Experiments reveal the magic wave number to be about $Fr = 0.16$.

Surface vessels. Let us now consider an ordinary vessel with a water replacing hull. These vessels generate a bow wave and additional waves along its length and its stern. Figure clearly shows the bow wave of a typical surface vessel.

This wave has a characteristic wavelength of about the length of the vessel. As long as the duck of Figure 9.5.1 moves at the characteristic speed of the wave its

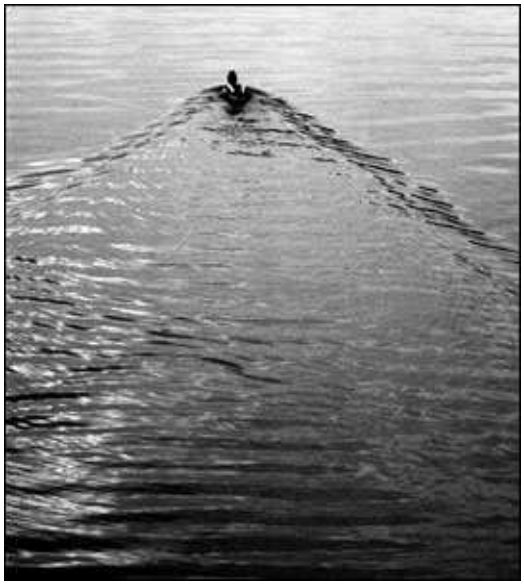


FIGURE 9.5.1. Bow wave created by a duck.

creates, it is quite efficient. Should it however try and swim faster, it has to overtake the bow wave in front of it, and that takes a large amount of energy. Ship designers know this too, and if you design a boat for racing it should take the familiar form shown in Figure 9.5.2. Boat design is largely about bringing the Froude number down as much as possible.

Animals are short by nature and therefore quite inefficient surface vessels. They do however, have a few clever design principles up their sleeves. Penguins are remarkable. Unable to fly, clumsy on land, they have evolved into very efficient sea creatures. When efficiency is important they swim submerged. This enables them to neatly circumvent the limitations placed by the Froude number, see Figure 9.5.3.

Instead of dealing with the physical limitations of surface swimming, Emperor penguins shown in Figure 9.5.3 have evolved amazing lung capacities and can stay submerged for up to 18 minutes, reaching depths of over 200m.

How Dinosaurs ran. No one has seen a dinosaur, much less one in action. We return to the question stated in the Introduction to this chapter. How can one tell how fast dinosaurs moved? R. McNeill Alexander studied the movement of modern animals in detail and came to the conclusion that the Froude number again

FIGURE 9.5.2. K2 racing canoe.

T 

FIGURE 9.5.3. Penguins are more efficient swimming submerged.

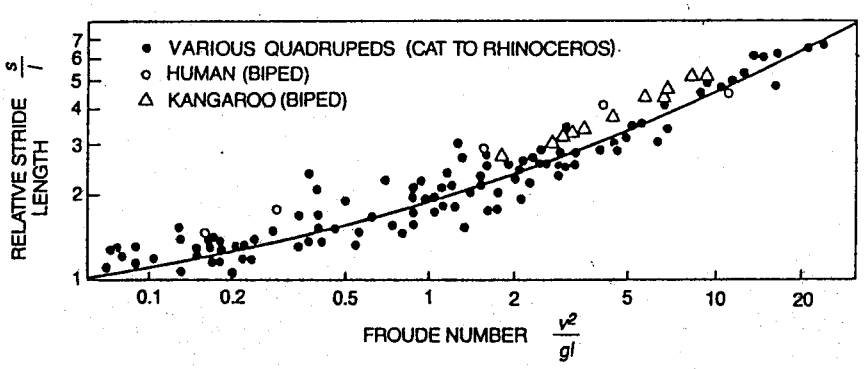


FIGURE 9.5.4. Ratio of stride— and leg length against Froude number.

determines the basic physics of animal motion. In this case the length parameter h is the length of the leg. This makes sense since movement is based on the continuous exchange of kinetic— and gravitational potential energies. The elastic energy that is stored in the tendons, ligaments and muscles that makes hopping such an efficient means of movement, is assumed to be similar for the animals under consideration.

The first observation is that a specific Froude number determines the gait of the animal; different animals change from walking to trotting at about the same Froude number of 2.55. One can also plot the relative stride length to leg length against the Froude number for a number of animals, including humans, bipeds such as Kangaroos, and various quadrupeds. They all fall roughly on the same curve as shown in reproduction from (Alexander) in Figure 9.5.4.

It so happens that dinosaurs left fossilized footprints in a number of places see Figure 9.5.5. How does this help us to measure their speed? Since leg length can be estimated from the size of the footprint—a footprint should be about one quarter of the leg length—and the stride length can be measured, the appropriate Froude number can be read from the graph. And from there the walking speed of the dinosaur can be estimated. It is estimated that the large sauropods walked at about $1m/s$, painfully slow for animals with three meter long hind legs. You should have no problem staying out of their way.

Bond number. We met the Bond number earlier when we derived the dispersion relation for water waves. It describes the ration of the gravitational force to surface

FIGURE 9.5.5. Fossilized footprints.

tension force and can be written as

$$Bo = \frac{mg}{\gamma l}, \tag{9.2}$$

where γ is the surface tension and l is a length. Suppose you need to walk on water supported by surface tension in which case l is the length of the leg-water contact. In order to be supported by surface tension we need $Bo < 1$. It should be clear that you need to be either extremely light, or the contact length l should be very long. If you are a water strider weighing a tenth of a gram, your legs touching the water need not be extraordinary long, 1.3 mm should do, as shown in Figure 9.5.6. Note the relatively long legs and the large contact region.

FIGURE 9.5.6. Surface tension supporting a water strider.

EXERCISE 21. Figure out how long your feet need to be in order to walk on water. Surface tension depends on the water temperature but you can use a value of $72 \times 10^{-3} Nm^{-1}$. Surface tension increases a little for colder water, but not enough to make a real difference. We are of course not talking about very low temperatures as in ice, where the mechanism is completely different.

CHAPTER 10

Asymptotics

10.1. Introduction.

The governing equations for various phenomena in nature and in the sciences often take the form of ODEs, PDEs, integral equations, linear or nonlinear systems of equations, or combinations of the above. The three most important methods to gain information from such equations roughly fall in the categories of analytical, numerical, and asymptotic methods.

Analytical (exact) methods: For all but the very simplest model problems (the importance of which we however do not want to under estimate), this avenue is rarely possible. A vast number of seemingly innocent looking integrals, ODEs, algebraic equations, etc., just do not have solutions that can be expressed in a finite number of terms of our standard elementary functions. Most variable coefficient second order ODEs fall in this category (even if they are linear), as does finding the roots of a polynomial equation of degree 5, or finding a function whose derivative is e^x/x, etc. Although computer algebra systems (such as Mathematica and Maple) are incredibly competent in advanced calculus and can flawlessly carry out very lengthy algebraic tasks, they still can not come up with closed form solutions when these just do not exist.

Numerical methods: As a general rule, if a set of equations is *well posed*, numerical methods will in theory be able to provide good approximations. However, numerical (or graphical) outputs may provide much less insights than either exact or approximate formulas. There are also many situations of interest when numerics gets increasingly difficult and its cost becomes prohibitive. Examples include when integrands get extremely peaked or extremely oscillatory and, more generally, when

problems includes phenomena on vastly different space or time scales (such as large-scale flows with fine scale turbulence present). Any attempt to resolve the fine scale by 'brute force numerics' becomes hopelessly expensive on the large scales.

Asymptotic methods: This is the topic of the present chapter. It provides analytic approximations that often get increasingly more accurate in situations similar to that where numerics gets increasingly difficult. In asymptotic methods, an unsolvable mathematical task is replaced by a sequence of solvable ones. Typically, a rough approximation is first derived by hand, and symbolic algebra can then be used very effectively to generate expansions with many terms, increasing the accuracy of the approximation. This approach often complements regular numerical methods, being particularly strong when the other is particularly weak, and vice versa. Although the two topics naturally belong together, modern texts very rarely touch on both. Among many excellent texts on asymptotic methods, we want in particular to mention [**9**].

This chapter is only intended to give a very brief flavor of the rich topic of asymptotic methods, which contains far more techniques and application areas than we can possibly touch on here. Significant omissions include singular perturbations and boundary layer methods (essential to fluid mechanics), WBKBJ methods (which play a large role for example in quantum mechanics) and multiple scale analysis. Before turning to actual asymptotic methods, let us consider two examples.

10.1.1. Example 1: The van der Pol oscillator. It is described by the ODE

$$\frac{d^2y}{dt^2} + k\,(y^2-1)\frac{dy}{dt} + y = 0. \tag{10.1}$$

When $k = 0$, the general solution $y(t) = c_1 \cos t + c_2 \sin t$ has the period $P(k = 0) = 2\pi$, regardless of the choice of initial conditions. When $k > 0$, the ODE becomes nonlinear and can no longer be solved in closed form. It features negative damping when $|y| < 1$ and positive damping when $|y| > 1$. To be more precise, if we define the *energy* as $E = \frac{1}{2}\,(y^2 + \dot{y}^2)$ then it follows readily that y

$$\frac{dE}{dt} = -k(y^2-1)\dot{y}^2.$$

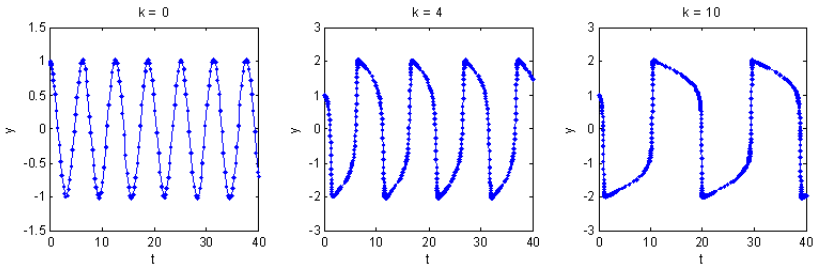


FIGURE 10.1.1. Solutions of the van der Pol oscillator for three different k-values (when starting with $y(0) = 1$, $y'(0) = 0$).

Thus the energy steadily increases if $|y| < 1$, and steadily decreases if $|y| < 1$. This leads to solutions as seen in Figure 10.1.1.

For any non-zero initial condition, the solutions will settle into patterns for which the period $P(k)$ depends only on k, giving the solid curve in Figure 10.1.2. Clearly, $P(k)$ seems to increase with k, but just how fast when $k \to \infty$? Numerical solutions then become difficult. Matlab's particularly robust ODE solver ode15s fails already for $k = 15$ because of the sharp spikes in $\frac{dy}{dt}$. With asymptotic analysis, one can however quite readily prove a result such as

$$P(k) \approx \begin{cases} 2\pi(1 + \frac{1}{16}k^2 + \ldots) & \text{as } k \to 0 \\ 1.6137\, k + 7.0143\, k^{-1/3} + \ldots & \text{as } k \to \infty \end{cases}, \tag{10.2}$$

where the two constants are $3 - \log 2 \approx 1.6137$ and 7.0143 is three times the first root of the Bessel equation $J_{1/3}(x) + J_{-1/3}(x) = 0$. These two 'leading order' estimates (10.2) are also shown in Figure 10.1.2.

More terms can be worked out with symbolic algebra, providing still better approximations.

10.1.2. Example 2: Radial basis function coefficients on an infinite unit-spaced grid in 1-D. Radial basis functions (RBFs) are discussed in Chapter 14. Their main use is for interpolation and for solving PDEs over scattered node sets on bounded domains in two or more dimensions, combining excellent accuracy with the ability to carry out local node refinement wherever the solutions need to be particularly well resolved. However, analysis in much simpler settings can sometimes give valuable insights. One such case is finding the RBF expansion coefficients

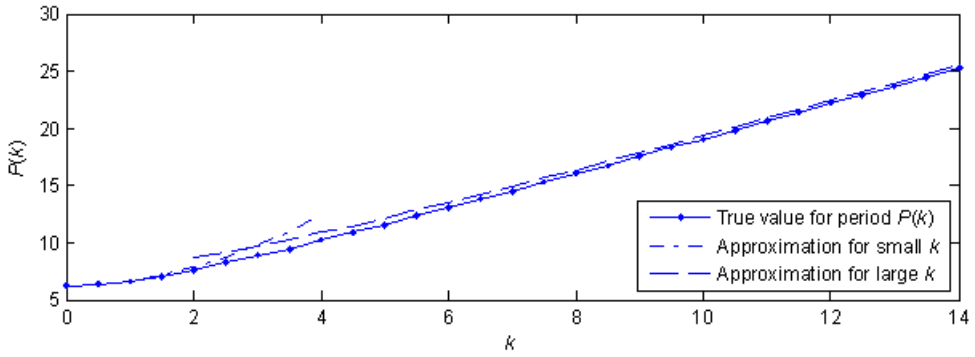


FIGURE 10.1.2. True value for the period $P(k)$ compared to the leading order approximations for $k \to 0$ and $k \to \infty$.

λ_k, $k \in Z$ (the set of all integers from $-\infty$ to $+\infty$) so that the RBF expansion

$$s(x) = \sum_{k=-\infty}^{\infty} \lambda_k \phi(|x - x_k|) \tag{10.3}$$

with $x_k = k$ reproduces *cardinal data*

$$s(x_k) = \begin{cases} 1 & k = 0 \\ 0 & k \neq 0 \end{cases} ,$$

for integer k. Such *cardinal interpolants* $s(x)$ play a similar role as the individual terms in Lagrange's interpolation formula (Section 12.2). As shown in **[6]** (we skip the details here, but the tools are those of Fourier transforms, cf. Section 7.3), one can find 'closed form' expressions for the λ_k coefficients. For example, in the case of multiquadric RBFs ($\phi(r) = \sqrt{1 + (\varepsilon r)^2}$) in the special case of $\varepsilon = 1$, one obtains

$$\lambda_k = -\frac{1}{4\pi} \int_0^{2\pi} \frac{e^{ik\xi}}{\sum_{j=0}^{\infty} \frac{K_1(2\pi j + \xi)}{2\pi j + \xi} + \sum_{j=1}^{\infty} \frac{K_1(2\pi j - \xi)}{2\pi j - \xi}} d\xi \ , \tag{10.4}$$

where K_1 stands for the first order K-Bessel function. Technically, this is called a closed form solution since it only involves sums and an integral, but it is very unpractical to work with, and it does not offer good insights. However, the method of *steepest descent* (Section 10.5.3) shows that there are two main components to the values for λ_k, and that each of these can be approximated as a rapidly converging

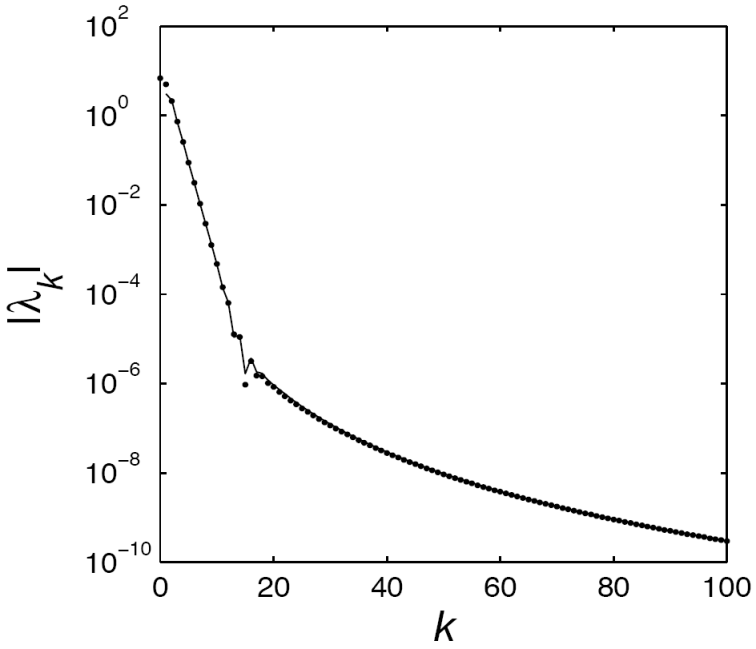


FIGURE 10.1.3. The magnitude of the RBF coefficients λ_k plotted against k. Dots: Exact values, Piecewise linear curve: Connecting the values for $k = 1, 2, \ldots$ that are given by the two-term approximation (10.5).

series. Keeping only the first term in each gives

$$\lambda_k \approx \underbrace{(-1)^{k+1} 17.3\, e^{-1.04k} + \ldots}_{\text{exponential part}} \quad \underbrace{-\frac{3}{k^5} + \ldots}_{\text{algebraic part}} \quad . \tag{10.5}$$

This is a significant simplification of (10.4)—and is also extraordinary accurate, as seen in Figure 10.1.3.

10.2. Algebraic Equations.

Algebraic equations can be used to illustrate some approaches for generating approximate solutions of increasing accuracies. The task we set ourselves is to explore how the roots of an algebraic (polynomial) equation change when a coefficient undergoes a small change. To be specific, let us investigate how the root near $x = 1$

for

$$x^2 + \varepsilon x - 1 = 0 \ , \tag{10.1}$$

varies as a function of ε where ε is assumed to be small, $|\varepsilon| << 1$. We consider three approaches of increasing generality. With the last of the approaches, we arrived at a widely applicable one.

10.2.1. Analytic solution. As it happens, this equation can be solved in closed form; $x = -\frac{\varepsilon}{2} \pm \sqrt{1 + (\frac{\varepsilon}{2})^2}$. Using the binomial expansion

$$(1+z)^\alpha = 1 + \alpha z + \frac{\alpha(\alpha-1)}{1\cdot 2} z^2 + \frac{\alpha(\alpha-1)(\alpha-2)}{1\cdot 2\cdot 3} z^3 + \dots \ , \tag{10.2}$$

one immediately finds that the root near $x = 1$ is given by

$$x = 1 - \frac{1}{2}\varepsilon + \frac{1}{8}\varepsilon^2 - \frac{1}{128}\varepsilon^4 + \frac{1}{1024}\varepsilon^6 - + \dots \tag{10.3}$$

However, the whole point with perturbation expansions is to clarify cases without closed form solutions. Let us therefore look at two possibilities that do not use the analytic result for the root.

10.2.2. Direct iteration.

We can rewrite (10.1) as $x^2 = 1 - \varepsilon x$, i.e. $x = \sqrt{1 - \varepsilon x}$. It is now tempting to set $x_0 = 1$ and then iterate

$$x_{n+1} = \sqrt{1 - \varepsilon x_n}, \ \ n = 0, 1, 2, \dots , \tag{10.4}$$

just as we do in Section 13.2, where we call it Fixed Point Iteration. Mathematica makes the algebra simple. The statement

`n` $= 6$; `x` $= 1$; `Do[x = Series[`$\sqrt{1 - \epsilon\ \texttt{x}}$, {$\epsilon$, 0,`n`}, `Print[x],{k,1,n}]`

produces the successive results

$1 - \frac{\epsilon}{2} - \frac{\epsilon^2}{8} - \frac{\epsilon^3}{16} - \frac{5\epsilon^4}{128} - \frac{7\epsilon^5}{256} - \frac{21\epsilon^6}{1024} + O[\epsilon]^7$
$1 - \frac{\epsilon}{2} + \frac{\epsilon^2}{8} + \frac{\epsilon^3}{8} + \frac{11\epsilon^4}{128} + \frac{3\epsilon^5}{64} + \frac{19\epsilon^6}{1024} + O[\epsilon]^7$
$1 - \frac{\epsilon}{2} + \frac{\epsilon^2}{8} + 0\epsilon^3 - \frac{9\epsilon^4}{128} - \frac{5\epsilon^5}{64} - \frac{55\epsilon^6}{1024} + O[\epsilon]^7$
$1 - \frac{\epsilon}{2} + \frac{\epsilon^2}{8} + 0\epsilon^3 - \frac{\epsilon^4}{128} + \frac{\epsilon^5}{32} + \frac{57\epsilon^6}{1024} + O[\epsilon]^7$
$1 - \frac{\epsilon}{2} + \frac{\epsilon^2}{8} + 0\epsilon^3 - \frac{\epsilon^4}{128} + 0\epsilon^5 - \frac{15\epsilon^6}{1024} + O[\epsilon]^7$

$1 - \frac{\epsilon}{2} + \frac{\epsilon^2}{8} + 0\epsilon^3 - \frac{\epsilon^4}{128} + 0\epsilon^5 + \frac{\epsilon^6}{1024} + O[\epsilon]^7$

(where we have included also terms with zero coefficients; Mathematica would typically not print these). With every iteration, one more coefficient becomes correct. Having observed that, there is no need to carry the initial expansions any further than to only include terms we know are correct. The Mathematica statement can therefore be simplified to

$\texttt{n} = 6;\ \texttt{x} = 1;\ \texttt{Do[x = Series[}\sqrt{1 - \epsilon\ \texttt{x}},\ \{\epsilon, 0, \texttt{k}\}, \texttt{Print[x],\{k,1,n\}]}$

Although we do not use the analytic solution for the root, this method is also unsatisfactory in that we utilize a simple way to write down the plausible iteration (10.4). For higher order equations, with more terms present and ε maybe entering in more complex ways, that is unlikely to work. Hence, we need a still more general idea.

10.2.3. Equating coefficients. The idea is to look for coefficients in an expansion

$$x = 1 + a_1\varepsilon + a_2\varepsilon^2 + a_3\varepsilon^3 + \ldots + a_n\varepsilon^n + O(\varepsilon^n)$$

such that, when substituting this into the left hand side of (10.1), the coefficients for all powers of ε up through ε^n all vanish. The sequence of relations we obtain at the different orders of ε, becomes

$$\begin{aligned} \varepsilon: &\quad 1 + 2a_1 = 0 \\ \varepsilon^2: &\quad a_1 + a_1^2 + 2a_2 = 0 \\ \varepsilon^3: &\quad a_2 + 2a_1a_2 + 2a_3 = 0 \\ &\text{etc.} \end{aligned}$$

The essential point that emerges is that we can solve these equations in turn, at every step obtaining a new coefficient in terms of already calculated ones: $a_1 = -\frac{1}{2},\ a_2 = -\frac{1}{2}\left(a_1 + a_1^2\right) = \frac{1}{8},\ a_3 = 0, \ldots$

Mathematica again allows this to be automated. If we want to display the results for all the steps along the way, we execute in turn the lines

$\texttt{n} = 6;\ \texttt{x} = 1 + \sum_{\texttt{i}=1}^{\texttt{n}} \texttt{a}_\texttt{i}\ \epsilon^{\texttt{i}} + \ \texttt{O[}\epsilon\texttt{]}^{\texttt{n+1}}$

```
x^2 + ϵ x - 1 = = 0
LogicalExpand[%]
Solve[%]
```

with the last statement producing the output

$$\left\{\left\{a_6 \to \tfrac{1}{1024}, a_5 \to 0, a_4 \to -\tfrac{1}{128}, a_3 \to 0, a_2 \to \tfrac{1}{8}, a_1 \to -\tfrac{1}{2}, \right\}\right\}$$

It can quite frequently happen (and most certainly will in the case of roots with multiplicities higher than one) that the resulting expansion also includes fractional powers of ε. Insightful 'trial and error' is an invaluable tool in this field!

10.3. Convergent vs. Asymptotic expansions

The series expansion obtained in the previous section converges whenever $|\varepsilon| < 2$. It turns out that one often encounters expansions that are nowhere convergent. If our first instinct would be to reject all divergent expansions as nonsense, we would be in excellent company. Niels Henrik Abel (1802-1829; arguably the greatest ever Scandinavian mathematician) is quoted as saying "The divergent series are the invention of the Devil, and it is a shame to base on them any demonstration whatsoever". However, divergent expansions can be tremendously effective in many cases. We next give two illustrations of this. After that some discussion is needed in order to (i) establish when divergent expansions can and cannot be used, and (ii) show how to 'accelerate' them in order to obtain as much valuable information as possible from only a few leading terms.

10.3.1. Two examples of divergent expansions.

10.3.1.1. *The error function.* Most scientific computer systems have the error function[**15**]

$$\mathrm{Erf}(z) = \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2} dt \tag{10.1}$$

as a standard built-in function (although in case of Matlab at present limited to real values of z). Suppose we don't have the function available, and need to approximate it at $z = 6$. Let us investigate two possible approaches.

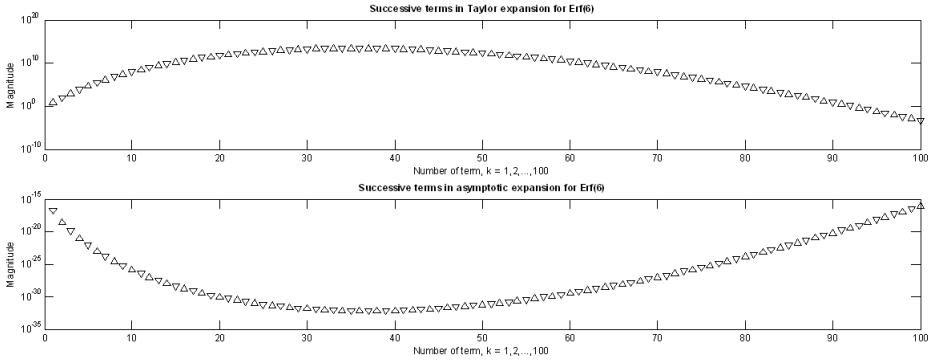


FIGURE 10.3.1. The sizes of the successive terms in the Taylor (top) and the asymptotics (bottom) expansions for evaluating Erf(6). In both cases, $\triangle$ indicates a positive term and $\triangledown$ a negative term.

Taylor expansion. From the Taylor expansion of e^t follows,

$$\begin{aligned} e^t \quad &= 1 + \frac{1}{1!}t + \frac{1}{2!}t^2 + \frac{1}{3!}t^3 + \dots \\ \Rightarrow \qquad e^{-t^2} \quad &= 1 - \frac{1}{1!}t^2 + \frac{1}{2!}t^4 - \frac{1}{3!}t^6 + - \dots \\ \Rightarrow \frac{2}{\sqrt{\pi}}\int_0^z e^{-t^2}dt &= \frac{2}{\sqrt{\pi}}\left(z - \frac{1}{3\cdot 1!}z^3 + \frac{1}{5\cdot 2!}z^5 - \frac{1}{7\cdot 3!}z^7 + - \dots\right). \end{aligned} \tag{10.2}$$

The radius of convergence for all three expansions is $R_c = \infty$ and a first inclination might be to use (10.2) for numerically evaluating Erf(6).

The sizes of the successive terms in (10.2) are shown in the top part of Figure 10.3.1. Although $z = 6$ still is quite close to the origin, the result is nevertheless a complete disaster. Since the series is oscillating, the error wherever we truncate it, is less than the first omitted term. To get the terms to become of size 10^{-16}, we need to go well past 100 terms. Even if we do that, the numerically result is totally destroyed by earlier terms in the expansion that reached sizes around 10^{+16}. Cancellations between these massive terms would (in standard double precision) leave an error of size $O(1)$. Thus even after a lot of work we get an answer that is unlikely to have even a single correct digit.

Asymptotic expansion. Since $\int_0^\infty e^{-t^2}dt = \sqrt{\pi}/2$, we can write $\mathrm{Erf}(z)$ as

$$\mathrm{Erf}(z) = 1 - \frac{2}{\sqrt{\pi}}\int_z^\infty e^{-t^2}dt. \tag{10.3}$$

This looks generally promising (if z is positive), since the integrand then becomes quite small, due to the rapid decay of the Gaussian e^{-t^2}. Furthermore, we can integrate by parts as many times as we like,

$$\int_z^\infty e^{-t^2}dt = \frac{e^{-z^2}}{2z} - \int_z^\infty \frac{e^{-t^2}}{2t^2}dt,$$
$$\int_z^\infty \frac{e^{-t^2}}{2t^2}dt = \frac{e^{-z^2}}{4z^3} - \int_z^\infty \frac{3e^{-t^2}}{4t^4}dt,$$
$$\text{etc.}$$

giving

$$\text{Erf}(z) = 1 - \frac{e^{-z^2}}{z\sqrt{\pi}}\left(1 - \frac{1}{2z^2} + \frac{1\cdot 3}{(2z^2)^2} - \frac{1\cdot 3\cdot 5}{(2z^2)^3} + - \dots\right), \tag{10.4}$$

again with the property that the error is always smaller than the first omitted term (as can be seen from the derivation process which, after each integration by parts, gives an integral with known sign as exact remainder). The size of the expansion terms for $z = 6$ now evolves as seen in the bottom part of Figure 10.3.1. After only one term, the error is below 10^{-16}, and it falls below 10^{-30} after about 30 terms. However, if we keep going further, the error eventually grows without bound. Viewed as an infinite series, (10.4) is always divergent (totally in contrast to (10.2) which always converges). Nevertheless, it is the latter expansion that is superior whenever z is not very close to zero.

We considered only positive z-values above. In the complex z-plane, the Erf function grows to extremely large values within the two sectors $\frac{\pi}{4} < \theta < \frac{3\pi}{4}$ and $-\frac{\pi}{4} > \theta > -\frac{3\pi}{4}$ (measuring the angle θ from the real axis), as shown in Figure 10.3.2. Without getting into any details, we point out that truncated versions of (10.4) work everywhere in the sector $-\frac{3\pi}{4} < \theta < +\frac{3\pi}{4}$; the better the further z is away from the origin. Swapping the leading "1" to "-1" gives an expansion valid outside $-\frac{\pi}{4} < \theta < \frac{\pi}{4}$. In contrast to Taylor expansions that are valid within some circle in the complex plane, asymptotic expansions are typically valid within angular sectors. It is common that one needs to use different expansions in different sectors, but it can also happen that different expansions are valid within the same sectors—here

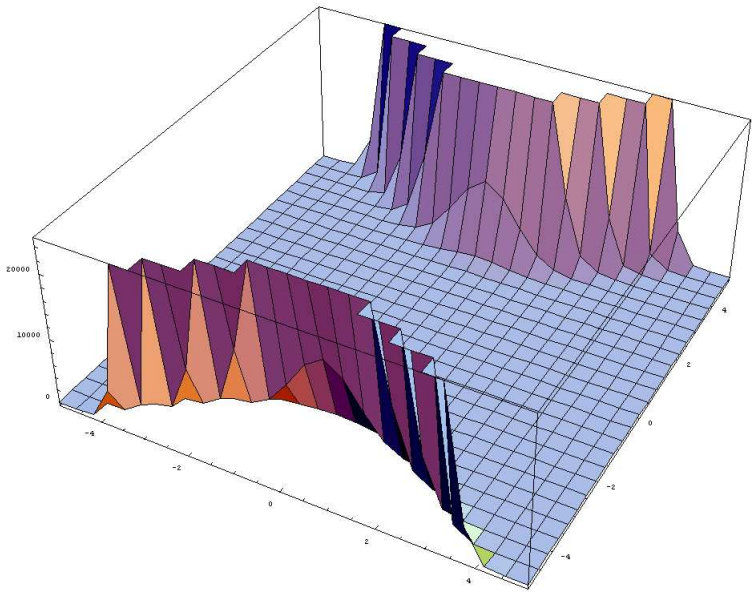


FIGURE 10.3.2. The magnitude of Erf(z) in the complex z = x + iy -plane. The real axis extends to the right (and slightly down) and the imaginary axis backwards (and slightly to the right).

illustrated by the fact that both versions of (10.4), starting with either +1 or -1, are valid in the sectors where Erf(z) 'explodes' in size.

10.3.1.2. *Euler-McLaurin's formula.* This formula is widely used for approximating slowly converging, non-oscillatory, infinite sums. There are several rigorous and very elegant derivations available in the literature. However, a mathematical modeler often needs to work heuristically in order to come up with customized formulas. To illustrate such an approach, we first recall that the trapezoidal rule approximates an integral with a sum. If we apply it to a convergent integral $\int_N^\infty f(x)dx$, use a step size $h = 1$, and noting that $f(x \longrightarrow \infty) = 0$, it suggests that $\sum_{k=N}^\infty f(k) \approx \int_N^\infty f(x)dx + \frac{1}{2}f(N)$. We may also remember that the leading error term in the trapezoidal rule contains a factor of h^2 times the first derivative at the end points. With $h = 1$, powers of h disappear, and one might guess that the remainder can be further refined by including more derivative terms,

$$(10.5)\quad \sum_{k=N}^{\infty} f(k) = \int_N^\infty f(x)dx + \alpha_0 f(N) + \alpha_1 f'(N) + \alpha_2 f''(N) + \alpha_3 f'''(N) + - \ldots,$$

where the coefficients α_n remain to be found. In Section 7.2 (on Fourier series and Fourier transforms), we make good use of the fact that any periodic function over $[-\pi, \pi]$ can be written as a linear combination of e^{ikx}, $k \in Z$, and writing functions that decay fast enough over $[-\infty, \infty]$ as combinations of $e^{i\omega x}$, $\omega \in R$ (real). One might guess that decaying functions $f(x)$ for increasing x can similarly be written as combinations of e^{-zx} where $\mathrm{Re} z > 0$. Since the coefficients need to be the same for all decaying function, we substitute $f(x) = e^{-zx}$ into (10.5) giving, after a few quick simplifications,

$$\frac{1}{1-e^{-z}} - \frac{1}{z} = \alpha_0 - \alpha_1 z + \alpha_2 z^2 - \alpha_3 z^3 + - \dots \ . \tag{10.6}$$

The unknown coefficients now follow uniquely from the Taylor expansion of the left hand side, giving

$$\sum_{k=N}^{\infty} f(k) = \int_N^{\infty} f(x)dx + \frac{1}{2}f(N) - \frac{1}{12}f'(N) + \frac{1}{720}f'''(N) - \frac{1}{30240}f^{(5)}(N) + - \dots \ . \tag{10.7}$$

The Taylor series (10.6) has a radius of convergence $R_c = 2\pi$ (the distance to the nearest singularity, located at $\pm 2\pi i$). Had R_c been infinite, the right hand side of (10.7) might well have been convergent, but in general it diverges since increasing order derivatives often grow in size like factorials. However, that does not stop (10.7) from being immensely useful as an asymptotic expansion—increasingly accurate the larger N is—provided we only use a limited number of terms.

As a test problem, consider the task of numerically evaluating Euler's constant γ, defined by

$$\gamma = \lim_{N\to\infty} \left(\left(\sum_{k=1}^{N} \frac{1}{k} \right) - \log N \right) \approx 0.57721 \dots \ . \tag{10.8}$$

Besides e and π, Euler's constant is perhaps the one that most frequently appears in unexpected situations. In contrast to e and π, it is more 'elusive', both analytically and numerically. While it has been known for centuries that e and π are irrational (not the ratio of two integers; the tools of elementary Calculus suffice for short proofs), the status of γ in this regard remains an open question (providing an opportunity for everlasting fame for an interested student!). To evaluate γ numerically, we

first rewrite (10.8) as

$$\gamma = 1 + \sum_{k=2}^{\infty} \left(\frac{1}{k} + \log(k-1) - \log(k) \right) \tag{10.9}$$

$$= 1 + \sum_{k=2}^{\infty} \left(\frac{1}{k} + \log(1 - \frac{1}{k}) \right). \tag{10.10}$$

The size of the kth term in the sum is approximately $\frac{1}{k} + \left(-\frac{1}{k} + \frac{1}{2k^2} - \frac{1}{3k^3} + \ldots \right) \approx \frac{1}{2k^2}$, and $\int_N^\infty \frac{1}{2k^2} dk = \frac{1}{N}$, so summing (10.10) directly until the error becomes less than 10^{-10} would require about 10^{10} terms. Maybe a really fast computer can do that in one second. But, say we want an error of 10^{-30}. That would then take 10^{20} seconds—definitely impractical given that astronomers estimate the time since the big bang start of the universe to be about 10^{17} seconds.

Using Euler-McLaurin's formula instead with $f(k) = \frac{1}{k} + \log\left(1 - \frac{1}{k}\right)$, we can for example sum the first 9 terms directly and then apply it with $N = 10$,

$$\begin{array}{rl}
1 + \sum_{k=2}^{9} f(k) & = +0.6317436767 \\
\int_{10}^{\infty} f(k) dk & = -0.0517553591 \\
\frac{1}{2} f(10) & = -0.0026802578 \\
-\frac{1}{12} f\,'(10) & = -0.0000925926 \\
\frac{1}{720} f\,'''(10) & = +0.0000001993 \\
-\frac{1}{30240} f^{\,(5)}(10) & = -0.0000000015 \\
------ & ------ \\
\sum_{k=1}^{\infty} f(k) & \approx +0.5772156650
\end{array}$$

The result is wrong only by one unit in the last digit. The more terms we sum directly to start with, the faster the expansion will converge. This method quickly gives hundreds or thousands of digits, should that ever be needed.

10.3.2. Some properties of asymptotic expansions. The examples above illustrate the usefulness of using only a finite number of terms of a divergent series to approximate a function. Let us now make it more precise what we mean by an asymptotic series. Accordingly consider the formal series, i.e. it may be convergent

or divergent,

$$\sum_{n=0}^{\infty} a_n \varepsilon^n$$

This series yields an asymptotic approximation of $f(\varepsilon)$, written as

$$f(\varepsilon) \sim \sum_{n=0}^{M} a_n \varepsilon^n, \tag{10.11}$$

if

$$\frac{f(\varepsilon) - \sum_{n=0}^{M} \varepsilon^n}{\varepsilon^M} \to 0 \quad \text{as} \quad \varepsilon \to 0,$$

for *any* fixed non-negative integer M. This means that even if the series diverges for fixed ε as the number of terms go to infinity, one obtains an approximation of $f(\varepsilon)$ for any *fixed* number of terms, if $\varepsilon \to 0$. It should be clear that any convergent series is also an asymptotic series. Note that there is no requirement that M need to be large, in fact useful approximations might be obtained for a small numbers of terms.

It is however, not necessary for asymptotic expansions to take on the form of a Taylor expansion around $\varepsilon = 0$ as in (10.11); they can also use other types of expansion functions. In the more general case we write

$$f(\varepsilon) \sim \sum_{n=0}^{M} a_n f_n(\varepsilon) \tag{10.12}$$

if

$$\frac{f(\varepsilon) - \sum_{n=0}^{M} a_n f_n(\varepsilon)}{f_M(\varepsilon)} \to 0 \quad \text{as} \quad \varepsilon \to 0,$$

where M is again *any* non-negative integer.

It is possible that one function can have many asymptotic expansions, for example,

$$\begin{aligned} \tan(\varepsilon) &\sim \varepsilon + \tfrac{1}{3}\varepsilon^3 + \tfrac{2}{15}\varepsilon^5 \\ &\sim \sin\varepsilon + \tfrac{1}{2}(\sin\varepsilon)^3 + \tfrac{3}{8}(\sin\varepsilon)^5 \\ &\sim \varepsilon\ \cosh(\sqrt{\tfrac{2}{3}}\varepsilon) + \tfrac{31}{270}\left(\varepsilon\ \cosh(\sqrt{\tfrac{2}{3}}\varepsilon)\right)^5. \end{aligned}$$

However, once we have decided on the expansion functions $f_n(\varepsilon)$, then the coefficients a_n are uniquely determined. If two functions have been expanded according to (10.12)

using the same expansion functions, we can add and subtract the expansions term-by-term. Multiplication and division is trickier since we may end up with different types of expansion functions, but works anyway in many cases (such as when we expand in powers of ε). If so, the result will be asymptotic in the same sense as the original functions. Term-by-term integration is always valid, but (in contrast to Taylor expansions), differentiation term-by-term may fail. It should also be noted that different functions can have the same asymptotic expansion. For example, the three functions $\frac{\tanh 1/\varepsilon}{1+\varepsilon}$, $\frac{1+e^{-1/\varepsilon}}{1+\varepsilon}$, and $\frac{1}{1+\varepsilon}$ share for $\varepsilon \longrightarrow 0+$ the asymptotic expansion $1-\varepsilon+\varepsilon^2-\varepsilon^3+-\ldots$, and are therefore described as 'asymptotically equal'.

10.3.3. Convergence acceleration. With the asymptotic expansions considered so far, it has been quite easy to generate any number of terms. In more complicated cases, it can require a lot of work to obtain just the few terms, making it important to extract as much information as possible from just a few leading terms. This is especially the case if the sequence of approximations diverges right from the start, rather then first appearing to converge and only ultimately diverges. Padé approximations and Shanks' method are two examples of powerful acceleration methods.

10.3.3.1. *Padé approximation.* The idea is to convert a truncated Taylor polynomial

$$T_{M+N}(x) = \sum_{n=0}^{M+N} a_n x^n \tag{10.13}$$

to rational form

$$P_M^N(x) = \frac{\sum_{n=0}^{N} A_n x^n}{\sum_{n=0}^{M} B_n x^n} \tag{10.14}$$

in such a way that as many derivatives as possible match at $x = 0$. After normalization through $B_0 = 1$, the expressions $T_{M+N}(x)$ and $P_M^N(x)$ both contain $M + N + 1$ parameters. For example, if we are given $T_5(x)$, we can then compute P_3^2 as follows:

$$\frac{A_0 + A_1 x + A_2 x^2}{1 + B_1 x + B_2 x^2 + B_3 x^3} = a_0 + a_1 x + +a_2 x^2 + a_3 x^3 + a_4 x^4 + a_5 x^5 + O(x^6).$$

		N — order of numerator				
		0	1	2	3	.
M —	0	1	$1-\frac{x}{2}$	$1-\frac{1}{2}x+\frac{1}{3}x^2$	$1-\frac{1}{2}x+\frac{1}{3}x^2-\frac{1}{4}x^3$	.
order of	1	$\frac{1}{1+\frac{x}{2}}$	$\frac{1+\frac{x}{6}}{1+\frac{2x}{3}}$	$\frac{1+\frac{x}{4}-\frac{x^2}{24}}{1+\frac{3x}{4}}$	$\frac{1+\frac{3x}{10}-\frac{x^2}{15}+\frac{x^3}{60}}{1+\frac{4x}{5}}$	.
denomi-	2	$\frac{1}{1+\frac{x}{2}-\frac{x^2}{2}}$	$\frac{1+\frac{x}{2}}{1+x+\frac{x^2}{6}}$	$\frac{1+\frac{7x}{10}+\frac{x^2}{30}}{1+\frac{6x}{5}+\frac{3x^2}{10}}$	$\frac{1+\frac{5x}{6}+\frac{x^2}{15}-\frac{x^3}{180}}{1+\frac{4x}{3}+\frac{2x^2}{5}}$	.
nator	3	$\frac{1}{1+\frac{x}{2}-\frac{x^2}{12}+\frac{x^3}{24}}$	$\frac{1+\frac{19x}{30}}{1+\frac{17x}{15}+\frac{7x^2}{30}-\frac{x^3}{90}}$	$\frac{1+x+\frac{11x^2}{60}}{1+\frac{3x}{2}+\frac{3x^2}{5}+\frac{x^3}{20}}$	$\frac{1+\frac{17x}{14}+\frac{x^2}{3}+\frac{x^3}{140}}{1+\frac{12x}{7}+\frac{6x^2}{7}+\frac{4x^3}{35}}$	.
	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	

TABLE 1. Beginning of the Padé table for $f(x) = 1-(1/2)x+(1/3)x^2-(1/4)x^3+-\cdots$

After multiplying both sides with the denominator and equating the coefficients of the different powers of x, we obtain the two relations

$$\begin{bmatrix} a_2 & a_1 & a_0 \\ a_3 & a_2 & a_1 \\ a_4 & a_3 & a_2 \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ B_3 \end{bmatrix} = - \begin{bmatrix} a_3 \\ a_4 \\ a_5 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} A_0 \\ A_1 \\ A_2 \end{bmatrix} = \begin{bmatrix} a_0 & 0 & 0 \\ a_1 & a_0 & 0 \\ a_2 & a_1 & a_0 \end{bmatrix} \begin{bmatrix} 1 \\ B_1 \\ B_2 \end{bmatrix}.$$

The first relation can be solved for B_1, B_2, B_3 after which A_0, A_1, A_2 follow from the second one. The utility of this type of conversion is illustrated in the following example.

Example: Suppose we are given the Taylor expansion

$$f(x) = 1 - \frac{1}{2}x + \frac{1}{3}x^2 - \frac{1}{4}x^3 + \frac{1}{5}x^4 - \dots \tag{10.15}$$

and wish to approximate $f(2)$ (at which point the series (10.15) is rapidly diverging). The Padé table shown in Table 1 lists the corresponding rational functions $P_M^N(x)$ for increasing M and N. The top row (denominators of degree zero) merely repeats the successive truncations of (10.15). Especially the entries down the main diagonal typically provide much improved functional approximations. If we numerically evaluate these for $x = 2$, we get rapid geometric convergence to $\frac{1}{2}\log 3$, with the error reaching 10^{-16} for $M = N = 13$. This amounts to an analytic continuation of (10.15) since $f(x) = \frac{1}{x}\log(1+x)$ when $|x| < 1$).

□

10.3.3.2. *Shanks' method.* With this approach we typically do not accelerate a sequence of functions, but instead a sequence s_n of numerical values. We start by the much oversimplified assumption that we are dealing with an approximation s_n of s, given by

$$s_n = s + \alpha\, q^n, \tag{10.16}$$

where α and q are two unknown constants. Writing down (10.16) also for $n-1$ and $n+1$ gives three equations from which we can eliminate α and q, giving the relation

$$s = s_n - \frac{(s_{n+1} - s_n)(s_n - s_{n-1})}{(s_{n+1} - s_n) - (s_n - s_{n-1})}.$$

Whenever we have three consecutive s_n-values, we can thus obtain an approximation for s. A convenient way to arrange the extrapolation work is to place the original s_n-values in a column, followed by another column (2 steps shorter) with extrapolated results, then another column 2 steps shorter again with a further extrapolation from the previous column, etc.

Example: We consider again the test case we used for the Padé method above, i.e. to approximate the function (10.15) at $x = 2$ based only on some leading entries in the sequence of partial sums

$$s_n = 1 - \frac{1}{2}\,2 + \frac{1}{3}\,2^2 + - \ldots + \frac{(-1)^{n-1}}{n}\,2^{n-1}, \;\; n = 1, 2, \ldots \; .$$

Noting that one can write $s_n = s_{n-1} + \frac{(-1)^{n-1}}{n} 2^{n-1}$, one can calculate the table just described,

$$
\begin{array}{lll}
s_1 & \to & 1.0000 \\
s_2 = s_1 - \frac{1}{2}\times 2 & \to & 0.0000 \\
s_3 = s_2 + \frac{1}{3}\times 2^2 & \to & 1.3333 \quad 0.5714 \\
s_4 = s_3 - \frac{1}{4}\times 2^3 & \to & -0.6667 \quad 0.5333 \\
s_5 = s_4 + \frac{1}{5}\times 2^4 & \to & 2.5333 \quad 0.5641 \quad 0.5504 \\
s_6 = s_5 - \frac{1}{6}\times 2^5 & \to & -2.8000 \quad 0.5333 \quad 0.5487 \\
s_7 = s_6 + \frac{1}{7}\times 2^6 & \to & 6.3429 \quad 0.5684 \quad 0.5497 \quad 0.5493 \\
s_8 = s_7 - \frac{1}{8}\times 2^7 & \to & -9.6571 \quad 0.5247 \quad 0.5490 \quad 0.5493 \\
s_9 = s_8 + \frac{1}{9}\times 2^8 & \to & 18.7873 \quad 0.5829 \quad 0.5496 \quad 0.5493 \quad 0.5493 \\
\vdots \qquad\qquad \vdots & & \vdots \qquad \vdots \qquad \vdots \qquad \vdots \qquad \vdots \qquad \ddots
\end{array}
$$

Despite the fact that the sequence s_n is wildly oscillating, the top entries in successive columns seem to converge. The error in the last entry in line 9 (as compared to $\frac{1}{2}\log 3$, when using Matlab's standard double precision arithmetic) has in fact decreases to $3\cdot 10^{-16}$, after which it starts to slowly grow again.

□

As we saw in Section 10.3.1, direct summation of the leading terms of an asymptotic (divergent) series can work very well. However, in more difficult situations (with expansions that diverge right from the start, or when only very few terms are available), acceleration methods are invaluable (although they are by no means infallible—use with care!). One benefit of having acceleration methods available is that we only rarely need to be concerned about whether the expansions are convergent or divergent, with the latter case probably being the more common one.

Exercise 22. Evaluate numerically to ten decimal places,

$$\sum_{n=2}^{\infty} \frac{1}{n\left(\log n\right)^2}. \quad (\text{answ: } \approx 2.1097428012)$$

How many terms would one need to include, if one sums directly, i.e. for which value of N is

$$\sum_{n=N+1}^{\infty} \frac{1}{n\left(\log n\right)^2} < 10^{-10}? \quad \left(\text{answ: } \approx 10^{434\,000\,000}\right)$$

□

EXERCISE 23. Calculate the Padé approximation of the function with Taylor expansion

$$f(x) = x - x^3 + x^5 - x^7 + \cdots .$$

First note that the radius if convergence of the series is $R = 1$. Use the Padé approximation to find the poles of $f(x)$ in the complex plane. Experiment with different values of M and N.

□

EXERCISE 24. Find the Padé approximation of

$$f(x) = x + \frac{1}{3}x^3 + \frac{2}{15}x^5 + \frac{17}{315}x^7 + \cdots .$$

Use $M = 2$ and find the positions of the poles in the Padé approximation. Compare these values with the value of $\pm\pi/2$. Any idea what $f(x)$ might be?

□

EXERCISE 25. Find the Padé approximation for $f(2)$ where $f(x) = 1 + x + x^2 + x^3 + x^4 + x^5 + \cdots$. Use $M = 3,\ N = 2$. Note that this series is also divergent but the example above indicates that this needs not be a problem. Nevertheless for this example one can get into trouble trying to solve for the Padé coefficients. Since the matrix that appears in the equation for the B_n coefficients is singular, in this case it allows an infinite number of different solutions for the B_n coefficients. Let us try a few solutions. For $B_1 = -1$, $B_3 = 0 = B_2$, calculate the A_m to find that the Padé approximation is given by

$$\frac{1}{1-x}.$$

Another choice is $B_1 = -\frac{1}{2} = B_2,\ \ B_3 = 0$, and calculating the values of the A_n gives the Padé approximation

$$\frac{1 + \frac{1}{2}x}{1 - \frac{1}{2}x - \frac{1}{2}x^2}.$$

To see that this is also a valid Padé approximation, write

$$\begin{aligned}\frac{1+\frac{1}{2}x}{1-\frac{1}{2}x-\frac{1}{2}x^2} &= 1+x\frac{1+\frac{1}{2}x}{1-\frac{1}{2}x-\frac{1}{2}x^2}\\ &= 1+x+x^2\frac{1+\frac{1}{2}x}{1-\frac{1}{2}x-\frac{1}{2}x^2}\\ &= 1+x+x^2+x^3\frac{1+\frac{1}{2}x}{1-\frac{1}{2}x-\frac{1}{2}x^2}\\ &\text{etc} \quad .\end{aligned}$$

Now calculate $f(2)$ using the Padé approximation for both cases. What do you observe? Finally choose $M = 1,\ N = 4$, and calculate the Padé approximation for $f(2)$. Conclusion: Since $f(x) = \frac{1}{1-x}$ is already in Padé form, the singularity of the system is an indication that the choice of M and N allows a redundancy. However, any consistent choice of coefficients provide a valid, albeit an unnecessary complicated approximation.

□

10.4. An example of a perturbation expansion for an ODE

The topic of asymptotic expansions for ordinary differential equations (ODEs) is immense, and can barely be touched upon here. We limit ourselves to a single example, indicating how an analytically unsolvable ODE can be approximated by a sequence of solvable ones. In this case, the expansion approach aims at providing approximations across a complete domain. In other often occurring situations, one would instead be interested in how solutions behave near a singular point, as the independent parameter goes towards infinity, or when a parameter ε causes the coefficient for the highest order derivative to vanish as $\varepsilon \to 0$, frequently leading to *boundary layers*—thin transition regions with very sharp gradients.

Example: Projectile problem. Suppose a projectile (think canon ball) is shot straight up, attaining height $y(t)$ after time t. It starts at the earth's surface (height zero, $y(0) = 0$) with an initial velocity $y'(0) = v_0$. With g denoting the acceleration of gravity, and R the radius of the earth, its height obeys the ODE $\frac{d^2y}{dt^2} = -\frac{g\,R^2}{(y+R)^2}$. Several simplifications assumptions have already been made, such as no air resistance,

earth not rotating, but at least we have assumed the earth to be round and that the force of gravity decays appropriately with height. Guided by *non-dimensionalization* and *scaling* (Chapter 9), we change variables $y = \frac{v_0^2}{g} y^*$, $t = \frac{v_0}{g} t^*$, and introduce a small parameter $\varepsilon = \frac{v_0^2}{gR}$. After dropping the stars for y^* and t^*, the ODE takes the simpler non-dimensional form

$$\frac{d^2y}{dt^2} = -\frac{1}{(1+\varepsilon\, y)^2}, \quad y(0) = 0, \;\; y'(0) = 1. \tag{10.1}$$

The idea is to replace this analytically unsolvable ODE with a series of solvable ones. We do this in three different ways, all aiming towards determining the functions $y_k(t)$ in an asymptotic expansion

$$y(t) = y_0(t) + \varepsilon\, y_1(t) + \varepsilon^2\, y_2(t) + \ldots \tag{10.2}$$

□

10.4.1. Equate powers.

Substituting (10.2) into (10.1) gives, after some expansions and simplifications, a separate ODE for each power of ε:

$$\begin{array}{lll}
\varepsilon^0: & y_0''(t) = -1, & y_0(0) = 0, \;\; y_0'(0) = 1, \\
\varepsilon^1: & y_1''(t) = 2y_0(t), & y_1(0) = 0, \;\; y_1'(0) = 0, \\
\varepsilon^2: & y_2''(t) = 2y_1(t) - 3y_0^2(t), & y_2(0) = 0, \;\; y_2'(0) = 0, \\
\vdots & \vdots &
\end{array}$$

Both the creation and the solution of these successive ODEs are greatly simplified by a symbolic algebra system. In Mathematica, the statements:

```
n = 2; y[t_]:=∑_{k=0}^{n} y_k[t] ε^k;
ODEs = LogicalExpand[Series[y''[t]+1/(1+ε y[t])^2, {ε,0,n}] == 0]
BC = Flatten[Table[{y_k[0]==0,y_k'[0]==0},{k,0,n}]];BC[[2]]=y_0'[0]==1;BC
unknowns =Table[y_k[t],{k,0,n}]
DSolve[Prepend[BC,ODEs],unknowns,t]
```

produces the following four lines of output, describing in turn the sequence of ODEs, their boundary conditions, lists the unknown functions, and finally gives their functional form

$1 + (\mathtt{y}_0)''\mathtt{[t]==0\ \&\&\ -2\ y}_0\mathtt{[t]\ +\ }(\mathtt{y}_1)''\mathtt{[t]==0\ \&\&\ 3\ y}_0\mathtt{[t]^2-2\ y}_1\mathtt{[t]\ +}$
$(\mathtt{y}_2)''\mathtt{[t]==0}$
$\mathtt{\{y}_0\mathtt{[0]==0,\ }(\mathtt{y}_0)'\mathtt{[0]==1,\ y}_1\mathtt{[0]==0,\ }(\mathtt{y}_1)'\mathtt{[0]==0,\ y}_2\mathtt{[0]==0,\ }(\mathtt{y}_2)'\mathtt{[0]==0\}}$
$\mathtt{\{y}_0\mathtt{[t],\ y}_1\mathtt{[t],\ y}_2\mathtt{[t]\}}$
$\mathtt{\{\{y}_0\mathtt{[t]}\to \tfrac{1}{2}(2\mathtt{t-t}^2),\ \mathtt{y}_1\mathtt{[t]}\to \tfrac{1}{12}(4\mathtt{t}^3\mathtt{-t}^4),\ \mathtt{y}_2\mathtt{[t]}\to \tfrac{1}{360}(-90\mathtt{t}^4\mathtt{+66t}^5\mathtt{-11t}^6)\mathtt{\}\}}$

10.4.2. Parametric differentiation.

The functions $y_0(t),\ y_1(t),\ y_2(t), \ldots$ are the coefficients of the Taylor expansion of $y(t) = y(t, \varepsilon)$ in terms of ε (cf. equation (10.2)). So we can get them by setting $\varepsilon = 0$ in $y(t, \varepsilon)$, $\dfrac{\partial y(t,\varepsilon)}{\partial \varepsilon}$, $\dfrac{1}{2!}\dfrac{\partial y(t,\varepsilon)}{\partial \varepsilon}$, etc. In Mathematica, this can again be implemented in just a few lines (for an arbitrary value of n):

$\mathtt{n\ = 2;\ y[t\_]:=}\sum_{k=0}^{n}\mathtt{y}_k\mathtt{[t]\ }\epsilon^{\mathtt{k}}\mathtt{;}$

$\mathtt{Do[ode=If[k==0,\ y''[t]+}\dfrac{1}{(1+\epsilon\ \mathtt{y[t]})^2}\mathtt{,D[ode,}\epsilon\mathtt{]];}$
$\mathtt{sol=DSolve[\{ode==0/.}\epsilon\to 0\mathtt{,y}_k\mathtt{[0]==0,\ y}_k'\mathtt{[0]==If[k==0,1,0]\},}$
$\mathtt{y}_k\mathtt{[t],t];y}_k\mathtt{[t\_]=y}_k\mathtt{[t]/.sol[[1]],\{k,0,n\}];\ \ y[t]}$

producing the output

$\frac{1}{2}(2t - t^2) + \frac{1}{12}(4t^3 - t^4)\varepsilon + \frac{1}{360}(-90t^4 + 66t^5 - 11t^6)\varepsilon^2.$

10.4.3. Direct iteration. The first step this time is to rewrite the ODE so that setting the right hand side to zero provides the initial $\varepsilon = 0$ approximation. Since the right hand side of (10.1) takes the value -1 for $\varepsilon = 0$, we add one to both sides,

$$\frac{d^2y}{dt^2} + 1 = 1 - \frac{1}{(1+\varepsilon\ y)^2} = \varepsilon\ \frac{2y + \varepsilon y^2}{(1+\varepsilon\ y)^2}, \quad y(0) = 0,\ \ y'(0) = 1.$$

One can then iterate

$$\frac{d^2y_{n+1}}{dt^2} + 1 = \varepsilon\frac{2y_n + \varepsilon y_n^2}{(1+\varepsilon\ y_n)^2},\ y_{n+1}(0) = 0,\ \ y'_{n+1}(0) = 1,\ \ n = 0, 1, 2, \ldots\ ,$$

where $y_0(t) = -\frac{1}{2}t^2 + t$. Before we give a short Mathematica code that solves it automatically, it may be of some value to derive some of the terms by hand.

Given $y_0(t)$ and anticipating an $O(\varepsilon)$ correction, we keep only first order terms on the right hand side,

$$\frac{d^2y_1}{dt^2} + 1 = \varepsilon 2y_0(t).$$

Since $y_0(t)$ is a solution for $\varepsilon = 0$, we need to calculate the $O(\varepsilon)$ contribution to find,

$$y_1(t) = \left(-\frac{1}{2}t^2 + t\right) + \varepsilon\left(\frac{1}{3}t^2 - \frac{1}{12}t^4\right).$$

In order to find $y_2(t)$ we only need to find the $O(\varepsilon^2)$contribution from

$$\begin{aligned}\frac{d^2y_2}{dt^2} + 1 &= \varepsilon\frac{2y_1 + \varepsilon y_1^2}{(1+\varepsilon y_1)^2} \\ &= \varepsilon\left(2(y_0(t) + 2\varepsilon y_\varepsilon(t) + \varepsilon y_0(t)^2\right)(1 - 2\varepsilon y_0(t) - 2\varepsilon y_\varepsilon(t)) + O(\varepsilon^3),\end{aligned}$$

where we have written $y_1(t) = y_0(t) + \varepsilon y_\varepsilon(t)$. Simplifying, it follows that we need to solve

$$\frac{d^2y_2}{dt^2} + 1 = 2\varepsilon y_0(t) + \varepsilon^2\left(-3y_0(t)^2 + 2y_\varepsilon(t)\right) + O(\varepsilon^3).$$

Since we only need the second order contribution, having already calculated up to first order contributions, we solve

$$\begin{aligned}\frac{d^2y}{dt^2} &= \varepsilon^2\left(-3y_0(t)^2 + 2y_\varepsilon(t)\right) \\ &= \varepsilon^2\left(-\frac{11}{12}t^4 + \frac{11}{3}t^3 - 3t^2\right)\end{aligned}$$

to find that

$$y_2(t) = \left(-\frac{1}{2}t^2 + t\right) + \varepsilon\left(\frac{1}{3}t^2 - \frac{1}{12}t^4\right) + \varepsilon^2\left(-\frac{11}{360}t^6 + \frac{11}{60}t^5 - \frac{1}{4}t^4\right).$$

Again, it is much easier to let Mathematica do the algebra:

```
n = 2; LHS[y_]:=D[y,{t,2}]+1; RHS[y_]:=ε(2y+εy^2)/(1+ε y)^2; y_-1[t_]:=0;
Do[sol=DSolve[{LHS[y_k[t]]==Normal[Series[RHS[y_k-1[t]],{ε,0,k}]],
y_k[0]==0, y_k'[0]==1},y_k[t],t]; y_k[t]=y_k[t]/.sol[[1]],{k,0,n}];
Collect[y_n[t],ε]
```

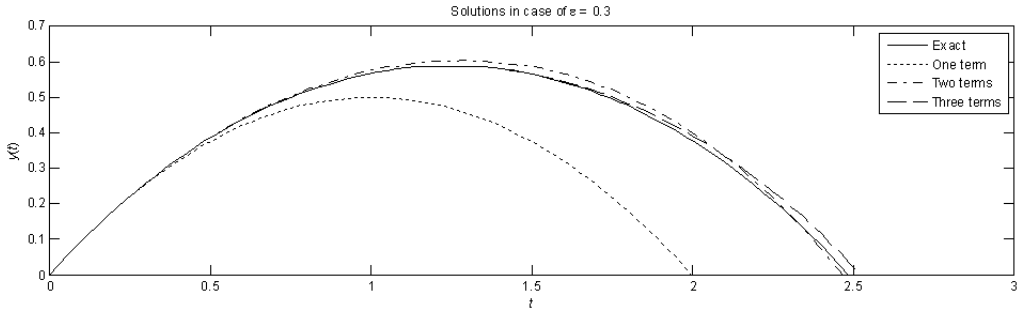


FIGURE 10.4.1. Comparison between the exact solution to (10.1) and the three leading approximations in case of $\varepsilon = 0.3$.

producing the output

$$y_2(t) \;\; = \tfrac{1}{360}(360t - 180t^2) + \tfrac{1}{360}(120t^3 - 30t^4)\varepsilon + \tfrac{1}{360}(-90t^4 + 66t^5 - 11t^6)\varepsilon^2.$$

Correct, but for some reason arranged with a common denominator of 360.

Figure 10.4.1 compares the exact solution against its three leading approximations in the case of $\varepsilon = 0.3$. Not surprisingly, the accuracies are the best near $t = 0$, where the two initial conditions are specified. However, the approximations are also quite accurate across the full time span of the problem (and will become even more accurate for smaller ε values).

10.5. Asymptotic methods for integrals.

This is again such an extensive topic that we can only give a brief flavor here, in particular since we cannot go deeply into analytic function theory. The types of integrals we consider and some appropriate methods for them are summarized in Table 2.

We have already come across integration by parts in connection with Erf(z) in Section 10.3.1.1. In the subsections below, we briefly describe some of the other techniques.

Some classes of integrals	**Techniques for asymptotics**
Variable domain $\int_0^x \cdots$, $\int_x^\infty \cdots$	Taylor or Laurent expansions Integration by part Expand and integrate term-by-term Change variable to Laplace integral
Laplace integral $\int_a^b f(t)e^{x\phi(t)}dt$, $\phi(t)$ real, $x \longrightarrow \infty$	Integration by part Laplace method / Watson's lemma Steepest descent
Fourier integrals $\int_a^b f(t)e^{ix\phi(t)}dt$, $\phi(t)$ real, $x- > \infty$	Integration by part Stationary phase Steepest descent
Contour integrals $\int_C f(t)e^{z\phi(t)}dt$; $\phi(t)$ may be complex Includes Laplace and Fourier	Steepest descent

TABLE 2. Brief summary of some classes of integrals and appropriate asymptotic techniques

10.5.1. Laplace method / Watson's lemma. If one can bring an integral to the form

$$I(x) = \int_0^b f(t)e^{-xt}dt \ \ (b > 0), \text{ with } f(t) = t^\alpha \sum_{n=0}^{\infty} a_n t^{\beta_n} \text{ (for } t \to 0+) \tag{10.1}$$

then the complete asymptotic expansion for $x \to \infty$ is given by

$$I(x) \sim \sum_{n=0}^{\infty} \frac{a_n \Gamma(\alpha + \beta_n + 1)}{x^{\alpha+\beta_n+1}}, \tag{10.2}$$

where $\Gamma(\cdot)$ is the gamma function (see Appendix). The concept is simple. For increasing x, only the tiniest neighborhood of $t = 0$ matters for the function $f(t)$. So it makes no difference if we change b to ∞. All the integrals that arise when we use the expansion for $f(t)$ can then be evaluated exactly in terms of gamma functions, giving the right hand side of (10.2).

Example: We revisit the Erf(z) example from Section 10.3.1.1. Starting from (10.3), we first change variables $t = z + \tau$ in order to shift the integral to the interval

$[0,\infty]$. After one more variable change, $u = 2\tau$,

$$\int_z^\infty e^{-t^2}dt = e^{-z^2}\int_0^\infty e^{-\tau^2}e^{-2\tau\, z}d\tau = \frac{1}{2}e^{-z^2}\int_0^\infty e^{-(u/2)^2}e^{-zu}du,$$

we arrive at an integral that is in exactly the form needed for Watson's lemma. Using $e^{-(u/2)^2} = 1 - \frac{u^2}{1\cdot 2^2} + \frac{u^4}{2!\cdot 2^4} - \ldots$, (10.2) gives

$$\int_z^\infty e^{-t^2}dt \sim \frac{1}{2}e^{-z^2}\left\{\frac{1}{1} - \frac{1}{(2z^2)^1} + \frac{1\cdot 3}{(2z^2)^2} - \frac{1\cdot 3\cdot 5}{(2z^2)^3} + - \ldots\right\}.$$

□

Another application of Laplace method / Watson's lemma arises with integrals of the form

$$I(x) = \int_a^b f(t)e^{x\phi(t)}dt,$$

where $\phi(t)$ takes its maximum value at one of the end points a or b. One can then change variable to get the integral into the form (10.1). If the maximum value happens to occur at an interior point $t = c$, one can split the interval in two, and c becomes an end point for both sub-intervals. If $\phi''(c) < 0$, there is an easy formula for picking up the leading order asymptotic term,

$$I(x) \sim \frac{\sqrt{2\pi}f(c)e^{x\phi(c)}}{\sqrt{-x\phi''(c)}}. \tag{10.3}$$

Example: At first glance, it may look as if

$$I(x) = \int_0^\infty e^{-\frac{1}{t}}e^{-xt}dt = \frac{2}{\sqrt{x}}K_1(2\sqrt{x})$$

(with K_1 denoting a first order Bessel function) is already in the desired form for applying Watson's lemma. However $e^{-\frac{1}{t}}$ goes to zero too fast as $t \to 0$ for the required type of expansion in (10.1) to exist. The next idea then is to consider the whole exponent $-xt - \frac{1}{t}$ as a function of t, and note that it has a 'movable' maximum at $t = \sqrt{x}$. We therefore change variable $t = s/\sqrt{x}$ to make this maximum point stationary. The integral now becomes $I(x) = \frac{1}{\sqrt{x}}\int_0^\infty e^{-\sqrt{x}(s+1/s)}ds$. With $f(s) = 1$ and $\phi(s) = s + 1/s$ (featuring a local maximum at $s = 1$), (10.3) applies directly, giving $I(x) \sim \frac{\sqrt{\pi}e^{-2\sqrt{x}}}{x^{3/4}}$.

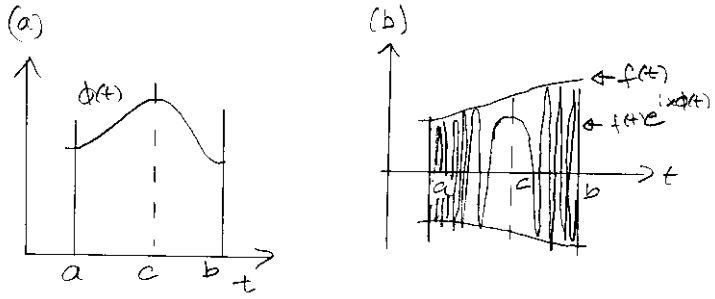


FIGURE 10.5.1. Schematic sketches of (a) $\phi(t)$ and (b) $f(t)\ e^{i\,x\,\phi(t)}$ (real or imaginary part) in cases for which the stationary phase approach applies.

□

10.5.2. Stationary phase.

This method applies to integrals $\int_a^b f(t)e^{ix\phi(t)}dt$, where $\phi(t)$ is real and takes its maximum value at $t = c$ inside $[a, b]$.

Figure 10.5.1 sketches a typical situation. As $x \to \infty$, the contribution from the vicinity of $t = c$ becomes increasingly dominant because of all the cancellations that happen elsewhere. By local expansions around $t = c$, it is not hard to pick up the leading asymptotic term for large values of x,

$$I(x) \sim \sqrt{\frac{2\pi}{x|\phi''(c)|}} f(c)\ e^{i[x\phi(c)\pm\frac{\pi}{4}]}, \tag{10.4}$$

where the sign marked $\pm$ has to be chosen to match the sign of $\phi''(c)$. Further terms are difficult to obtain by this approach (since they no longer depend only on what happens near $t = c$). If one needs an extended (or complete) expansion, *steepest descent* usually offers the best chance.

Example: We want to find the leading behavior of

$$I(x) = \int_0^{\pi/2} e^{ix\cos t}dt. \tag{10.5}$$

We can immediately apply (10.4) with $\phi(t) = \cos t$ featuring a stationary point at $t = 0$, at which $\phi''(0) = -1$. Since this point is at the domain boundary (rather

than inside the domain), we need to halve the result from (10.4), giving $I(x) \sim \sqrt{\frac{\pi}{2x}} e^{i(x-\pi/4)}$.

□

10.5.3. Steepest descent. This approach is by far the most powerful one for obtaining both leading terms and complete expansions of many integrals. Unfortunately, it is also the approach that requires the most theoretical background, placing it out of the scope of this book. Let it only be said here that, if the integrand in $\int_C f(t)e^{z\phi(t)}dt$ is an *analytic function* of t, one is then free to change the contour C along which one carries out the integration without this affecting the result (as long as the path, while it is being moved, does not cross any singularities). For example, one might want to change a path from a to b along the real axis into one that makes a detour in the complex plane. A function like e^{ixt} that is very highly oscillatory along the real t-axis (with our usual assumption of $x \to \infty$), instead decays extremely fast if we move upwards in the complex plane, and Watson's lemma may then provide the complete asymptotic expansion in cases for which stationary phase could only give the leading term. One can often find a *stationary point* t_s in the complex plane (characterized by $\phi'(t_s) = 0$) and an brief curve segment passing through it along which $\mathrm{Re}\phi(t)$ descends in value, such that the complete expansion follows from just local expansions at this t_s-location. We cannot give any demonstration of the general steepest descent approach here, but we can illustrate the idea of path change:

Example: Find the next two terms in the expansion for (10.5). We can change variable $s = \cos t$ to obtain

$$I(x) = \int_0^1 \frac{1}{\sqrt{1-s^2}} e^{ixs} ds.$$

Instead of integrating along the real axis from $s = 0$ to $s = 1$, we integrate straight up the imaginary axis, then far up we go horizontally over to $\mathrm{Re}(s)= 1$, and finally straight down until we reach $s = 1$. The integrand is vanishingly small along the top horizontal path. Along the imaginary axis, we change variable $s = i\sigma$ to obtain $I_1(x) = i\int_0^\infty \frac{1}{\sqrt{1+\sigma^2}} e^{-x\sigma} d\sigma$, and along the other vertical path $s = 1 + i\sigma$ to obtain $I_2(x) = ie^{ix}\int_0^\infty \frac{1}{\sqrt{\sigma(\sigma-2i)}} e^{-x\sigma} d\sigma$. Both of these integrals are exactly what is needed

Name of function	**Special cases of more general families of ODEs**	**Example of a solution to the ODE listed to the left**
AiryAi	$\frac{d^2y}{dx^2} - xy = 0$	$\int_{-\infty}^{\infty} \cos(\frac{t^3}{3} - tx)dt = 2\pi\ Ai(x)$
BesselK	$\frac{d^2y}{dx^2} + \frac{1}{x}\frac{dy}{dx} - (1 + \frac{1}{x^2})\ y = 0$	$\int_0^{\infty} e^{-1/t}\ e^{-xt}dt = \frac{2}{\sqrt{x}}K_1(2\sqrt{x})$
BesselJ	$\frac{d^2y}{dx^2} + \frac{1}{x}\frac{dy}{dx} + y = 0$	$\int_0^{\pi/2} e^{i\ x\ \cos t}dt =$
StruveH	$\frac{d^2y}{dx^2} + \frac{1}{x}\frac{dy}{dx} + y - \frac{2}{\pi\ x} = 0$	$= \frac{\pi}{2}\ (J_0(x) + i\ H_0(x))$

TABLE 3. Some examples we have come across of integral solutions to linear variable coefficient ODEs

for Watson's lemma. After a bit of algebra, we get

$$I(x) \sim \sqrt{\frac{\pi}{2}}\frac{1}{x^{1/2}}e^{i(x-\pi/4)} + \frac{i}{x} - \sqrt{\frac{\pi}{2}}\frac{1}{8x^{3/2}}e^{i(x+\pi/4)} + O(x^{-5/2}).$$

We recognize the first term from the stationary phase calculation. The change of contour approach opened up the possibility of calculating any number of terms.

□

The method in this last example is virtually identical to what was used in Section 10.1.2. Instead of evaluating (10.4) along the real axis from 0 to 2π, we follow an equivalent path to the one in the example above. Then it turns out that only the local properties at three points in the complex plane matters for the result.

A footnote to the examples above of oscillatory integrals: We have just seen several such examples, summarized in Table 3. A good general reference is [**15**].

Not surprisingly, these are just glimpses of a systematic concept. There are direct procedures available to write down integral representations for all the independent solutions to many linear variable coefficient ODEs. Important cases are listed in handbooks of special functions, such as [**15**], mentioned above. Taylor (or Taylor-like) expansions can normally also be found for the solutions (assume unknown coefficients, substitute into the ODE, and then equate coefficients). However, even when the solutions are entire functions (with Taylor expansions converging everywhere; $R_c = \infty$), this approach quickly turns into a numerical disaster if we are interested in x increasing, as we saw in Section 10.3.1.1 for the error function. In contrast, integral representations can be approximated better and better in this same limit, and are therefore very useful in many contexts.

10.6. Appendix

10.6.1. The gamma function. The gamma function $\Gamma(z)$ appears regularly in asymptotic expansions of integrals, as encountered in Section 10.5, see for example, **[2, 15]**. There are three definitions of the gamma function that interest us. The first is in terms of an infinite limit, named after Euler,

$$\Gamma(z) = \lim_{n\to\infty} \frac{1\cdot 2\cdots n}{z(z+1)\cdots(z+n)} n^z, \quad z \neq 0, -1, -2, \ldots, \tag{10.1}$$

where z can be complex. From this definition immediately follows the important recursion relation,

$$\Gamma(z+1) = z\Gamma(z). \tag{10.2}$$

It also follows that

$$\Gamma(1) = \lim_{n\to\infty} \frac{1\cdot 2\cdots n}{1\cdot 2\cdots n\cdot(n+1)} n = 1.$$

Using the recursion relation it now follows that

$$\begin{aligned} \Gamma(2) &= 1\cdot\Gamma(1) = 1 \\ \Gamma(3) &= 2\cdot\Gamma(2) = 2, \end{aligned}$$

and finally that

$$\Gamma(n) = (n-1)!,$$

for n a positive integer.

The second definition, also due to Euler, is in terms of a definite integral, the Euler integral,

$$\Gamma(z) = \int_0^\infty e^{-t} t^{z-1} dt, \quad \mathrm{Re} z > 0, \tag{10.3}$$

where the condition on z is to ensure convergence of the integral. This is of course the form that is most useful for asymptotic expansions.

By changing variables, $t = s^2$ the gamma function can be written in another form that occurs often in practice,

$$\Gamma(z) = \int_0^\infty e^{-s^2} s^{2z-1} ds. \tag{10.4}$$

Thus we note that the gamma function is also related to the Gauss error integral for $z = \frac{1}{2}$, and

$$\Gamma\left(\frac{1}{2}\right) = \sqrt{\pi}.$$

The third definition is due to Weierstrass,

$$\frac{1}{\Gamma(z)} = ze^{\gamma z} \prod_{n=1}^{\infty} \left(1 + \frac{z}{n}\right) e^{-\frac{z}{n}}, \tag{10.5}$$

where γ is the Euler constant (10.8) encountered in Section 10.3.

It is not particularly hard to show the equivalence of the three definitions, for the details see Arfken [**2**].

10.6.2. Matlab code for van der Pol oscillator. We first rewrite (10.1) as a system of two first order ODEs. With $y_1(t) = y(t)$, $y_2(t) = y'(t)$, it becomes

$$\frac{d}{dt}\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \begin{bmatrix} y_2 \\ k(1-y_1^2)y_2 - y_1 \end{bmatrix}.$$

Matlab takes automatically care of issues such as dynamically varying the step size, etc. Figure 10.1.1 was produces by the following Matlab code:

```
close all; clear all;
kv = [0 4 10];%Compute the van der Pol solution for three different
k-values
for n = 1:3
k = kv(n);
dy = @(t,y)[y(2);k*(1-y(1)^2)*y(2)-y(1)]; %ODE RHS to be used by ODE solver
[T,Y] = ode15s(dy,[0,40],[1,0]); %Use for ex.  Matlab's built-in ode15s
subplot(1,3,n); plot(T,Y(:,1),'.-'); %Plot the result
title (['k = ',num2str(k)]); xlabel('t'); ylabel('y')
end
```

Part 3

NUMERICAL TECHNIQUES

CHAPTER 11

LINEAR SYSTEMS: LU, QR AND SVD FACTORIZATIONS

11.1. Introduction.

A general linear system $A\mathbf{x} = \mathbf{b}$ of m equations in n unknowns can be graphically written as

$$\left[\ A\ \right]_{m\times n}\ \left[\ \mathbf{x}\ \right]_n = \left[\ \mathbf{b}\ \right]_m\ . \tag{11.1}$$

The main goal of this chapter is to solve *any* linear system of the form (11.1). If you have had some exposure to the solution of linear systems you might be tempted to think that this is quite trivial. For a square system $m = n$ you might know that there is a unique solution provided the determinant is not zero, i.e. the system is non-singular. If not, a solution does not exist. The fact that we emphasize '*any* linear system' of the form (11.1) makes life a little more interesting. A few examples should illustrate what we have in mind.

Example:: Solve for $x = 1$. This example is so trivial that it hardly warrants comment. The point is that is of the form (11.1) with $m = n = 1$. Since the determinant is non-zero, the solution is trivial.

Example:: Let us now complicated things slightly and solve simultaneously for

$$\begin{aligned} x &= 1 \\ x &= 2. \end{aligned}$$

At first glance this does not make sense, x cannot simultaneously equal 1 and 2. Yet, it is of the form (11.1) with $A = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, and $\mathbf{b} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$. The best one can do in this case is to solve it as best we can, perhaps choosing

$x = 1.5$ as the best compromise. This turns out to be the least squares solution.

Example:: At the other extreme we may also want to solve for $x + y = 1$. Again this is of the form (11.1) with $A = \begin{bmatrix} 1 & 1 \end{bmatrix}$ and $\mathbf{b} = 1$. Now the problem is that we have an infinite number of solutions, and we have to select one. The question is to identify a natural choice. Is there any obvious reason why you would choose $x = \frac{1}{2} = y$? Towards the end of this chapter you will realize that this is the so-called generalized solution.

In the process of learning how to solve general systems of the form (11.1) you will be exposed to a number of key issues, issues of fundamental importance, theoretically and in practice.

A key issue is to determine whether systems such as (11.1) have zero, one or infinitely many solutions (these turn out to be the only possibilities). Section **??** presents a (primarily theoretical) procedure to determine the solution set—reducing the matrix to *echelon form*. The most important case of linear systems is probably when there is exactly one solution. *Gaussian elimination* is then appropriate (Section 11.2. The case with no solution is also ubiquitous—it arises for example if more measurements are made in an experiment than there are free parameters to be determined. The least-squares problem then amounts to finding the vector x that minimizes the error $\|A\mathbf{x} - \mathbf{b}\|$ (where $\|\mathbf{z}\| = \|\mathbf{z}\|_2 = \sqrt{\Sigma_{i=1}^{n} z_i^2}$). The case of infinitely many solutions arises when we have more unknowns than equations. One can think of these equations as a constraint on the variables and then use the freedom provided by the infinitely many solutions to select one that optimizes some desired quantity. This situation arises in filter design, for example.

Closely connected with solving linear systems is the task of factorizing a matrix A into a product of simpler matrices. Gaussian elimination is equivalent to the first of the three factorizations below:

$A = PLU$	P is a permutation matrix, L is a lower- and, U is an upper triangular matrix
$A = QR$	Q is unitary and R is upper triangular
$A = U\Sigma V^*$	U, V are both unitary and Σ is diagonal with nonnegative elements on its diagonal

The three factorizations go under the names of LU, QR and SVD (singular value decomposition) respectively. We discuss them in Sections 11.2—11.5, and we will see in Section 11.6 how they can be used to obtain solutions, in some sense, of *any*

$m \times n$ system of equations. Thus you will be able to solve in a systematic way, all the examples given above.

11.2. Gaussian elimination.

In its simplest form, Gaussian elimination proceeds just like the reduction to echelon form in the previous section—the main difference lies only in the fact that things now proceed in a very systematic manner. The procedure is best described by means of an example.

Solve the system

$$\begin{aligned} x_1 - 2x_2 + 2x_3 &= 1 \\ -x_1 - x_2 + 3x_3 &= 1 \\ 3x_1 + x_2 - 2x_3 &= 2 \end{aligned}$$

It is convenient to write down only the coefficients, and proceeding by removing the entries below the main diagonal in successive columns, gives,

$$\left[\begin{array}{rrr|r} \mathbf{1} & -2 & 2 & 1 \\ -1 & -1 & 3 & 1 \\ 3 & 1 & -2 & 2 \end{array}\right] \Rightarrow \left[\begin{array}{c} \text{Multiply the first row with } \boxed{\text{-1}} \text{ and subtract from row 2} \\ \text{then} \\ \text{multiply the first row with } \boxed{3} \text{ and subtract from row 3} \end{array}\right]$$

$$\Rightarrow \left[\begin{array}{rrr|r} \mathbf{1} & -2 & 2 & 1 \\ \boxed{\text{-1}} & \mathbf{-3} & 5 & 2 \\ \boxed{3} & 7 & -8 & -1 \end{array}\right]$$

Before we proceed to the next and final step, note that instead of writing the zeroes we have just created, we write the multipliers (the values inside the boxes) for future use.

The 7 in the last row is removed by multiplying the second row with $\boxed{\text{-7/3}}$ and subtract it from the third row,

$$\left[\begin{array}{rrr|r} \mathbf{1} & -2 & 2 & 1 \\ \boxed{-1} & \mathbf{-3} & 5 & 2 \\ \boxed{3} & 7 & -8 & -1 \end{array}\right] \Rightarrow \left[\begin{array}{rrr|r} \mathbf{1} & -2 & 2 & 1 \\ \boxed{-1} & \mathbf{-3} & 5 & 2 \\ \boxed{3} & \boxed{-7/3} & \mathbf{11/3} & 11/3 \end{array}\right]$$

Backsubstitution now proceeds by noting that the last equation tells us that $x_3 = 1$; knowing this, the middle equation tells us that $x_2 = 1$, and from the top equation we obtain $x_1 = 1$. It is important to note that our ability to solve the system depends crucially on the fact that none of the *pivots* indicated in boldface, is zero.

The basic elimination process is not quite satisfactory as it stands:

(1) If any of the pivots is zero, the elimination process breaks down. For example, if the first pivot, i.e. the (1,1) element, had been zero, the multiplier would have been infinite with the result that no multiple of the first row can eliminate any of the nonzero entries in the first column. Similar breakdowns can occur at any stage. If a pivot is very small, the corresponding row is multiplied by a large multiplier, and this looses accuracy in floating point arithmetic.

(2) It is quite common that one needs to solve a system with many right hand sides (RHSs). If they are all known initially, we can just place them side-by-side when we eliminate as above. But it is also common that they become known only one after another. We then do not want to repeat the work on the coefficient part.

(3) The coefficient matrix has often a lot of zero elements, or some other special structure, e.g. it may be tri-diagonal or banded. Such structures can often be exploited to greatly reduce both operations and storage.

We now address these issues in turn.

(1) Let us illustrate the problem of small pivots with a simple example,

$$\left[\begin{array}{cc} \epsilon & 1 \\ 1 & 1 \end{array}\right] \left[\begin{array}{c} x \\ y \end{array}\right] = \left[\begin{array}{c} 1 \\ 2 \end{array}\right].$$

The exact solution is easily seen to be

$$x = \frac{1}{1-\epsilon}, \quad y = \frac{1-2\epsilon}{1-\epsilon}.$$

Assuming that ϵ is machine precision, i.e. $1 \pm \epsilon \rightarrow 1$ in floating point arithmetic, the original system becomes

$$\begin{bmatrix} \epsilon & 1 \\ 0 & 1 - \frac{1}{\epsilon} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ 2 - \frac{1}{\epsilon} \end{bmatrix}.$$

Solving this in floating point arithmetic gives $y = 1, \quad x = 0$. It is easy to see where the problem occurs: had we been able to represent y as $y = 1 - \epsilon$ ($O(\epsilon^2)$ from its exact value), we get $x = 1$ ($O(\epsilon)$ from its exact value). Thus an $O(\epsilon)$ approximation in our variables, leads to an $O(1)$ change in the solution—clearly an unstable situation. If, on the other hand, we first exchange the two rows and then proceed as before we get (in floating point arithmetic),

$$\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

or $x = 1, \quad y = 1$. Now an $O(\epsilon)$ approximation of our variables leads to an $O(\epsilon)$ approximation of the result—clearly a stable situation.

Since the solution of a linear system is totally independent of the order in which the equations are written down, one can therefore interchange rows in order to ensure that the pivots are as large as possible. Thus at the k-th stage the k-th row is interchanged with the row below it with the largest element in the k-th first position. This of course implies that the magnitudes of the multipliers never exceed one—growth in intermediate quantities becomes less likely. In practice this *partial pivoting* procedure generally results in a stable procedure.

Although row interchanges occur as part of the elimination process, conceptually one can imagine that all the necessary row interchanges are performed right at the start. This is achieved by multiplying the original array by an appropriate *permutation* matrix P. For example the first and second rows of a 3×3 matrix is interchanged by a multiplication (from the left) by

$$P = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

In general a permutation matrix is obtained by interchanging the rows of the identity matrix. For example, the matrix P above is obtained by interchanging the first and second rows of the identity matrix. It should therefore be clear that that the determinant of a permutation matrix is ± 1 where the sign is positive in case of an even number and negative in case of an odd number of of row interchanges.

(2) The multipliers that we conveniently kept in the example above, allows a factorization of the original matrix,

$$A = \begin{bmatrix} 1 & -2 & 2 \\ -1 & -1 & 3 \\ 3 & 1 & -2 \end{bmatrix}$$

into a product $A = LU$ where L is lower triangular and U upper triangular. In fact both matrices are already present in the final matrix of the example. The L matrix consists of the multipliers, and U is the result of the elimination process,

$$L = \begin{bmatrix} 1 & 0 & 0 \\ \boxed{\text{-1}} & 1 & 0 \\ \boxed{3} & \boxed{\text{-7/3}} & 1 \end{bmatrix}, \quad U = \begin{bmatrix} 1 & -2 & 2 \\ 0 & -3 & 5 \\ 0 & 0 & 11/3 \end{bmatrix}.$$

The reader should verify that the product LU actually recovers A.

The reason why this happens is best understood if we realize that each elimination can be achieved by a matrix multiplication. Thus we find that

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & -2 & 2 \\ -1 & -1 & 3 \\ 3 & 1 & -2 \end{bmatrix} = \begin{bmatrix} 1 & -2 & 2 \\ 0 & -3 & 5 \\ 3 & 1 & -2 \end{bmatrix}$$

and

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -3 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & -2 & 2 \\ 0 & -3 & 5 \\ 3 & 1 & -2 \end{bmatrix} = \begin{bmatrix} 1 & -2 & 2 \\ 0 & -3 & 5 \\ 0 & 7 & -8 \end{bmatrix}.$$

Putting it all together we get

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 7/3 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -3 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & -2 & 2 \\ -1 & -1 & 3 \\ 3 & 1 & -2 \end{bmatrix} = \begin{bmatrix} 1 & -2 & 2 \\ 0 & -3 & 5 \\ 0 & 0 & 11/3 \end{bmatrix}.$$

The inverses of the matrices on the left hand side is immediate: one only changes the sign of the entry below the diagonal. Multiplying with the inverses from the left therefore gives,

$$\begin{aligned} \begin{bmatrix} 1 & -2 & 2 \\ -1 & -1 & 3 \\ 3 & 1 & -2 \end{bmatrix} &= \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 3 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -7/3 & 1 \end{bmatrix} \begin{bmatrix} 1 & -2 & 2 \\ -1 & -1 & 3 \\ 3 & 1 & -2 \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 3 & -7/3 & 1 \end{bmatrix} \begin{bmatrix} 1 & -2 & 2 \\ -1 & -1 & 3 \\ 3 & 1 & -2 \end{bmatrix} \end{aligned}$$

Note how the first three matrices on the right hand side multiply together—their product is formed simply by putting the nonzero entries together in a single matrix. Try it!

A few comments are in order.

- In the form $LU\mathbf{x} = \mathbf{b}$, the linear system is quickly solved by two successive back substitutions: First $L\mathbf{w} = \mathbf{b}$ and then $U\mathbf{x} = \mathbf{w}$.
- No additional storage is required for L and U. As we have seen in the example both can be fitted back in the memory A used to occupy.
- The LU factorization is the more expensive part of the process, requiring $O(n^3)$ operations for an $n \times n$ matrix whereas the two back

substitution steps require on $O(n^2)$ operations. Thus for multiple right hand sides, the expensive factorization is done only once.

- The determinant of A is given by the product of the diagonal elements of U. Thus A is singular if and only if one of pivots is zero.

A slight complication is introduced if partial pivoting is incorporated into the LU factorization. Again it is useful to imagine that all the row interchanges are done beforehand, resulting in a new matrix PA. The LU factorization therefore becomes $PA = LU$. Note the following:

- In practice the row interchanges are performed during the factorization process. Since this is done independently form the right hand side, it is necessary to keep a record of these interchanges in order to be able to do the same interchanges during the back substitution steps.
- It is not efficient to physically swap rows in computer memory. It turns out that it can be simulated most efficiently, and at the same time keeping track of the interchanges for use during the back substitution steps.

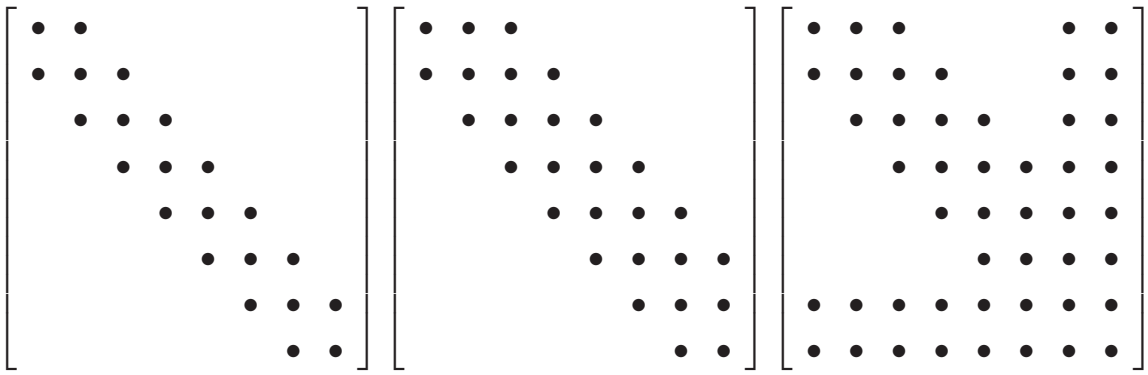
  The key idea is to reference the appropriate rows through an index vector, let us call it $\mathbf{p}$. Initially the elements of $\mathbf{p}$ are set as $\mathbf{p}(i) = i, \;\; i = 1, \ldots, n$. A record of a row interchange is kept by interchanging the same rows of $\mathbf{p}$. For example, if rows i and j are interchanged, we set $\mathbf{p}(i) = j$ and $\mathbf{p}(j) = i$. The new i-th row of A is addressed as $A(\mathbf{p}(i), :)$. Since $p(i) = j$, this actually refers to the original row j. etc.

(3) The three sparsity patterns that arise most frequently are:

a.: tri-diagonal

b.: banded, and

c.: banded with some extra full rows and or columns (or corners), as shown



Although very complex sparsity patterns often arise in applications, strategies for them fall well outside the scope of this summary. For the three cases above:

(a): If the matrix is not diagonally dominant, i.e. if

$$|a_{i,i} < |a_{i,i-1}| + |a_{i,i+1}|$$

we need to pivot. Tri-diagonal then becomes a special case of type (b)—general banded matrix. If it happens to be diagonally dominant, pivoting can be shown to be unnecessary, and the very fast *Thomas algorithm* can be used. This is equivalent to unpivoted Gaussian elimination, but is usually described differently. We aim for a standard *LU* factorization, which will be of the form,

$$\begin{bmatrix} a_1 & b_1 & & & \\ c_1 & a_2 & b_2 & & \\ & c_2 & a_3 & b_3 & \\ & & \ddots & \ddots & \ddots \\ & & c_{n-2} & a_{n-1} & b_{n-1} \\ & & & c_{n-1} & a_n \end{bmatrix} = \begin{bmatrix} 1 & & & \\ \ell_1 & 1 & & \\ & \ddots & \ddots & \\ & & \ell_{n-1} & 1 \end{bmatrix} \begin{bmatrix} d_1 & u_1 & & & \\ & d_2 & u_2 & & \\ & & \ddots & \ddots & \\ & & & d_{n-1} & u_{n-1} \\ & & & & d_n \end{bmatrix}.$$

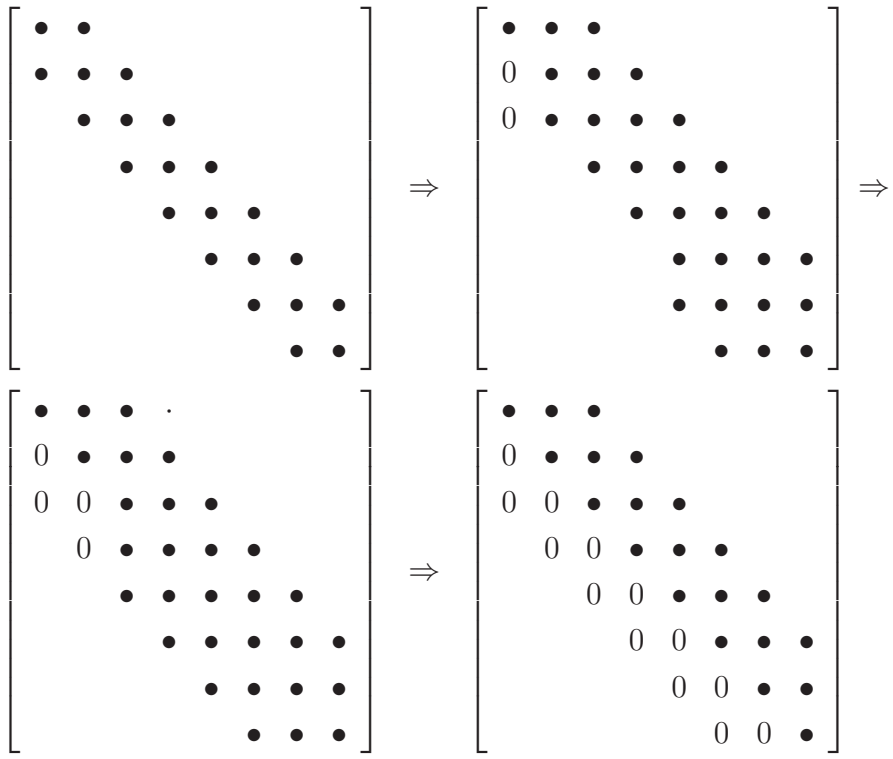
For: the right hand side to equal the left hand side, we need

$$\begin{array}{lccc} \text{row 1:} & & d_1 = a_1, & u_1 = b_1, \\ \text{row 2:} & \ell_1 d_1 = c_1, & \ell_1 u_1 + d_2 = a_2, & u_{23} = a_{23}, \\ \text{row 3:} & \ell_2 d_2 = c_2, & \ell_2 u_2 + d_3 = a_3, & u_3 = b_3, \\ & \text{etc} & & \end{array}$$

Note that the super-diagonal of A is copied into the super diagonal of U. The remaining equations give us recursively $d_1,\ \ell_1,\ d_2,\ \ell_2,\ d_3$, etc. The two back substitutions become equally quick recursions.

One can obviously save a lot of storage by just storing the diagonal elements of A in a few vectors—the LU factorization then overwrites these vectors, requiring no additional storage.

(b): In the banded case we basically have to use the same strategy as for full matrices. The only difference is that L and U (even when partial pivoting is employed) will have nonzero entries only in some central diagonals. So again it is much better to only store these diagonals (in the columns of a matrix), and confine all the arithmetic to within it. The following schematic illustrations show how the U matrix with the fill-in is generated in a case where the band is equally wide on both sides of the main diagonal, and partial (row–) pivoting is used. Removed entries are marked as "0".



The U-part will not only fit back into the matrix that used to hold the diagonals, it can directly overwrite in the space that the diagonals of A used to occupy. Had the RHS been available initially—and been manipulated with the matrix—we would now be ready for the back substitution and no additional storage would be needed. If the RHS was not available we also need to store one integer vector of pivot information, exactly as in the case of the full LU decomposition. In addition we need the L-matrix, which contains the multipliers as before. using the same diagonal-based storage as before, the matrix structures would be,

$$A = \begin{bmatrix} \bullet & \bullet & & & & & \\ \bullet & \bullet & \bullet & & & & \\ & \bullet & \bullet & \bullet & & & \\ & & \bullet & \bullet & \bullet & & \\ & & & \bullet & \bullet & \bullet & \\ & & & & \bullet & \bullet & \bullet \\ & & & & & \bullet & \bullet \end{bmatrix}, \quad L = \begin{bmatrix} 1 & & & & & & \\ \bullet & 1 & & & & & \\ \bullet & \bullet & 1 & & & & \\ & \bullet & \bullet & 1 & & & \\ & & \bullet & \bullet & 1 & & \\ & & & \bullet & \bullet & 1 & \\ & & & & \bullet & \bullet & 1 \end{bmatrix},$$

$$U = \begin{bmatrix} \bullet & \bullet & \bullet & & & & \\ & \bullet & \bullet & \bullet & & & \\ & & \bullet & \bullet & \bullet & & \\ & & & \bullet & \bullet & \bullet & \\ & & & & \bullet & \bullet & \bullet \\ & & & & & \bullet & \bullet \\ & & & & & & \bullet \end{bmatrix}, \quad P = \begin{bmatrix} \bullet \\ \bullet \\ \bullet \\ \bullet \\ \bullet \\ \bullet \\ \bullet \end{bmatrix}$$

With the information in L, U and P we can rapidly—and repeatedly—solve the linear system $A\mathbf{x} = \mathbf{b}$ whenever a RHS vector $\mathbf{b}$ becomes available.

(c): If one happens to have available solvers for general full and banded linear systems, one can solve this case of banded with extra rows/columns quickly. Schematically we write the system to solve as (assuming for simplicity that we have the RHS available from the start)

$$\left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right] \begin{bmatrix} \mathbf{X}_1 \\ \mathbf{X}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix}$$

First we solve (by a band solver) the many-RHS system

$$\left[\begin{array}{c} A \end{array}\right] \left[\begin{array}{c|c} E & \mathbf{b}_3 \end{array}\right] = \left[\begin{array}{c|c} B & \mathbf{b}_1 \end{array}\right].$$

The original system is now equivalent to

$$\left[\begin{array}{c|c} I & E \\ \hline C & D \end{array}\right] \left[\begin{array}{c} \mathbf{X}_1 \\ \mathbf{X}_2 \end{array}\right] = \left[\begin{array}{c} \mathbf{b}_3 \\ \mathbf{b}_2 \end{array}\right]$$

where I is the identity matrix. This holds because we could have achieved the same thing (more costly) by multiplying the original system from the left by

$$\left[\begin{array}{c|c} A^{-1} & 0 \\ \hline 0 & I \end{array}\right].$$

Now we can easily—and explicitly—add multiples of the top rows to clear out C, while turning $\mathbf{b}_2$ into $\mathbf{b}_4$:

$$\left[\begin{array}{c|c} I & E \\ \hline 0 & D \end{array}\right] \left[\begin{array}{c} \mathbf{X}_1 \\ \mathbf{X}_2 \end{array}\right] = \left[\begin{array}{c} \mathbf{b}_3 \\ \mathbf{b}_4 \end{array}\right]$$

Next we solve the square system (with one RHS)

$$\left[\begin{array}{c} D \end{array}\right] \left[\begin{array}{c} \mathbf{X}_2 \end{array}\right] = \left[\begin{array}{c} \mathbf{b}_4 \end{array}\right]$$

which has separated out at the bottom, independently of the rest of the equations. Knowing $\mathbf{X}_2$, we find the remaining unknowns $\mathbf{X}_1$ through

$$\begin{bmatrix} \\ \mathbf{X}_1 \\ \\ \end{bmatrix} = \begin{bmatrix} \\ \mathbf{b}_3 \\ \\ \end{bmatrix} - \begin{bmatrix} \\ E \\ \\ \end{bmatrix} \begin{bmatrix} \\ \mathbf{X}_2 \\ \\ \end{bmatrix}$$

11.3. QR factorization—Householder matrices.

One way to interpret Gaussian elimination is to note that it can be used to split a matrix A into a product $A = PLU$. The system $Ax = b$ then takes the form $PLUx = b$, which is quickly solvable in three steps: $Px_1 = b$, $Lx_2 = x_1$, $Ux = x_2$.

Another very useful factorization of A which not only also works for solving $Ax = b$, but has many other uses as well, is $A = QR$ where Q is unitary and R is upper triangular. R has the same structure as U in the PLU-factorization; it just happens to be traditional to call it R in the QR context.

Definition: Q is called *unitary* if it satisfies $QQ^* = Q^*Q = I$ (Here, I is the identity matrix, and the star denotes transpose and conjugate). Q is called *hermitian* if it satisfies $Q = Q^*$.

□

Some notable properties of unitary matrices are

(1) Q_1 and Q_2 unitary $\Longrightarrow$ Q_1Q_2 unitary.
(2) $\|Q\,x\| = \|x\|$ for any vector x.
(3) A unitary, hermitian matrix is its own inverse, i.e. if QQ^*I and $Q = Q^*$ then $Q^2 = I$, or $Q = Q^{-1}$.

The most effective way to achieve a QR factorization of a matrix A is by multiplying A with a suitable series of *Householder matrices* H.

Definition: A Householder matrix H is a matrix of the form $H = I - 2\omega\omega^*$. Here ω is a column vector satisfying $\omega^*\omega = 1$ (i.e. ω is of length one).

□

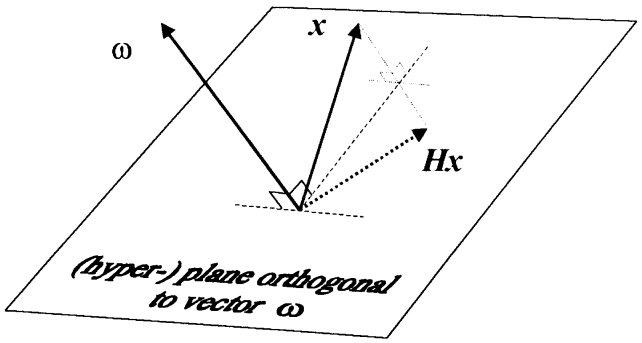


FIGURE 11.3.1. Illustration of how applying H to a vector x causes it to be reflected in the (hyper-) plane orthogonal to ω.

Notable properties of H-matrices include

(1) H is Hermitian, since $H^* = I^* - 2(\omega\omega^*)^* = I - 2\omega\omega^* = H$;
(2) H is unitary, since $H^*H = HH^* = (I - 2\omega\omega^*)(I - 2\omega\omega^*) = I - 4\omega\omega^* + 4(\omega\omega^*)(\omega\omega^*) = I - 4\omega\omega^* + 4\omega(\omega^*\omega)\omega^* = I$;
(3) H is its own inverse ($H^{-1} = H$), and it satisfies $H^2 = I$;
    (a) $H\omega = -\omega$, since $H\omega = (I - 2\omega\omega^*)\omega = \omega - 2\omega = -\omega$.
    (b) $H\mathbf{v} = \mathbf{v}$ if v is orthogonal to ω, since $H\mathbf{v} = (I - 2\omega\omega^*)\mathbf{v} = \mathbf{v} - 0\omega = \mathbf{v}$.

The last observations (under item 4) show that the effect of applying H to any vector $\mathbf{x}$ is to change the sign of the component of x which falls in the same direction as ω, but otherwise leave x unchanged, i.e. H reflects $\mathbf{x}$ about the plane orthogonal to ω. Figure 11.3.1 illustrates this in a space of 3-D vectors. From this observation, we can conclude that if we have any two vectors $\mathbf{x}$ and $\mathbf{y}$ of equal length, we can pick ω in the direction of $\mathbf{x} - \mathbf{y}$ to get $\mathbf{y} = H\mathbf{x}$. We will soon see that this ability to find an H-matrix that takes any $\mathbf{x}$ into any $\mathbf{y}$ (with two restrictions noted below) is at the core of almost all applications of H-matrices. We need to state this result in a way that does not rely on just geometric intuition in 3-D:

Theorem: If two vectors $\mathbf{x}$ and $\mathbf{y}$ satisfy

$$\text{(11.1)} \qquad \begin{cases} \mathbf{x}^*\mathbf{x} = \mathbf{y}^*\mathbf{y} \\ \mathbf{x}^*\mathbf{y} \text{ real} \qquad \text{(automatically satisfied if } \mathbf{x} \text{ and } \mathbf{y} \text{ are real} \end{cases}$$

then we can find an ω and with it a corresponding H, such that $H\mathbf{x} = \mathbf{y}$.

Proof: (outline): In order to find an expression for H, we first note, after expanding H as $I - 2\omega\omega^*$, that $H\mathbf{x} = \mathbf{y}$ implies

$$\omega = \frac{\mathbf{x} - \mathbf{y}}{2\omega^*\mathbf{x}}$$

Since the denominator is a scalar, ω must be a vector in the same direction as $\mathbf{x} - \mathbf{y}$. We also know that ω is a unit length vector. So the only two possible choices are therefore

$$\text{(11.2)} \qquad \omega = \pm\frac{\mathbf{x} - \mathbf{y}}{\|\mathbf{x} - \mathbf{y}\|}$$

Next step is to show that both of these choices indeed work. Using either of these choices for ω, we form $H = I - 2\omega\omega^*$ and multiply out $H\mathbf{x}$. With help of both parts of (11.1) this will in a few lines simplify down to $\mathbf{y}$. (This sounds really simple, but the algebra is a little bit tricky, so we leave it out here). □

We note that both conditions in (11.1) indeed are required. If $H\mathbf{x} = \mathbf{y}$ is to hold, then

(1) $\mathbf{y}^*\mathbf{y} = (H\mathbf{x})^*(H\mathbf{x}) = \mathbf{x}^*H^*H\mathbf{x} = \mathbf{x}^*\mathbf{x}$,
(2) $\mathbf{x}^*\mathbf{y} = \mathbf{x}^*H\mathbf{x} = \mathbf{x}^*H^*\mathbf{x} = \mathbf{y}^*\mathbf{x} = (\mathbf{x}^*\mathbf{y})^*$. The scalar quantity $\mathbf{x}^*\mathbf{y}$ is therefore unchanged when it is complex conjugated, i.e. it must be real.

Implementation of QR factorization: With these H-matrices, we have just the tool we need to factorize A into $A = QR$. Looking only at the first column of A, we can find H_1 so that

$$\begin{bmatrix} \bullet & \bullet & \bullet & \cdots & \bullet \\ \bullet & & & & \bullet \\ \bullet & & H_1 & & \bullet \\ \vdots & & & & \vdots \\ \bullet & \bullet & \bullet & \cdots & \bullet \end{bmatrix} \begin{bmatrix} \bullet & \bullet & \bullet & \cdots & \bullet \\ \bullet & & & & \bullet \\ \bullet & & A & & \bullet \\ \vdots & & & & \vdots \\ \bullet & \bullet & \bullet & \cdots & \bullet \end{bmatrix} = \begin{bmatrix} \pm\|\mathbf{a}_1\| & \bullet & \bullet & \cdots & \bullet \\ 0 & \bullet & & & \bullet \\ 0 & \bullet & A_1 & & \bullet \\ \vdots & \vdots & & & \vdots \\ 0 & \bullet & \bullet & \cdots & \bullet \end{bmatrix}$$

where $\mathbf{a}_1$ is the first column of A. Note that it is best to choose the opposite sign to that of a_{11}—then no risk of cancellations in the denominator of (11.2). Next we find H_2 (of one dimension less) so that

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & \bullet & \bullet & & \bullet \\ 0 & \bullet & H_2 & & \bullet \\ \vdots & \vdots & & & \vdots \\ 0 & \bullet & \bullet & \cdots & \bullet \end{bmatrix} \begin{bmatrix} \bullet & \bullet & \bullet & \cdots & \bullet \\ 0 & \bullet & & & \bullet \\ 0 & \bullet & A_1 & & \bullet \\ \vdots & \vdots & & & \vdots \\ 0 & \bullet & \bullet & \cdots & \bullet \end{bmatrix} = \begin{bmatrix} \bullet & \bullet & \bullet & \cdots & \bullet \\ 0 & \bullet & \bullet & & \bullet \\ 0 & 0 & \bullet & A_2 & \bullet \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & 0 & \bullet & \cdots & \bullet \end{bmatrix}$$

etc. By multiplying A successively from the left with this sequence of unitary matrices, the result has become upper triangular. Writing this as $Q^*A = R$, we get $A = QR$.

If A is an $n \times n$ matrix, the leading order operation count for factorizing $A = QR$ becomes $\frac{4}{3}N^3$ operations - twice as expensive as an $A = PLU$ factorization by Gaussian elimination. However the cost is only half of a QR factorization based on *Givens rotations*, and the same as *fast Givens* (the main alternatives—not discussed here).

11.4. Rotations.

In this section we describe a very useful application of unitary matrices. Consider the motion of a rigid object. It can really do only one, or both, of two things: it can translate and it can rotate. Translation of the object is describe by a simple position vector. In order to describe its rotation we need to develop the necessary tools.

Let Q be a real 3×3 matrix (we are only interested in $3D$ rotations of real objects). Since Q maps any vector $\mathbf{x}$ to a vector $\mathbf{y} = Q\mathbf{x}$, the question is when does Q describe a rotation? It should be clear that a good way of characterizing a rotation is that the length of vectors should remain unchanged, i.e. we need $\mathbf{x}^T\mathbf{x} = \mathbf{y}^T\mathbf{y}$, for all vectors $\mathbf{x}$. This means that

$$\begin{aligned} \mathbf{x}^T\mathbf{x} &= \mathbf{y}^T\mathbf{y} \\ &= \mathbf{x}^T Q^T Q\mathbf{x}, \end{aligned}$$

for all $\mathbf{x}$. It therefore follows that $Q^TQ = I = QQ^T$, i.e. that Q be *orthogonal*.

Since $\det(Q^TQ) = \det(Q)^2 = \det(I) = 1$, it follows that general rotation matrices have $\det(Q) = \pm 1$. If we now think of a rotation matrix that depends on a parameter, the rotation angle θ, $\det(Q)$ cannot changes sign by a smooth change in the angle of rotation. Therefore $\det(Q) = \pm 1$ really divides the rotation matrices into two separate classes. Let us look at an example.

Example: Let

$$Q = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

If $\mathbf{x} = [1 \;\; 0 \;\; 0]^T$ then $Q\mathbf{x} = [\cos\theta \;\; \sin\theta \;\; 0]$, i.e. Q describes a rotation anti-clockwise around the z-axis through an angle θ. A simple calculation shows that $\det(Q) = 1$. □

Let us look at another example.

Example: Let Q be one of the Householder matrices of the previous section,

$$Q = I - 2\omega\omega^T$$

with $\omega^T\omega = 1$. Recall (or quickly verify again) that $Q^TQ = I = QQ^T$, i.e. Q is a rotation matrix in the sense of our earlier definition that it preserves lengths of vectors. However, we saw that Householder matrices reflect vectors over the plane orthogonal to ω. Moreover, in this case $\det(Q) = -1$. (Why?) □

From these examples it should be clear that the lengths of vectors are also preserved by reflections, characterized by $\det(Q) = -1$. And our argument above showed that one cannot produce a reflection by a pure rotation. Henceforth, pure rotations with $\det(Q) = 1$ will be referred to simply as 'rotations'. The case $\det(Q) = -1$ is similarly referred to as 'reflections'.

Physically a pure rotation in 3D is described by an angle θ and an axis of rotation $\mathbf{a} = [a_1 \;\; a_2 \;\; a_3]^T$, normalized since we are only interested in the direction of $\mathbf{a}$, i.e. $\mathbf{a}^T\mathbf{a} = 1$. Thus we effectively need three parameters in order to completely describe a rotation, the angle, and two parameters describing the direction of the axis of rotation. We need to know how to do two things: First, given an angle θ and axis of rotation, $\mathbf{a}$, construct Q. Secondly, given Q, extract θ and $\mathbf{a}$.

Instead of rotating a vector through an angle θ around an axis $\mathbf{a}$, imagine that we rotate the (Euclidean) coordinate axes around the same axis but through $-\theta$. This means that *all* vectors keep their length. More precisely, for any vector $\mathbf{x}$ depending on θ we have that $\mathbf{x}^T(\theta)\mathbf{x}(\theta) = \text{const}$. Differentiation gives

$$\mathbf{x}^T(\theta)\mathbf{x}'(\theta) = 0, \tag{11.1}$$

where $'$ means the derivative with respect to θ. We leave it as an exercise to show that this equation is satisfied for all vectors $\mathbf{x}$ if and only if

$$\mathbf{x}' = S\mathbf{x}, \tag{11.2}$$

where S is an anti-symmetric matrix

$$S = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix},$$

with $a_1^2 + a_2^2 + a_3^2 = 1$.

Note that, apart from the scaling, S is just the most general 3×3 anti-symmetric matrix; its identification as the axis of rotation needs justification. Indeed, note that the cross product $\mathbf{a} \times \mathbf{x}$ can be written as

$$\mathbf{a} \times \mathbf{x} = S\mathbf{x}$$

with S given as above. This has a neat physical interpretation: $\mathbf{x}' = \mathbf{a} \times \mathbf{x}$ points in the direction orthogonal to both $\mathbf{a}$ *and* $\mathbf{x}$. Since S, hence $\mathbf{a}$ remains constant during the rotation, $\mathbf{x}(\theta)$ rotates in a plane orthogonal to $\mathbf{a}$ which defines the axis of rotation.

Solving (11.2) gives,

$$\mathbf{x}(\theta) = \exp(S\theta)\mathbf{x}(0).$$

Thus the rotation matrix Q is written as

$$Q = \exp(S\theta), \tag{11.3}$$

and it only remains to find a closed-from expression for Q. A simple calculation shows that $S^3 = -S$ which enables us to write

$$\begin{aligned} Q &= \exp(S\theta) \\ &= I + S\theta + \frac{1}{2}S^2\theta^2 + \frac{1}{3!}S^3\theta^3 + \frac{1}{4!}S^4\theta^4 + \cdots \\ &= I + S\theta + \frac{1}{2}S^2\theta^2 - \frac{1}{3!}S\theta^3 - \frac{1}{4!}S^2\theta^4 + \cdots \\ &= I + S(1 - \frac{1}{3!}\theta^3 + \cdots) + S^2(\frac{1}{2}\theta^2 - \frac{1}{4!}\theta^4 + \cdots). \end{aligned}$$

It therefore follows that

$$Q = I + S\sin\theta + S^2(1 - \cos\theta). \tag{11.4}$$

Before we get to the meaning of $\mathbf{a}$ and θ in (11.3) and (11.4), let us look at the properties of Q. From (11.3) it follows that Q is indeed a rotation matrix. We only need to observe that

$$\begin{aligned} Q^T &= \exp(S^T\theta) \\ &= \exp(-S\theta). \end{aligned}$$

Emphasizing Q's dependence on θ by writing $Q(\theta)$ we also note that $Q^T(\theta) = Q(-\theta)$ so that $Q(\theta)Q(-\theta) = I$. This means that any rotation of a vector by θ is undone by a rotation in the opposite direction. In order to make this even clearer, suppose $\mathbf{a} = [0 \;\; 0 \;\; 1]^T$. From (11.4) follows that,

$$Q = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Thus Q is a rotation anti-clockwise through θ degrees around the z-axis. We therefore make the identification that θ is indeed the angle of rotation.

Note that the axis of rotation is the only vector that remains unchanged under a rotation, i.e. the axis of rotation is the eigenvector of the eigenvalue 1. It is easy to verify that $S\mathbf{a} = \mathbf{0}$ so that $Q\mathbf{a} = \mathbf{a}$. We therefore identify $\mathbf{a}$ with the axis of rotation.

One of our two objectives has been achieved: Given the angle θ and axis $\mathbf{a}$ of rotation, the corresponding rotation matrix is given by (11.4).

The remaining question, given Q find θ and $\mathbf{a}$, can be answered by noting that

$$Q - Q^T = 2\sin\theta S. \tag{11.5}$$

Since the left-hand side is known, the right-hand side gives the un-normalized $\mathbf{a}$ with the normalization factor equal to $2\sin\theta$. Thus both $\mathbf{a}$ and θ are calculated from (11.5). Note that there is no ambiguity regarding the sign of the normalization constant: a rotation clockwise through θ around $\mathbf{a}$ is the same as a rotation through $-\theta$ around $-\mathbf{a}$, and vise versa. There is however, an ambiguity between θ and $\pi - \theta$. This is resolved by noting that

$$\text{trace}(Q) = 1 + 2\cos\theta. \tag{11.6}$$

Example: For

$$Q = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{6}} \\ 0 & \frac{1}{\sqrt{3}} & \frac{2}{\sqrt{6}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{3}} & \frac{1}{\sqrt{6}} \end{bmatrix},$$

first check that $QQ^T = 1$ and $\det(Q) = 1$. It now follows that

$$\begin{aligned} Q - Q^T &= \begin{bmatrix} 0 & \frac{1}{\sqrt{3}} & -\frac{1}{\sqrt{6}} - \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{3}} & 0 & \frac{2}{\sqrt{6}} + \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{2}} + \frac{1}{\sqrt{6}} & -\frac{2}{\sqrt{6}} - \frac{1}{\sqrt{3}} & 0 \end{bmatrix} \\ &= 2\sin\theta \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}. \end{aligned}$$

We therefore find that

$$2\sin\theta\mathbf{a} = \begin{bmatrix} -\frac{2}{\sqrt{6}} - \frac{1}{\sqrt{3}} \\ -\frac{1}{\sqrt{6}} - \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{3}} \end{bmatrix}.$$

Since $\mathbf{a}$ is normalized, it follows that

$$4\sin^2\theta = \left(\frac{2}{\sqrt{6}} + \frac{1}{\sqrt{3}}\right)^2 + \left(\frac{1}{\sqrt{6}} + \frac{1}{\sqrt{2}}\right)^2 + \left(\frac{1}{\sqrt{3}}\right)^2$$

so that $\sin\theta = 0.9381$. In addition,

$$\begin{aligned} \text{trace}(Q) &= (\sqrt{3} + \sqrt{2} + 1)/\sqrt{6} \\ &= 1 + 2\cos\theta. \end{aligned}$$

so that $\cos\theta = 0.3464$ and $\theta = 1.2171$. □

Example: Since the product of two orthogonal matrices is again orthogonal, the result of two consecutive rotations can be written as a single rotation. Suppose we first rotate on object around the z-axis though 45° and then around the x-axis through 60°. We want to write it as a single rotation, around a yet to be determined axis.

For the first rotation, $a_1 = [0 \;\; 0 \;\; 1]^T$ with

$$S_1 = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad S_1^2 = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

so that

$$\begin{aligned} Q_1 &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} + \sin\theta_1 \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} + (1 - \cos\theta_1) \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \\ &= \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \sin\theta_1 & \cos\theta_1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \end{aligned}$$

For the second rotation it follows similarly that,

$$Q_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta_2 & -\sin\theta_2 \\ 0 & \sin\theta_2 & \cos\theta_2 \end{bmatrix}.$$

The product of the two rotations (note the order!), is given by

$$\begin{aligned} Q &= Q_2 Q_1 \\ &= \begin{bmatrix} \cos\theta_1 & -\sin\theta_1 & 0 \\ \cos\theta_2\sin\theta_1 & \cos\theta_2\cos\theta_1 & -\sin\theta_2 \\ \sin\theta_1\sin\theta_2 & \cos\theta_1\sin\theta_2 & \cos\theta_2 \end{bmatrix}. \end{aligned}$$

We now extract the axis– and angle of rotation,

$$Q - Q^T = \begin{bmatrix} 0 & -\sin\theta_1(1+\cos\theta_2) & -\sin\theta_1\sin\theta_2 \\ \sin\theta_1(1+\cos\theta_2) & 0 & -\sin\theta_2(1+\cos\theta_1) \\ \sin\theta_1\sin\theta_2 & \sin\theta_2(1+\cos\theta_1) & 0 \end{bmatrix}.$$

Thus the new axis of rotation is,

$$2\sin\theta\mathbf{a} = \begin{bmatrix} \sin\theta_2(1+\cos\theta_1) \\ -\sin\theta_1\sin\theta_2 \\ \sin\theta_1(1+\cos\theta_2) \end{bmatrix},$$

with

$$2\sin\theta = \sqrt{(\sin\theta_2(1+\cos\theta_1))^2 + (\sin\theta_1\sin\theta_2)^2 + (\sin\theta_1(1+\cos\theta_2))^2}.$$

For $\theta_1 = \pi/4$ and $\theta_2 = \pi/3$ it follows that $\sin\theta = 0.9599$ and

$$\mathbf{a} = \begin{bmatrix} 0.7701 \\ -0.3190 \\ 0.5525 \end{bmatrix}.$$

Since

$$\begin{aligned} \text{trace}(Q) &= \cos\theta_1 + \cos\theta_2 + \cos\theta_1\cos\theta_2 \\ &= (3+\sqrt{2})/2\sqrt{2}, \end{aligned}$$

and $\cos\theta = 0.2803$, it follows that $\theta = 1.2867$. □

Following this example we note that any number of rotations can be written as a single rotation. Although a single reflection cannot be written as a rotation, any *even* number of reflections can be written as a single rotation (why?).

11.5. Singular Value Decomposition (SVD.)

The Singular Value Decomposition (or SVD for short) extracts, in a remarkably clear form, a lot of information about an arbitrary matrix A. In the context of the applications earlier in this book we needed, for the facial recognition problem discussed in Chapter 2, to find an orthonormal basis for the subspace spanned by a set of vectors. Although this can also be achieved by the QR factorization—the columns of Q in the QR factorization provide an orthonormal basis for the column space of A— we will soon see that the SVD provides even more useful information about A.

11.5.1. Definition of the SVD . The Singular Value Decomposition (SVD) of an $m \times n$ matrix A is given by

$$A = U\Sigma V^* \tag{11.1}$$

where U and V are unitary matrices, of sizes $m \times m$ and $n \times n$ respectively. Σ is a diagonal $m \times n$ matrix containing the *singular values* of A.

For example, if $m = 3$ and $n = 2$ then

$$\underset{A}{\begin{bmatrix} \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \end{bmatrix}} = \underset{U}{\begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix}} \underset{\Sigma}{\begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \\ 0 & 0 \end{bmatrix}} \underset{V^*}{\begin{bmatrix} \cdot & \cdot \\ \cdot & \cdot \end{bmatrix}}.$$

The singular values are (by convention) ordered in non-increasing order of magnitude, so that $\sigma_j \leq \sigma_i$ if $j > i$. There are exactly $s = \min(m, n)$ singular values, $\sigma_j, \quad j = 1, \ldots, s$, some of which may be zero. The number r of nonzero singular values is called the *rank* of A (i.e. $\sigma_j = 0, \quad j = r + 1, \ldots, s$). If $r = s$ then A is said to be of full rank.

11.5.2. Computation of the SVD. Expressions for the different factors in the SVD are easily obtained. Multiplying (11.1) from the right and left respectively with A^*, we obtain,

$$AA^*U = U\Sigma\Sigma^* \text{ and } A^*AV = V\Sigma^*\Sigma. \tag{11.2}$$

Thus the columns of U are the eigenvectors of the symmetric matrix AA^*, and similarly for V. The singular values are the positive square roots of the eigenvalues of AA^* or A^*A (we'll be more precise in a moment). Note that since both AA^* and A^*A are symmetric, their eigenvalues are always real. We leave it as an exercise to show that the eigenvalues are nonnegative.

There is one slight complication however—one has to be careful in the choice of the *sign* of the eigenfunctions. The relevant equation is straightforward to derive,

$$A\mathbf{v}_j = \sigma_j \mathbf{u}_j, \quad j = 1, \ldots, s, \tag{11.3}$$

where we have written

$$U = \begin{bmatrix} \mathbf{u}_1 & \cdots & \mathbf{u}_m \end{bmatrix} \text{ and } V = \begin{bmatrix} \mathbf{v}_1 & \cdots & \mathbf{v}_n \end{bmatrix}.$$

From (11.3) follows that one is free to choose the sign of either $\mathbf{v}_j$ *or* $\mathbf{u}_j$ but not both. Once the sign of, say $\mathbf{v}_j$ is chosen, the sign of $\mathbf{u}_j$ is determined by (11.3).

Let us now consider the situation where $n > m$ in more detail, leaving the case where $m > n$ to the reader. Since $n > m$, A^*A has more eigenvalues than AA^*, n and m respectively. Since we have exactly m singular values, A^*A has more eigenvalues than singular values, and the statement above that the singular values are the positive square roots of the eigenvalues of A^*A is therefore not strictly true. The remaining $n - m$ eigenvalues are all zero, since A^*A is the product of two matrices of rank no larger than m. The corresponding eigenvectors that go into the last $m - n$ columns of V, satisfy

$$A\mathbf{v}_j = \mathbf{0}, \quad j = n + 1, \ldots, m,$$

and therefore form an orthogonal basis of the null space of A, as explained in more detail below.

Although the formulas (11.2) allow us to calculate the SVD by calculating the eigenvalues and eigenvector of symmetric matrices, there is a loss of information due to rounding errors when A^*A or A^*A is formed (see [**?, ?, ?**]). This is easily demonstrated, using an example from [**?**].

Example: Consider

$$A = \begin{bmatrix} 1 & 1 \\ 0 & \sqrt{\eta} \end{bmatrix}$$

where $\eta = \epsilon/2$ and ϵ denotes machine precision, i.e. $1 + \eta$ rounds to 1. Its singular values are close to $\sqrt{2}$ and $\sqrt{\eta/2}$. Since

$$A^T A = \begin{bmatrix} 1 & 1 \\ 1 & 1+\eta \end{bmatrix}$$

it rounds to

$$\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix},$$

with singular values given by $\sqrt{2}$ and 0. Thus, rounding $1 + \eta$ to 1, changes the smaller singular value from $\sqrt{\eta/2} = \sqrt{\epsilon}/2$ to 0. This represents a considerable loss in significant digits in the small singular value. Note that it is the smallest singular values that are affected the most, therefore, for applications such as facial recognition where we ignore small singular values, this usually does not cause any problems.

□

For more information how to overcome this problem and calculate the SVD numerically, the reader is referred to [**?, ?**]. We also provide a little more detail in Section 3.3 for situations where $m >> n$.

11.5.3. Reduced form of the SVD. Because of the many zeros in Σ, a number of columns of U or rows in V^*—depending on the shape of A—may be redundant. If for example $m > n$ and A is of full rank, the areas inside the dashed lines in Figure 11.5.1(a) can always be removed without any loss of information. If $r < s$,

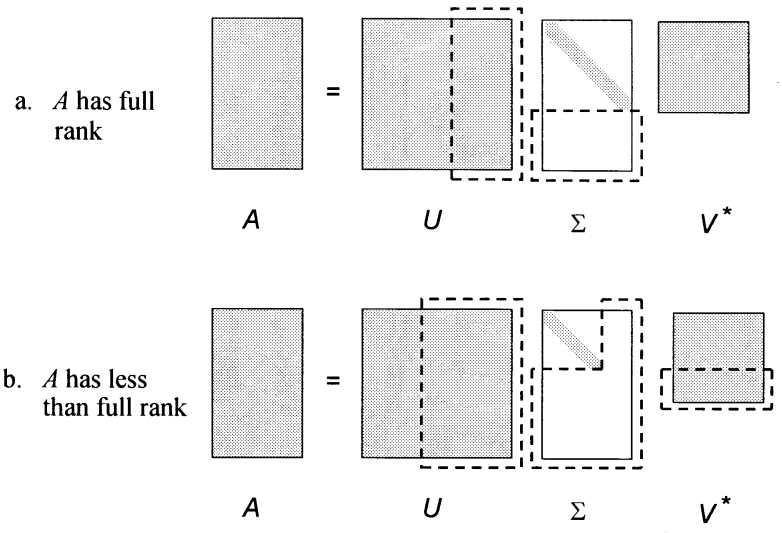


FIGURE 11.5.1. Regular and reduced forms of the SVD decomposition of a matrix. For the reduced form, omit areas inside the dashed boxes.

i.e. A is not of full rank, even more rows and columns can be removed, as illustrated in Figure 11.5.1(b). This gives the *reduced form* of the SVD.

We can formally state this as

$$A = \widehat{U}\,\Sigma_+\widehat{V}^* \tag{11.4}$$

where $\Sigma_+ = \mathrm{diag}(\sigma_1 \cdots \sigma_r)$, $\widehat{U}$ is an $m \times r$ matrix consisting of the first r columns of U and $\widehat{V}$ is a $n \times r$ matrix consisting of the first r columns of V.

From the reduced form follows that we can write A as a sum of r rank one matrices

$$A = \sum_{j=1}^{r} \sigma_j \mathbf{u}_j \mathbf{v}_j^*, \tag{11.5}$$

where the $\mathbf{u}_j$ and $\mathbf{v}_j$ are the columns of U and V respectively. In many cases, the σ_j go to zero quickly. Truncating the sum in (11.5) by discarding terms with small singular values, offers an effective means of data compression.

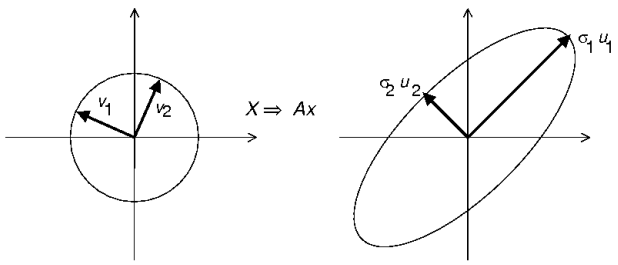


FIGURE 11.5.2. Sample illustration of the range of Ax when A is a 2×2 matrix and x is a unit length vector.

11.5.4. Geometric interpretation of the SVD. *The unit sphere* $\mathbf{x}$ (*all vectors with* $\|\mathbf{x}\| = 1$) *maps through* $\mathbf{y} = A\mathbf{x}$ *onto a hyper-ellipse . In particular, the* (*orthogonal*) *unit vectors* $\mathbf{v}_j$ *map into the (also orthogonal) semi-axes* $\sigma_j \mathbf{u}_j$ *of this hyper-ellipse.*

The Figure 11.5.2 illustrates this in the case when A is a 2×2 matrix.

A consequence of this result is that the largest singular value provides us with some sort of measure of the 'size' of A. The L^2 norm of the matrix A is defined as

$$\|A\|_2 = \sup_{\|\mathbf{x}\|_2 = 1} \|A\mathbf{x}\|_2 \tag{11.6}$$

where the L^2 norm of the vector on the right hand side is the usual Euclidean distance, i.e.

$$\|\mathbf{x}\|_2 = \sqrt{x_1^2 + \cdots + x_n^2}.$$

It follows from the principle above that

$$\|A\|_2 = \sigma_1. \tag{11.7}$$

This can also be demonstrated algebraically as follows: Since A^*A is hermitian it can be diagonalized by a matrix S, such that $S^*A^*AS = \Lambda := \mathrm{diag}(\lambda_j)$ where the λ_j are the eigenvalues of A^*A. Since A^*A is a semi- positive definite matrix, its eigenvalues are all non-negative. Thus we find that

$$\begin{aligned}\|A\|_2 &= \sup_{\|\mathbf{x}\|_2=1} (\mathbf{x}^* A^* A \mathbf{x})^{1/2} = \sup_{\|\mathbf{x}\|_2=1} (\mathbf{x}^* S^* \Lambda S \mathbf{x})^{1/2} \\ &= \sup_{\|\mathbf{x}\|_2=1} ((S\mathbf{x})^* \Lambda (S\mathbf{x}))^{1/2} = \sup_{\|\mathbf{y}\|_2=1} (\mathbf{y}^* \Lambda \mathbf{y})^{1/2} = \sup_{\|\mathbf{y}\|_2=1} \sqrt{\sum \lambda_j |y_j|^2} \\ &\leq \sup_{\|\mathbf{y}\|_2=1} \sqrt{\lambda_{\max} \sum |y_j|^2} = \sqrt{\lambda_{\max}} = \sigma_1,\end{aligned}$$

which is attainable by choosing $\mathbf{y}$ to be the appropriate column of the identity matrix.

11.5.5. Some theoretical consequences of the SVD decomposition. Let us first recall the following fundamental subspaces related to a matrix A:

- The column space of A is spanned by its columns,
- The row-space of A is spanned by its rows,
- The null-space of A is spanned by all vectors $\mathbf{x}$ such that $A\mathbf{x} = \mathbf{0}$,
- The left null-space of A is spanned by all vectors $\mathbf{y}$ such that $\mathbf{y}^* A = \mathbf{0}$. (Note that $\mathbf{y}^*$ is a row vector.)

We then have:

THEOREM 26. *If there are exactly r nonzero singular values with $r \leq s$ then the first r columns of U form an orthonormal basis for the column space of A, the first r columns of V form an orthonormal basis for the row space of A, the last $m - r$ columns of U form an orthonormal basis for the left null space of A, the last $n - r$ columns of V form an orthonormal basis for the null space of A.*

Proof: From (11.4) follows that every column of A can be written as a linear combination of the first r columns U, thus the column space of A is a subspace of the space spanned by the first r columns of U. Conversely, from (11.4) also follows that every one of the first r columns of U is a linear combination of the columns of A, implying that the first r columns of U lie in the column space of A. Thus (i) is established. A direct calculation shows that $A\mathbf{v} = \mathbf{0}$ where $\mathbf{v}$ is any one of the last $n - r$ columns of V. Thus they all lie in the null space of A. Conversely,

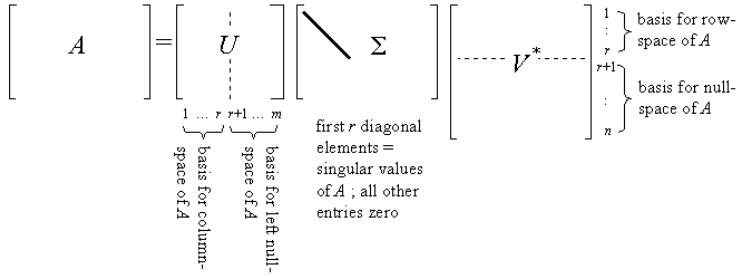


FIGURE 11.5.3. An illustration of the bases of the four fundamental subspaces

suppose $A\mathbf{x} = \mathbf{0}$. It follows from the SVD (11.1) that $\Sigma V^*\mathbf{x} = \mathbf{0}$, implying that $\mathbf{x}$ is orthogonal to the first r columns of V. Thus $\mathbf{x}$ lies in the space spanned by the last $m - r$ columns of the unitary matrix V and (iv) is proved. (ii) and (iii) follow by similar arguments (simply consider A^* instead of A). $\square$

This theorem is illustrated graphically in Figure 11.5.3.

An immediate and important consequence of this theorem is that the column space of A is orthogonal to its left null space, and the row space is orthogonal to the null space. The orthogonality of the four spaces is illustrated in Figure 11.5.4. Note how the matrix A always maps any vector $\mathbf{x}$ into its column space, $A\mathbf{x}$.

In practice experimental and numerical errors usually prevent the presence of singular values that are exactly zero. The following theorem states that a good approximation of A is obtained if the singular values close to zero are ignored. This turns out to be the best possible approximation. In fact, the best approximation of the hyper-ellipse, mentioned at the beginning of this subsection, by a one-dimensional ellipse is the line with the longest axis. The best approximation by a two-dimensional ellipse is obtained by taking the ellipsoid spanned by the longest and second longest axes, and so on. The following theorem makes this more precise.

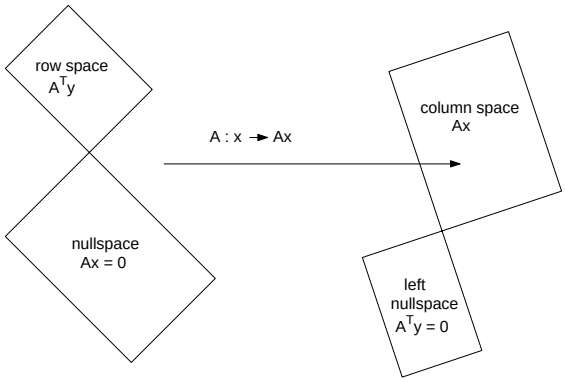


FIGURE 11.5.4. The orthogonality of the four fundamental subspaces

THEOREM 27. *If A_ν is the approximation of A obtained by keeping the first ν terms in (11.5),*

$$A_\nu = \sum_{j=1}^{\nu} \sigma_j \mathbf{u}_j \mathbf{v}_j^*. \tag{11.8}$$

then

$$\|A - A_\nu\|_2 = \inf_{\substack{B \in R^{m\times n} \\ rank(B) \le \nu}} \|A - B\|_2 = \sigma_{\nu+1} \tag{11.9}$$

where we define $\sigma_{\nu+1} = 0$ if $\nu = s$.

Proof: (Following [**?**]). Suppose there is a B with $\text{rank}(B) \le \nu$ such that $\|A - B\|_2 < \sigma_{\nu+1}$. The rank condition implies that the null space, $W \subseteq \mathrm{C}^n$ is of dimension $n - \nu$. Thus for any $\mathbf{w} \in W$ we have $A\mathbf{w} = (A - B)\mathbf{w}$ with

$$\|A\mathbf{w}\|_2 = \|(A - B)\mathbf{w}\|_2 \le \|A - B\|_2 \|\mathbf{w}\|_2 < \sigma_{\nu+1}\|\mathbf{w}\|_2.$$

This implies that $\|A\mathbf{w}\|_2 < \sigma_{\nu+1}\|\mathbf{w}\|_2$ on the $n - \nu$ dimensional subspace W. On the other hand the space spanned by the first $\nu + 1$ right singular vectors of A is an $(\nu + 1)$-dimensional subspace where $\|A\mathbf{w}\|_2 \ge \sigma_{\nu+1}\|\mathbf{w}\|_2$. Since the sum of the

dimensions of these spaces exceeds n, there must be a nonzero vector for which both $\|A\mathbf{w}\|_2 < \sigma_{\nu+1}\|\mathbf{w}\|_2$ and $\|A\mathbf{w}\|_2 \geq \sigma_{\nu+1}\|\mathbf{w}\|_2$ is satisfied, which is clearly a contradiction. $\square$

Thus if $\sigma_{\nu+1}$ is sufficiently small it is safe to keep only ν singular values. The idea is illustrates in Figure 11.5.5. The upper left hand image is the original 256×256 image and the other three images show the reconstruction using 20, 50 and 75 singular values.

It is also interesting to look a the singular values themselves. Figure 11.5.6 shows that they drop off rather rapidly, explaining why a reasonable reconstruction is obtained using a relatively small number of terms in the approximation, (11.9). However, we should point out that singular value compression is not by any means the best way of compressing general images. The JPEG algorithm and especially the more recent wavelet-based algorithms are much better.

Example:: Let

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0.01 & 1 & 1 \end{bmatrix}.$$

Calculating the SVD of A we find that

$$\Sigma = \begin{bmatrix} 2.83020006 & & & \\ & 1.41494759 & & \\ & & 1.409958 & \\ & & & 0.00306759 \end{bmatrix},$$

with U and V not written down (only showing the nonzero entries of Σ). From Σ it is clear that A is of rank 4. However, the fourth singular value is small in comparison with the other three and it is therefore possible that it is the result of some error that crept into the coefficients of A. Let us therefore find the best rank 3 approximation of A. For that purpose we drop

(a) Original (b) 20 singular values

(c) 50 singular values (d) 75 singular values

FIGURE 11.5.5. Original image and reconstructions keeping different numbers of singular values.

the smallest singular value from Σ to form

$$\widehat{\Sigma} = \begin{bmatrix} 2.83020006 & & & \\ & 1.41494759 & & \\ & & 1.409958 & \\ & & & 0 \end{bmatrix}.$$

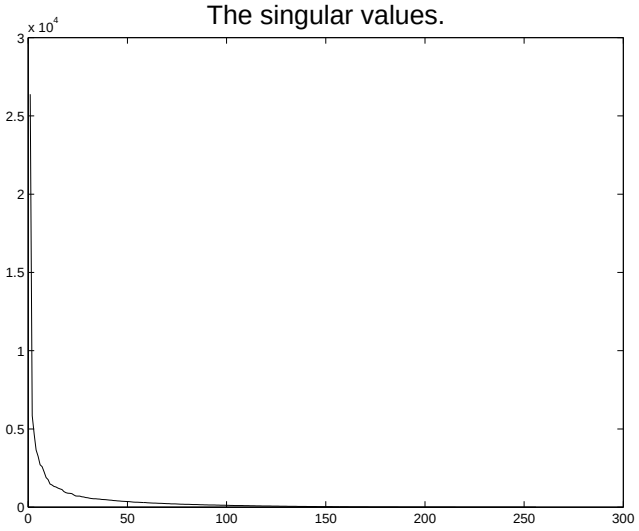


FIGURE 11.5.6. The singular values in non-increasing order.

The best rank 3 approximation of A is then given by $\widehat{A} = U\widehat{\Sigma}V^T$ with U and V the left and right singular vector matrices of A respectively. Using Python or Matlab to help with the calculations we find that

$$\widehat{A} = \begin{bmatrix} 0.999 & 1.001 & 1.001 & 0.999 \\ 1.000 & 0.000 & 1.000 & 0.000 \\ 1.001 & 0.999 & -0.001 & 0.001 \\ 0.000 & 1.000 & 0.000 & 1.000 \\ 0.001 & 0.009 & 0.999 & 1.001 \end{bmatrix},$$

written to three decimal places.□

Example:: In the previous example errors destroyed the null space of A, and we noticed that it has rank 4. If we need an approximation of the null space of A as it was prior to destruction, the best one can do is to calculate the null space of the best rank-3 approximation of A, i.e. we calculate the null space of $\widehat{A}$. According to Theorem 26 it is one-dimensional and a basis is

given by the last column of V on the SVD of A, i.e.

$$\widehat{\mathbf{n}} = \begin{bmatrix} -0.499 \\ 0.501 \\ 0.498 \\ -0.502 \end{bmatrix}.$$

Assuming that the original matrix, before contamination, is given by

$$A = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix},$$

we calculate its null space and find that it is given by

$$\mathbf{n} = \begin{bmatrix} -0.5 \\ 0.5 \\ 0.5 \\ -0.5 \end{bmatrix}.$$

Since both $\widehat{\mathbf{n}}$ and $\mathbf{n}$ are normalized, a useful comparison is to calculate the angle between the two, $\cos\theta = 0.999997$, accurate to the number of digits given. This means that the two vectors very nearly point in the same direction—a very good approximation of the null space of the original, uncontaminated, matrix indeed.

□

11.5.6. The SVD and covariances. In this section we approach the SVD from a slightly different point of view. Imagine that we are given m n-dimensional vectors, $\mathbf{X}_j, \quad j = 1, \ldots, m$ with a zero average, i.e. $\mathbf{A} = \frac{1}{m}\sum_{j=1}^{m} \mathbf{X}_j = \mathbf{0}$. If the average is not zero, we can always form a new system with zero average by subtracting the average from the original system.

Previously we asked questions such as to find an orthonormal basis for the subspace spanned by these vectors. Instead, let us now ask for the average direction in which the vectors are aligned. One can also think of it as the direction of maximum

deviation from the origin. In this sense we are looking for a correlation between the vectors. Mathematically, we are looking for the direction $\mathbf{u}$ that maximizes μ where

$$\mu = \max_{\|\mathbf{u}\|_2=1} \frac{1}{m} \sum_{j=1}^{m} (\mathbf{u}^* \mathbf{X}_j)^2 \tag{11.10}$$

Introducing the covariance matrix

$$C = \frac{1}{m} \sum_{j=1}^{m} \mathbf{X}_j \mathbf{X}_j^*$$

it is possible to rewrite (11.10) as

$$\mu = \max_{\|\mathbf{u}\|_2=1} (\mathbf{u}^* C \mathbf{u}) . \tag{11.11}$$

Since C is symmetric, it can be diagonalized with a unitary matrix, U, i.e.

$$C = U \Lambda U^*.$$

where $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_n)$ and we again order the eigenvalues λ_j in decreasing order of magnitude.

Exercise: Show that the eigenvalues of C are always non-negative. □

With the introduction of the covariance matrix C, (11.11) is rewritten as

$$\mu = \max_{\|\mathbf{u}\|_2=1} (\mathbf{u}^* U \Lambda U^* \mathbf{u}) \tag{11.12}$$

$$= \max_{\|\mathbf{y}\|_2=1} (\mathbf{y}^* \Lambda \mathbf{y}) . \tag{11.13}$$

where

$$\mathbf{y} = U^* \mathbf{u}.$$

Also note that the fact that U is unitary ensures that $\|\mathbf{y}\|_2 = \|\mathbf{u}\|_2 = 1$. Thus we are required to calculate

$$\mu = \max_{\|\mathbf{y}\|_2=1} \sum_{j=1}^{n} \lambda_j |y_j|^2 \tag{11.14}$$

subject to $\sum_{j=1}^{n} |y_j|^2 = 1$. Hence it follows that $\mu = \lambda_1$ and $\mathbf{y} = \mathbf{e}_1$ where $\mathbf{e}_j$ has a one in its j-th entry, the rest are all zeros. Since $\mathbf{u} = U\mathbf{y} = U\mathbf{e}_1$, it follows that the direction of maximum deviation, $\mathbf{u}$, we were looking for, is exactly the eigenvector, $\mathbf{u}_1$, of the covariance matrix C, belonging to the largest eigenvalue λ_1. Moreover, λ_1 provides a measure for the deviation in the direction of $\mathbf{u}_1$.

Let us now ask for the direction of maximum deviation orthogonal to $\mathbf{u}_1$. Thus we are again required to calculate (11.14) but this time subject to $\sum_{j=1}^{n} |y_j|^2 = 1$ *and* $\mathbf{e}_1^* \mathbf{y} = 0$. Since this implies that the first component of $\mathbf{y}$ equals zero, i.e. $y_1 = 0$, it follows that (11.14) is calculated subject to $\sum_{j=2}^{n} |y_j|^2 = 1$. This is achieved by $\mathbf{y} = \mathbf{e}_2$, or $\mathbf{u} = U\mathbf{e}_2 = \mathbf{u}_2$, the eigenvector of C belonging to the second largest eigenvalue.

Continuing in this fashion we arrive at the following result: The first eigenvector of the covariance matrix C points in the direction of maximum variation and the corresponding eigenvalue measures the variation in this direction, i.e. it is the *variance* in this direction. The subsequent eigenvectors point in the directions of maximum variation orthogonal to the previous directions with their associated eigenvalues again measuring the variations in these directions.

Finally notice that the same results are obtained by calculating the SVD of the matrix

$$X = \frac{1}{\sqrt{m}} [\mathbf{X}_1 \dots \mathbf{X}_m].$$

The directions of maximum variation are then given by the columns of U and the variances by the squares of the singular values.

Let us now illustrate these ideas two examples.

Example: Let us consider the problem of finding a face in an image. One possible (rather naive) approach would be to identify a number of objects that might qualify and measure their width and height. It is easy to imagine that the measurements

of any face should fall within a certain range. Anything outside the 'typical' range can be discarded. Anything inside the range can then be investigated in more detail for further face-like characteristics. The problem is to find the 'typical' range of measurements for faces.

Although this example is contrived—the number of features we measure is too low—it is not entirely unrealistic. In fact, one of the earliest identification systems, and at the time a serious rival for fingerprints, was based on a comprehensive measurements of individuals [?]. The system developed by Alphonse Bertillion in France during the 1870's employed eleven separate measurements of an individual: height, length and breadth of head, length and breadth of ear, length form elbow to end of middle finger, lengths of middle and ring fingers, length of left foot, length of the trunk, and length of outstretched arms from middle fingertip to middle fingertip. Apart from being to able distinguish between different individuals, it also allowed a classification system that enabled Bertillion to quickly locate the file of a criminal, given just the measurements. The system was so successful that France was one of the last countries to adopt fingerprints for personal identification, see [?].

Figure 11.5.7 shows the actual measurements of a number of faces from a student population. The width is measured from ear-to-ear and the height from the chin to between the eyes. Note the natural distribution of the sizes of the faces around a certain mean value. Given a particular measurement, the question is whether it belongs to this distribution. It should be clear that one cannot simply use the distance from the mean—the ellipse clearly gives a better indication of the *nature* of the distribution. It makes therefore more sense to classify measurements in the vicinity of the ellipse as belonging to the distribution. Of course this does not mean that it is actually a face, only that it is possibly a face and that a more detailed investigation is in order.

In this case the mean of the 48 facial measurements $\mathbf{x}_j = [x_j, y_j]^T$ is calculated, $\mathbf{a} = \sum_{j=1}^{48} \mathbf{x}_j$ and the matrix X is defined by

$$X = \frac{1}{\sqrt{48}} \left[\mathbf{x}_1 \cdots \mathbf{x}_{48} \right].$$

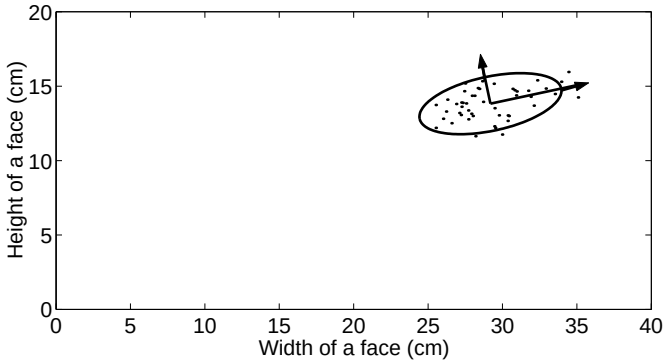


FIGURE 11.5.7. The distribution of the width and height measurements of a sample of faces.

The SVD of X now gives the 2×2 matrices U and Σ. The columns of U give the principle axes of the ellipse and in this case the size of the ellipse is two times the standard deviation.

For this example we find that

$$U = \begin{bmatrix} -0.9778 & -0.2095 \\ -0.2095 & 0.9778 \end{bmatrix} \text{ and } \Sigma = \begin{bmatrix} 2.43 & 0 \\ 0 & 0.91 \end{bmatrix}. \tag{11.15}$$

Incidentally, what happens if the mean is not removed from the measurements?

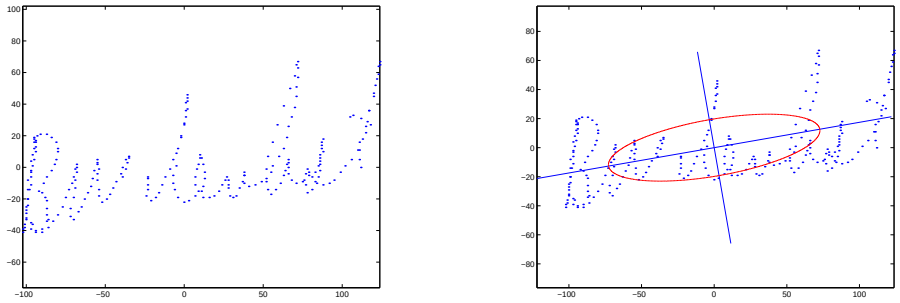
□

Example: Figure 11.5.8(a) shows 262 different points in (x,y)-space representing a discretized a handwritten signature. Note that the mean of these points is at the origin. Let X be the 2×262 matrix obtained when placing the (x,y)-values in successive columns, scaled by the factor $\frac{1}{\sqrt{262}}$ (scaling Σ with the number of point samples). When calculating the SVD of X, the 2×2 $U$and Σ-matrices are obtained from

$$XX^*U = U\Sigma^2.$$

For this example we find that

$$U = \begin{bmatrix} -0.9851 & 0.1718 \\ -0.1718 & -0.9851 \end{bmatrix} \text{ and } \Sigma = \begin{bmatrix} 73.79 & 0 \\ 0 & 19.52 \end{bmatrix}$$



(a) (b)

FIGURE 11.5.8. (a) Data set in the form of a signature. (b) The principle axes as well as the hyper-ellipse associated with the data set.

In Figure 11.5.8(b) we draw the columns of U together with the signature. The eigenvector belonging to the larger singular value, 73.79, indicates the direction of maximum variation between the vectors constituting the signature. The ellipse shown in the figure has principle semi-axes of lengths 73.79 and 19.52. Any automated signature verification system should be invariant with respect to the size and rotation of the signature. This example suggests a way (by no means the only possibility) of normalizing the orientation of the signatures—use the direction of maximum variation as the horizontal axis.

□

Let us now briefly consider what happens if the mean of the points is not at the origin. In fact let us assume that the mean is very far from the origin. If we now proceed as above by forming X from the different vectors constituting the signature, without removing the mean, and calculating $C = XX^*$, the first eigenvector still points in the direction of the maximum variation. Since the origin is now very far from the mean, it simply points in the direction of the mean and the second eigenvector is constrained to be orthogonal to the first one. Thus in this case the two eigenvectors fail to describe the variation between the data points themselves and become a function of the choice of the origin. For this reason it is often appropriate to subtract the mean from a data set.

11.6. Overdetermined linear system and the generalized inverse.

11.6.1. Solution by Normal Equations. An over-determined linear system can graphically be written

$$\begin{bmatrix} & \\ & A \\ & \end{bmatrix}_{m\times n} \begin{bmatrix} \\ x \\ \end{bmatrix}_n = \begin{bmatrix} \\ b \\ \\ \end{bmatrix}_m \tag{11.1}$$

where $m > n$. Typically, these systems lack a solution—there are too many constraints imposed on the variables $x_1, x_2, ..., x_n$. The question to ask is not what vector x makes $Ax - b = 0$, but what vector x makes $\|Ax - b\|$ as small as possible (unless otherwise stated $\|\cdot\|$ refers to the L^2-norm defined in the previous section; in the next section more general norms will be discussed). If we multiply (11.1) from the left with A^T, we obtain a square linear system known as the *normal equations*.

THEOREM 28. *If A is of full rank, the solution to the normal equations*

$$A^T A\mathbf{x} = A^T\mathbf{b} \tag{11.2}$$

is unique and provides the least squares solution to $Ax = b$.

Proof: We prove this theorem in two different ways; the first is a geometric proof and the second is purely algebraic.

Geometric proof:

The situation we are looking at is illustrated in Figure 11.6.1: we need to solve the system $A\mathbf{x} = \mathbf{b}$ where $\mathbf{b}$ is outside the column space of A. Therefore an exact solution does not exist. The idea is to find the solution of an alternative system $A\mathbf{x} = \hat{\mathbf{b}}$ where $\hat{\mathbf{b}}$ is the vector in the column space of A as close as possible to $\mathbf{b}$. This will of course be the case if $\mathbf{e} = \mathbf{b} - A\mathbf{x}$ is orthogonal to the columns space of A. From Figure 11.5.4, this means that $\mathbf{e}$ is in the left nullspace of A, or $A^T\mathbf{e} = \mathbf{0}$, which is exactly the normal equations (11.2).In order to show that $A^T A$ is non-singular, consider $A^T A\mathbf{x} = \mathbf{0}$. Therefore $A\mathbf{x}$ lies in both the column– and left null-space of A. Since they are orthogonal it follows that $A\mathbf{x} = \mathbf{0}$. On the other hand, the fact that

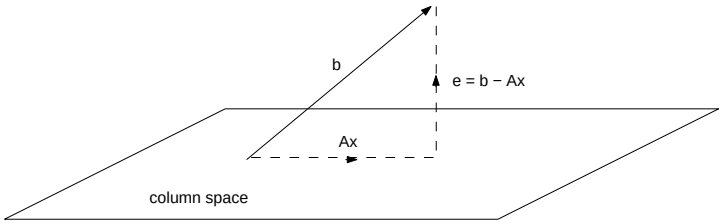


FIGURE 11.6.1. Orthogonal projection onto the column space.

A is of full rank implies that its null-space consists of only the zero element. Thus from $A^TA\mathbf{x} = \mathbf{0}$ implies that $\mathbf{x} = \mathbf{0}$, showing that A^TA is non-singular.

Algebraic proof:
Now the the idea is to demonstrate that if x satisfies $A^TAx = A^Tb$, then $\|Ax - b\| \leq \|Ay - b\|$ for any vector y. We denote these two residuals by $r_x = Ax - b$ and $r_y = Ay - b$ respectively. From the normal equations, we know that

$$A^Tr_x = 0, \text{ implying } r_x^TA = 0 \tag{11.3}$$

Then

$$r_y = Ay - b = (Ax - b) - (Ay - Ax) = r_x + A(y - x)$$

and

$$\|r_y\|^2 = (r_x + A(y - x))^T(r_x + A(y - x)) =$$

$$= \|r_x\|^2 + r_x^TA(y - x) + [A(y - x)]^Tr_x + \|A(y - x)\|^2 .$$

Both the two middle terms in the line above vanish because of (11.3). The last term is non-negative. Hence $\|r_y\|^2 \geq \|r_x\|^2$.□

The coefficient matrix in the normal equations is of the form A^TA. This is always a symmetric matrix. Assuming A has rank n (we discuss the general case in the next section), it is furthermore positive definite since $y^T(A^TA)y = \|Ay\|^2 > 0$ whenever

$y \neq 0$. This makes a couple of very effective numerical methods available for solving the normal equation system:

- *Cholesky's method* is conceptually similar to Gaussian elimination, but requires no pivoting and has about half the operation count.
- The *conjugate gradient method* is iterative, but often converges very fast. In this algorithm, one needs to repeatedly multiply the coefficient matrix with different vectors—no other operations are performed with the matrix entries. This makes this method particularly attractive if the matrix is sparse—zero entries can be fully utilized to save on arithmetic operations.

The big disadvantage with the normal equation approach is that, in some cases, the resulting system can be very sensitive to small errors in A and b—the process of forming A^TA can lead to an ill-conditioned linear system.

11.6.2. Solution by QR factorization. With A split as $A = QR$, we get $Ax - b = QRx - b$ and (since multiplying any vector by a unitary matrix leaves the 2-norm unchanged)

$$\|Ax - b\| = \|Rx - Q^*b\|$$

At this point, a picture illustrating the 'shape' of $Rx - Q^*b$ helps:

Regarding the components in the vector $Rx - Q^*b$: choosing different x does not change the last $m - n$ rows and we can make the first n rows all zero by solving $R\mathbf{x} = \mathbf{b}_1$ where $\mathbf{b}_1$ is the first n rows of $Q^*\mathbf{b}$. It should be clear that this minimizes the norm of the residual.

We can formulate this observation in a convenient algorithm as follows:

The cost in terms of arithmetic operations is somewhat larger than in the normal equations approach, but the algorithm can be shown to be entirely numerically stable.

11.6.3. Solution by SVD factorization and the generalized inverse. The procedure is almost identical to the one based on the QR factorization. After noting that

$$\|Ax - b\| = \|\Sigma V^*x - U^*b\|$$

we ignore, as before, the last $m-n$ rows and solve the resulting square $n \times n$ system. However, we can do even better. Let us now consider the completely general situation

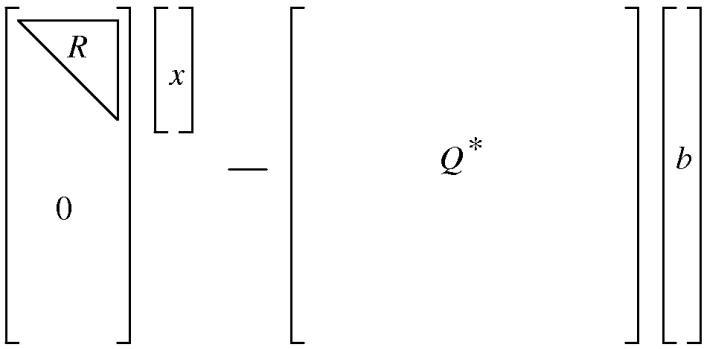


FIGURE 11.6.2. Shape of the QR factors.

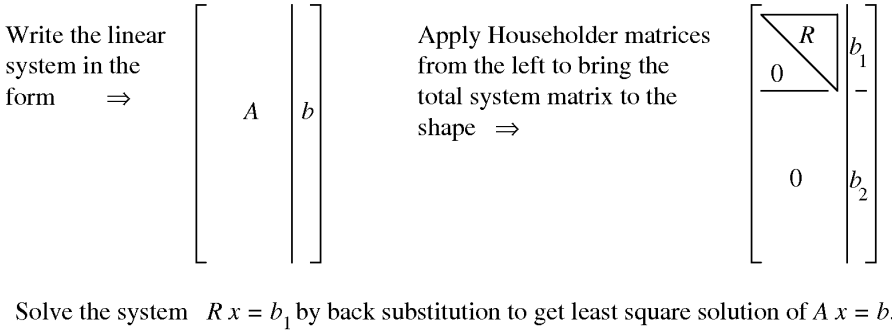


FIGURE 11.6.3. Solution by QR factorization.

shown in Figure 11.6.4. In this case we want to 'solve' the system $A\mathbf{x} = \mathbf{b}$ where $\mathbf{b}$ does not necessarily lie the column space of A and/or the null-space is empty, i.e. A may also be rank deficient.

We already know the answer to the problem of $\mathbf{b}$ not being in the column space of A: we project $\mathbf{b}$ orthogonally onto the column space, i.e. we write

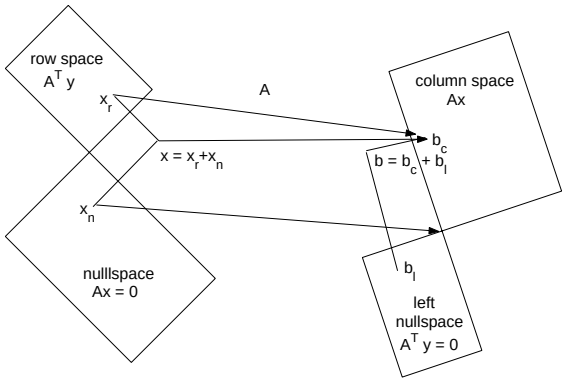


FIGURE 11.6.4. The generalized inverse.

$$\mathbf{b} = \mathbf{b}_c + \mathbf{b}_l,$$

with $\mathbf{b}_c$ and $\mathbf{b}_l$ in the column– and left null-spaces of A, respectively. Solving $A\mathbf{x} = \mathbf{b}_c$ amounts to solving the normal equations (11.2). However, since A is rank deficient (its null-space is not the trivial one), there is not a unique solution for this system—adding any element of the null-space to a solution gives another solution. For example, let $\mathbf{x}_p$ be any solution of $A\mathbf{x}_p = \mathbf{b}_c$, clearly $\mathbf{x} = \mathbf{x}_p + \mathbf{x}_n$ for any $\mathbf{x}_n \in \text{null}(A)$, is also a solution. However, it is possible to identify among all these solutions a unique one with special properties. suppose we decompose $\mathbf{x}_p$ and write it (uniquely) as the sum of its orthogonal parts in the row– and null-spaces of A,

$$\mathbf{x}_p = \mathbf{x}_r + \mathbf{x}_n.$$

Then $A\mathbf{x}_r = \mathbf{b}_c$, i.e. $\mathbf{x}_r$ is also a solution. The point is, $\mathbf{b}_r$ is the *unique* solution in the row space of A. Suppose there is another solution in row(A), $A\mathbf{y}_r = \mathbf{b}_c$ we find that $A(\mathbf{x}_r - \mathbf{y}_r) = \mathbf{0}$, showing that $\mathbf{x}_r - \mathbf{y}_r$ lies in both the row– and the null-space of A. Since these spaces are orthogonal, $\mathbf{x}_r - \mathbf{y}_r$ is orthogonal to itself—it has to be zero.

We have now identified a candidate for a unique solution of the general system $A\mathbf{x} = \mathbf{b}$, namely the unique vector in the row space of A that solves the system

obtained from projecting $\mathbf{b}$ orthogonally onto the column space of A. Any other solution of this system adds an orthogonal component from the null-space of A to $\mathbf{x}_r$, i.e. any other solution has a larger L^2 norm. Thus the unique element in row(A) is also the solution with the smallest norm. It only remains to find a formula for $\mathbf{x}_r$; for this the SVD is particularly convenient.

Using the reduced form of the SVD, we substitute

$$A = U_r \Sigma_r V_r^T$$

in the normal equations to obtain

$$V_r^T \mathbf{x} = \Sigma_r^{-1} U_r^T \mathbf{b}.$$

Since A is not necessarily of full rank this may be an under-determined system with possibly an infinite number of solutions. As explained above, a unique solution exists if we restrict ourselves to the row space of A, i.e. if we let $\mathbf{x}_r = A^T \mathbf{y} = V^r \Sigma_r U_r^T \mathbf{y}$ for some $\mathbf{y}$. Substitution into the expression above yields

$$\Sigma_r U_r^T \mathbf{y} = \Sigma_r^{-1} U_r^T \mathbf{b}.$$

Pre-multiplying with V_r yields the generalized inverse

$$\mathbf{x}_r = V_r \Sigma_r^{-1} U_r^T \mathbf{b}. \tag{11.4}$$

Note that in case A is (square and) non-singular, this solution simply becomes $\mathbf{x} = A^{-1}\mathbf{b}$. In case it is rectangular ($m < n$) and of full rank, it reduces to the standard linear least squares solutions. And it also provides a unique solution in the general case when A is any $m \times n$ matrix.

Although the SVD provides the most general answer in terms of the generalized inverse, it should be kept in mind that is is relatively expensive to compute. For example, the operation count in computing the QR factorization ($\frac{4}{3}n^3$ for a full $n \times n$ matrix A) is significantly lower than for the SVD ($4n^3$ for Σ only, $26n^3$ for U, Σ, V; these numbers are approximate since SVD procedures are iterative). For this reason,

QR factorization is usually the preferred approach for least squares problems when A is of full rank.

Let us now return to the examples of the Introduction, Section 11.1.

EXAMPLE 29. Returning to the example at the beginning of the section we find the generalized solution of

$$\begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = 1.$$

The null space is given by $x + y = 0$, and $\begin{bmatrix} 1 \\ -1 \end{bmatrix}$ is a basis of the null space. We are looking for a solution orthogonal to the null space or, equivalently, a solution in the row space of A. A basis for the row space is $\begin{bmatrix} 1 & 1 \end{bmatrix}$ This implies that the solution we are looking for should satisfy $x = y$, i.e. the generalized solution is $\begin{bmatrix} x \\ y \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$, see Figure 11.6.5.

□

EXERCISE 30. Consider the system

$$\begin{aligned} x + y &= 1 \\ 2x + 2y &= 1. \end{aligned}$$

(1) Write the system in the form $A\mathbf{x} = \mathbf{b}$.
(2) Calculate the four fundamental spaces of A, and draw them in a figure.
(3) Can you calculate a least squares solution of the system? Why not?
(4) Plot $\mathbf{b}$ in your figure, and note that it does not lie in the column space of A. Find the orthogonal projection of $\mathbf{b}$ on the column space of A. Call it $\widehat{\mathbf{b}}$.
(5) Find all the solutions of $A\mathbf{x} = \widehat{\mathbf{b}}$, and draw them in your figure. Find the generalized solution and plot it in the figure.
(6) Write down the SVD of A without calculating any eigenvalues and eigenvectors. Explain how you figure out the singular values of A.

□

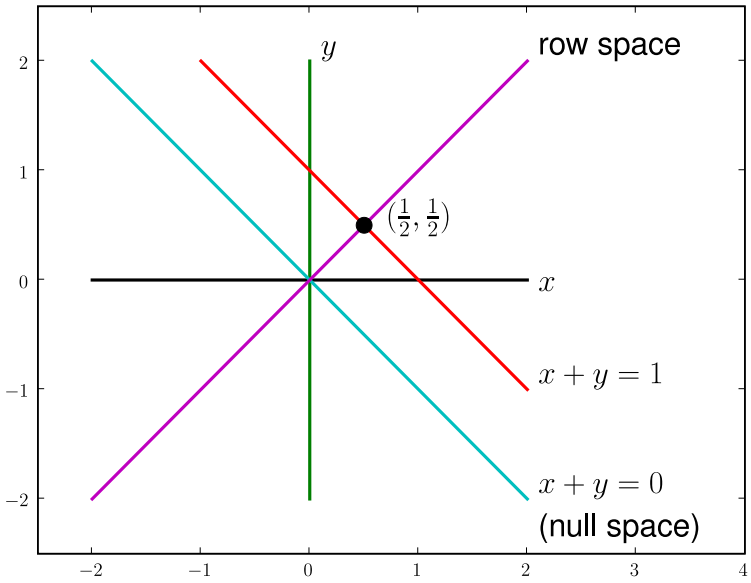


FIGURE 11.6.5. The generalized solution.

11.7. Vector and matrix norms.

This section summarizes a few facts about *norms* —scalar quantities which in some sense measure the 'size' of a vector or a matrix. Norms are very widely used in linear algebra to estimate things like errors and rates of convergence.

DEFINITION 31. A *vector norm* associates with a vector x a scalar number that we denote $\|x\|$. To be called a *norm*, we furthermore require

$\|x\| \geq 0$, with equality if and only if $x = 0$.

$\|cx\| = |c| \ \|x\|$ for any complex number c.

$\|x + y\| \leq \|x\| + \|y\|$ (triangle inequality). □

These three requirements may appear somewhat arbitrary, but it turns out that requiring just them and nothing more makes norms a really useful concept. Many different vector norms are possible, and it depends on the context which one is most

suitable. The following are examples of vector norms (x_i denotes the components of x):

Examples of vector norms:

$\|x\|_{\max}$	$= \max_i \|x_i\|$	max-norm; also called 'infinity norm' and often denoted $\|x\|_\infty$
$\|x\|_1$	$= \sum \|x_i\|$	'one'-norm
$\|x\|_2$	$= \sqrt{\sum \|x_i\|^2}$	Eucledian norm or 'two'-norm

Most often, in this book, we assume the $\|\cdot\|_2$ norm if nothing else is specified.

DEFINITION 32. A *matrix norm* $\|A\|$associates with a square matrix A a scalar number. To be called a matrix norm, we require

$\|A\| \geq 0$, with equality if and only if $A = 0$.

$\|cA\| = |c| \ \|A\|$ for any complex number c.

$\|A + B\| \leq \|A\| + \|B\|$ (triangle inequality).

$\|AB\| \leq \|A\| \, \|B\|$.□

The first three requirements correspond precisely to the three for vector norms - the fourth is additional.

THEOREM 33. *Given any vector norm, we can define an associated matrix norm by means of*

$$\|A\| = \max_{\|x\|=1} \|Ax\| \tag{11.1}$$

(note that both norms in the RHS are vector norms).

Proof: To prove this theorem, one needs to show that the quantity $\|A\|$ introduced in this way actually satisfies the four requirements. A very useful consequence turns out to be

$$\|Ax\| \leq \|A\| \, \|x\|$$

We must here use the same type of norm throughout the expression. □

THEOREM 34. *The matrix norms associated with the different vector norms above are:*

$$\begin{array}{lcll} \|A\|_{\max} & = & \max_i \sum_{j=1}^n |a_{ij}| & \textit{max row sum (taking absolute values)} \\ \|A\|_1 & = & \max_j \sum_{i=1}^n |a_{ij}| & \textit{max column sum (taking absolute values)} \\ \|A\|_2 & = & \rho(A^*A) & \textit{max eigenvalue to the matrix } A^*A \end{array}$$

□

Although (11.1) shows how every vector norm leads to a matrix norm, not all matrix norms can be generated in this way. A notable example is the Fröbenius norm $\|A\|_E = \sqrt{\sum_{i,\,j} |a_{ij}|^2}$. This can be shown to satisfy all matrix norm requirements. However, the fact that $\|I\|_E = \sqrt{n}$ rules out that it could have arisen from a vector norm through (11.1).

Some useful results, holding for all matrix norms, are

THEOREM 35. $\rho(A) \leq \|A\|$ *(with* $\rho(A)$ *denoting the* spectral radius *of* A*—the largest eigenvalue in magnitude).* □

THEOREM 36. $\|A\| < 1$ *implies that* $A^k \to 0$ *for* $k \to \infty$.□

11.8. Conditioning.

Section to be written.

CHAPTER 12

POLYNOMIAL INTERPOLATION

12.1. Introduction.

Figure 12.1.1 shows the coordinates of a signature recorded by a digitizing tablet. This representation is not quite convenient; it is more natural to connect the points and Figure 12.1.2 shows the same signature but this time connected with straight lines. This is of course, an example of a piecewise linear interpolation—the points are connected by pieces of linear polynomials.

Even this signature may not be quite what we want. Since the points are connected by straight lines the signature has a distinct jagged appearance. Although it is continuous, it does not have a continuous first derivative, and it shows in Figure **??**. A much smoother curve should be obtained if we connect the points with pieces with matching higher derivatives, typically first and second derivatives. Another, perhaps more subtle problem is that the original points are noisy to begin

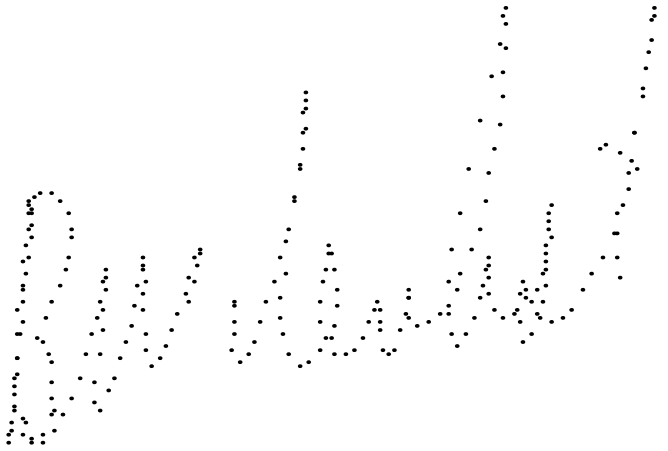


FIGURE 12.1.1. Signature recorded by a digitizing tablet.

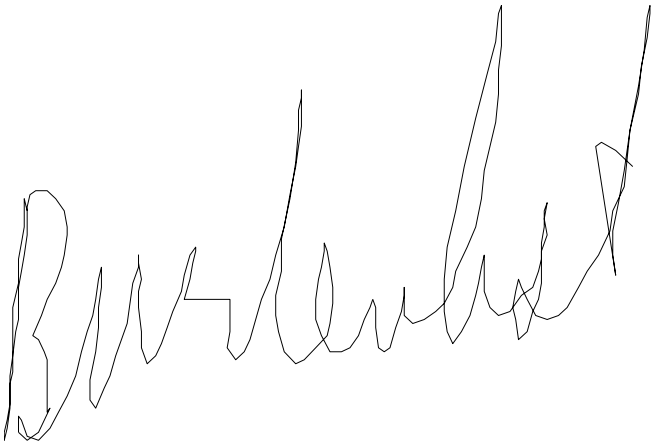


FIGURE 12.1.2. The same signature with the dots connected with straight lines.

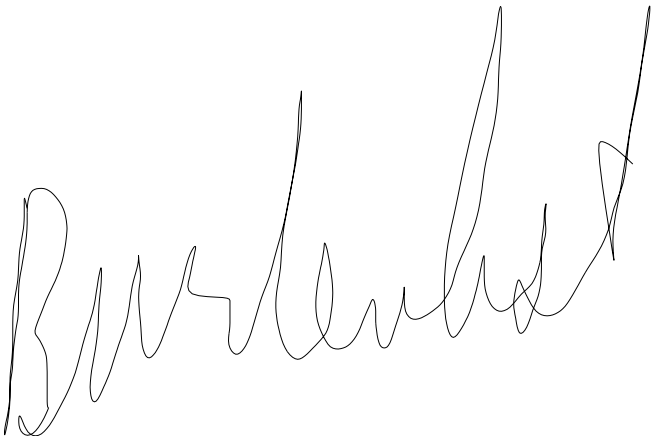


FIGURE 12.1.3. The same signature after smoothing.

with—because of the construction of the digitizing tablet, they are not in their correct positions. In many applications it is necessary to filter out the noise, i.e. to smooth the original points in some way. The result of one such filtering process is shown in Figure 12.1.3. In this Figure the approximating polynomial is piecewise cubic and also has continuous second derivatives.

In general, if we are given a set of points $(x_j, f(x_j)), \quad j = 0, \ldots, N$, the interpolation polynomial $p_N(x)$ of degree at most N, satisfies $p_N(x_j) = f(x_j), \quad j = 0, \ldots, N$. It is important to note that there is just one interpolation polynomial of degree at

most N satisfying the interpolation conditions at the $N+1$ interpolation points. For suppose there is a second one, $q_N(x)$, interpolating $f(x)$ at the same $N+1$ points. The difference $p_N(x) - q_N(x)$ is again a polynomial of degree at most N, but it has $N+1$ zeros, at the $N+1$ interpolation points. Since a polynomial of degree N has at most N real zeros, the difference has to be identically zero. This means that although one can write the interpolation polynomial in different ways, it remains exactly the same polynomial.

In this chapter we address the problem of constructing the interpolation polynomial, we investigate its accuracy and we'll discuss the problem of efficient implementation as well as the removal of noise in the data points. The latter two questions will lead us to subdivision schemes for curves.

12.2. The Lagrange interpolation polynomial.

Given the $N+1$ interpolation points

$$(x_j, f(x_j)), \quad j = 0, \ldots, N,$$

we form the $N+1$ polynomials of degree N, uniquely defined by,

$$L_k(x_j) = \delta_{jk}, \quad j, k = 0, \ldots, N. \tag{12.1}$$

The Lagrange form of the interpolation polynomial is then given by

$$p_N(x) = \sum_{j=0}^{N} f(x_j) L_j(x). \tag{12.2}$$

Note that the condition (12.1) ensures that $p_N(x_j) = f(x_j)$. An explicit expression for the $L_j(x)$'s is given by

$$L_j(x) = \frac{\prod_{\substack{k=0 \\ k\neq j}}^{N}(x - x_k)}{\prod_{\substack{k=0 \\ k\neq j}}^{N}(x_j - x_k)}. \tag{12.3}$$

This formula is very useful for theoretical purposes; as it stands however, it is computationally not very effective. A single evaluation of the Lagrange form of the interpolation polynomial takes $O(N^2)$ operations. Although this count can be reduced to $O(N)$ operations using the so-called barycentric formula, we'll reduce the count by introducing Newton's form of the interpolation polynomial.

12.3. Newton's form of the interpolation polynomial.

Newton's form of the interpolation polynomial is given by,

$$p_N(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \cdots + a_N(x - x_0)\cdots(x - x_{N-1}) \tag{12.1}$$

where the a_j's are constants to be determined such that $p_N(x)$ satisfies the interpolation condition $p_N(x_j) = f(x_j), \quad j = 0, \ldots, N$. One of the advantages of Newton's form is that it is built from lower order interpolation polynomials. Suppose that we have somehow obtained the a_j's such that $p_N(x)$ satisfies the interpolation condition. Using just the first k of these coefficients, let

$$q_k(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)(x - x_1) + \cdots + a_k(x - x_0)\cdots(x - x_{k-1}).$$

It follows that

$$p_N(x) = q_k(x) + (x - x_0)\cdots(x - x_k)r(x)$$

where $r(x)$ is a polynomial of degree at most $n-k-1$. The first important observation is that

$$f(x_j) = p_N(x_j) = q_k(x_j), \quad j = 1, \ldots, k,$$

implying that $q_k(x)$ is the interpolation polynomial interpolating at the first $k+1$ interpolation points. The second important observation is that a_k is the leading coefficient of the interpolation polynomial, interpolating $f(x)$ at $x_0, \ldots, x_k$. We want to emphasize this fact in our notation and therefore rewrite

$$a_k = f[x_0, \ldots, x_k],$$

so that Newton's form of the interpolation polynomial can be written as

$$\begin{aligned} p_N(x) &= f[x_0] + f[x_0, x_1](x - x_0) + f[x_0, x_1, x_2](x - x_0)(x - x_1) + \cdots \quad (12.2) \\ &\quad + f[x_0, \ldots, x_N](x - x_0) \cdots (x - x_{N-1}) \quad (12.3) \end{aligned}$$

Now we are ready to derive an expression for the coefficients. Consider the polynomial,

$$q(x) = r(x)\frac{x - x_0}{x_k - x_0} + s(x)\frac{x - x_k}{x_0 - x_k} \quad (12.4)$$

where $r(x)$ and $s(x)$ are both interpolation polynomials of degree at most $k-1$ with $r(x_j) = f(x_j), \quad j = 1, \ldots, k$ and $s(x_j) = f(x_j), \quad j = 0, \ldots, k-1$. Their leading coefficients are therefore given by $f[x_1, \ldots, x_k]$ and $f[x_0, \ldots, x_{N-1}]$, respectively. It should be easy to verify that $q(x)$ is therefore the interpolation polynomial satisfying, $q(x_j) = f(x_j), \quad j = 0, \ldots, k$. Its leading coefficient is therefore given by $f[x_0, \ldots, x_k]$ and comparing it with the leading coefficient of the right hand side of (12.4) gives,

$$f[x_0, \ldots, x_k] = \frac{f[x_1, \ldots, x_k] - f[x_0, \ldots, x_{k-1}]}{x_k - x_0} \quad (12.5)$$

With $f[x_j] = f(x_j), \quad j = 0, \ldots, N$ this is a recursive formula for the calculation of the coefficients of Newton's form of the interpolation polynomial. The recursion is displayed in the following table,

$$
\begin{array}{llll}
x_0 & f[x_0] & & \\
 & & f[x_0,x_1] & \\
x_1 & f[x_1] & & f[x_0,x_1,x_2] \\
 & & f[x_1,x_2] & & f[x_0,x_1,x_2,x_3] \\
x_2 & f[x_1] & & f[x_1,x_2,x_3] \\
 & & f[x_2,x_3] & \\
x_3 & f[x_3] & &
\end{array}
$$

where

$$f[x_2,x_3] = \frac{f[x_3]-f[x_1]}{x_3-x_2}$$

and

$$f[x_0,x_1,x_2,x_3] = \frac{f[x_1,x_2,x_3]-f[x_0,x_1,x_2]}{x_3-x_0},$$

for example.

The calculation of the differences (12.5) is the most expensive part of the algorithm, requiring $\frac{3}{2}N(N+1)$ operations. After this initial investment, a single polynomial evaluation is quite efficient if we rewrite the interpolation polynomial as,

$$p_N(x) = a_0 + (x-x_0)[a_1 + (x-x_1)[a_2 + (x-x_2)[\cdots[a_{N-1} + (x-x_{N-1})a_N]\cdots].$$

The evaluation starts at the innermost bracket, and the cost of a single evaluation is $3N$. In addition, should we decide to include additional interpolation points, it is not necessary to recalculate the entire difference table. Assuming that we kept the previously calculated values, we simply add the new point to the bottom of the table and calculate a single new diagonal.

12.4. Interpolation error and accuracy.

Given a smooth function $f(x)$ (a function with infinitely many derivatives) and its interpolation polynomial $p_N(x)$ at the points $x_j, \quad j = 0,\ldots,N$, i.e. $p_N(x_j) = f(x_j), \quad j = 0,\ldots,N$, the question is what is the error, $e(x) = f(x) - p_N(x)$? We

obtain an expression for the error by means of an intuitive argument. There are three basic considerations.

(1) Since the error is zero at the interpolation points, one can factor out a term of the form $w(x) = (x - x_0) \cdots (x - x_N)$.
(2) Since we know that the interpolation polynomial is unique, it follows that $e(x)$ should be identically zero whenever $f(x)$ is a polynomial of degree no higher than N. One therefore expects the error to also contain a term of the form $f^{(N+1)}(x)$. Since it is not clear where we need to evaluate $f^{(N+1)}(x)$ we express this uncertainty through an unknown function $\xi(x)$. Thus we use $f^{(N+1)}(\xi(x))$.
(3) If $f(x)$ is a polynomial of degree $N+1$, say $f(x) = x^{N+1}$, $e(x)$ is a polynomial of degree $N + 1$ with leading coefficient 1.

Putting all of this together, with $w(x)$ as given above, we find that

$$e(x) = w(x)\frac{f^{(N+1)}(\xi(x))}{(N+1)!}, \tag{12.1}$$

where $\xi(x)$ is unknown, except that it takes one some value inside the smallest interval containing x and the interpolation points.

It should be clear that the error depends on the nature of $f^{(N+1)}(x)$ as well as the interpolation points, through the factor $w(x)$. Given a function $f(x)$ there nothing one can do to control its derivatives. One can therefore try and reduce the error by a judicious choice of the interpolation points when possible, and/or by increasing the number of interpolation points. Let us investigate these possibilities by means of an example.

EXAMPLE 37. *The Runge phenomenon.* Let us interpolate

$$f(x) = \frac{1}{1 + k^2x^2} \tag{12.2}$$

over the interval $[-1, 1]$ for different values of k using equispaced interpolation points, $x_j = -1 + j\frac{2}{N}, \quad j = 0, \ldots, N$. We fix $N = 10$ and calculate the interpolation

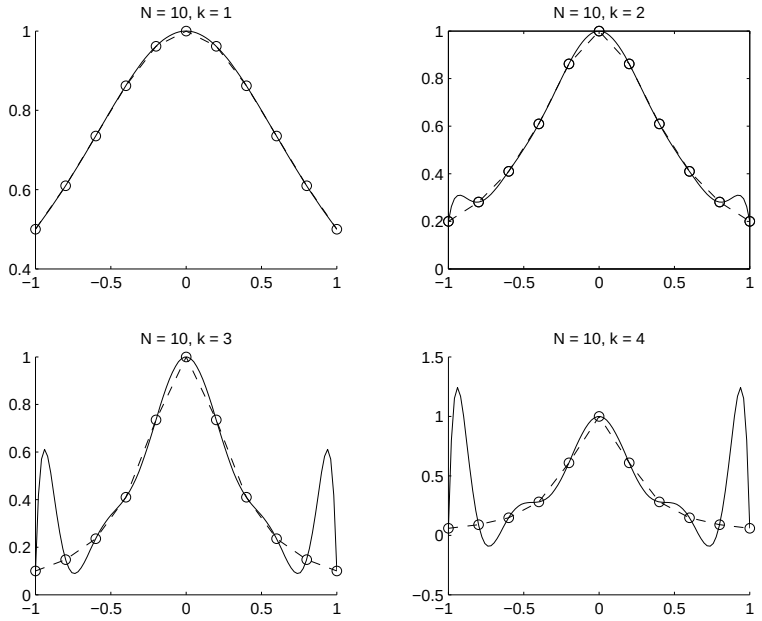


FIGURE 12.4.1. Equidistant interpolation.

polynomial for different values of k as shown in Figure 12.4.1. It is perhaps a little surprising that the interpolation deteriorates for increasing k. Let us try and reduce the error for $k = 3$ by increasing the number of interpolation points, as shown in Figure 12.4.2. Rather unexpectedly, there is no improvement in the error. In fact the error close to the edges of the interval is worse than before. □

We cannot give a full explanation of what is going wrong here. Partly it is due to the singularity at $x = \frac{\pm i}{k}$ in the complex plane, partly due to our choice of equidistant interpolation points. As the singularity moves closer to the real axis by increasing k, it has a greater effect on the quality of the interpolation polynomial. In order to illustrate the possible role of the choice of interpolation points, we develop a little more theory.

Suppose that among all the polynomials of degree not greater than N, interpolating at all possible choices of $N+1$ interpolation points, we choose the one with the smallest maximum error. It can be shown that this polynomial exists and is unique; constructing it is a different matter altogether. We want to compare the error in any given interpolation polynomial with this optimal error.

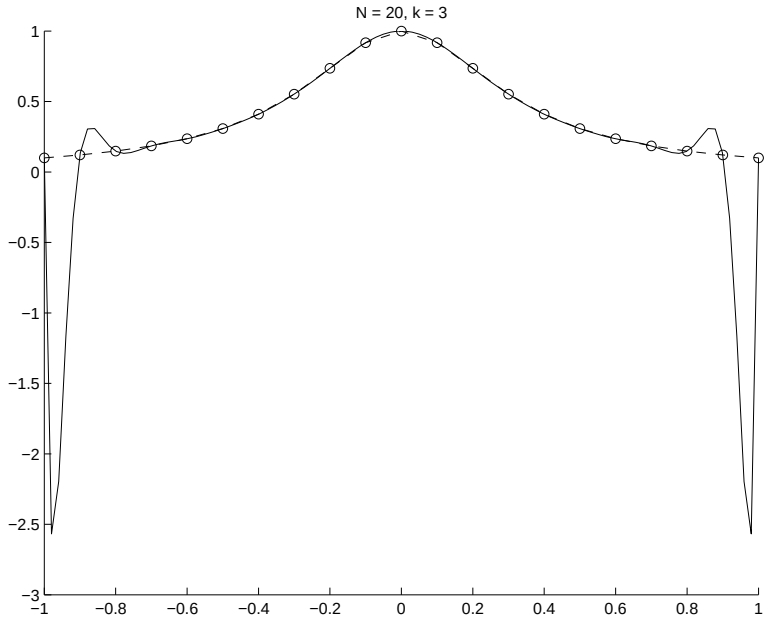


FIGURE 12.4.2. Increasing the number of interpolation points.

Let P denote the interpolation operator, i.e. $P : f(x) \longrightarrow p_N(x)$. Note that P is a linear operator satisfying the projection property, $P^2 = P$ which simply states that a polynomial interpolates itself. Denoting the optimal interpolation operator by $P_{\text{opt}}f$, we have

$$\begin{aligned} f - Pf &= f - P_{\text{opt}}f + P_{\text{opt}}f - Pf \\ &= f - P_{\text{opt}}f - P(f - P_{\text{opt}}f). \end{aligned}$$

Using the triangle inequality as well as the properties of the operator norm, it follows that

$$\|f - Pf\| \leq (1 + \|P\|)\|f - P_{\text{opt}}f\|, \tag{12.3}$$

where $\|P\|$ is known as the Lebesgue constant and it gives in idea of how well a particular interpolation scheme performs in comparison with the best possible.

If we use the ∞-norm the Lebesgue constant of the Lagrange interpolation polynomial is given by

$$\|P\|_\infty = \max_{a\le x\le b} \sum_{k=0}^{N} |L_k(x)|. \tag{12.4}$$

This easily follows from the definition of an operator norm,

$$\begin{aligned}
\|P\|_\infty := & \sup_{\|f\|_\infty=1} \|Pf\|_\infty \\
= & \sup_{\|f\|_\infty=1} \max_{a\le x\le b} \left| \sum_{k=0}^{N} f(x_k) L_k(x) \right| \\
= & \max_{a\le x\le b} \sup_{\|f\|_\infty=1} \left| \sum_{k=0}^{N} f(x_k) L_k(x) \right| \\
= & \max_{a\le x\le b} \sum_{k=0}^{N} |L_k(x)|.
\end{aligned}$$

For equispaced points, it is possible to show that

$$\|P\| = O\left(\frac{2^N}{N \ln(N)} \right), \tag{12.5}$$

clearly suggesting that equispaced points are far from optimal since the bound grows exponentially with increasing order of the interpolation polynomial.

These ideas can be illustrated by looking graphically at the functions $L_k(x)$ in Lagrange's interpolation formula. In the top half of Figure 12.4.3, we see from front to back $L_k(x),\ k = 0, 1, \ldots, N$ (with $N = 10$) in the case that the interpolation points x_k ($k = 0, 1, \ldots, N$) are equispaced over [-1,1]. Apart from the little 'hump' of $L_k(x)$ at $x = x_k$, we notice very big edge oscillations for some of the functions. Linearly combining these $L_k(x)$ according to the Lagrange interpolation formula, is therefore likely to leave $p_N(x)$ also badly oscillatory near the ends of the interval. We can make this problem disappear by simply choosing our sampling points denser toward the end of the interval. In Figure 12.4.3(bottom), we see that the use of *Chebyshev*

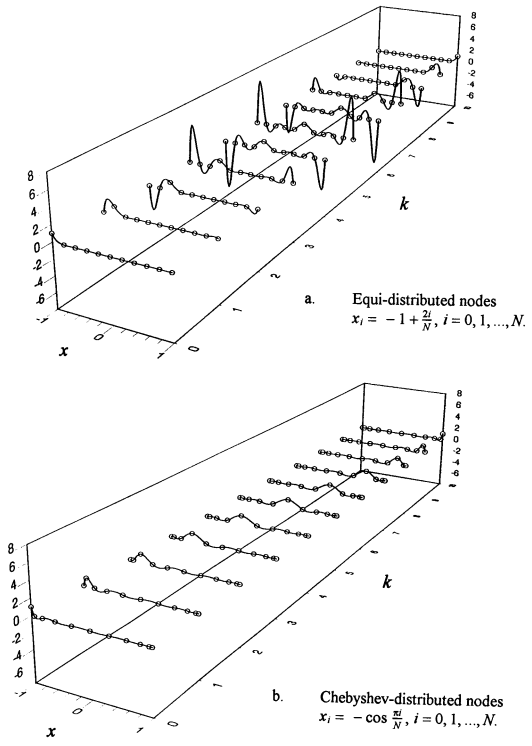


FIGURE 12.4.3. Basis functions $L_k(x)$ in Lagrange's interpolation polynomial for two different node distributions, $N = 10$. (a) Equi-spaced nodes. (b) Chebyshev distributed nodes $x_k = -\cos(\pi k/N), k = 0, \ldots, N$.

nodes $x_k = -\cos(\pi k/N)$ makes $L_k(x)$ perfectly well behaved. In this case, $p_N(x)$ converges to $f(x)$ whenever this is a continuous function, and very rapidly if $f(x)$ is many times differentiable.

Adding the Lagrange functions shown in Figure 12.4.3,

$$L(x) = \sum_{k=0}^{N} |L_k(x)|, \quad x \in [-1, 1]\,, \tag{12.6}$$

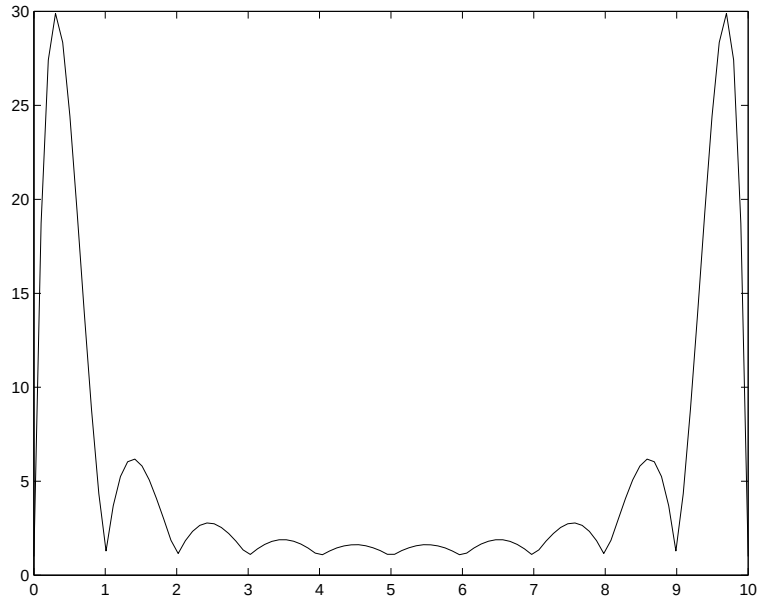


FIGURE 12.4.4. The Lebesgue function: equidistant points.

one obtains Figure 12.4.4 for $N = 10$ equidistant points, and Figure 12.4.5 for $N = 10$ Chebyshev points. It is straightforward to read the value of the Lebesgue constant from these graphs.

All of this suggest that Chebyshev points are really much better than equidistant points. Indeed, if we use $N = 10$ Chebyshev points to interpolate the Runge example above with $k = 3$, Figure 12.4.6 shows a significant improvement. In general, one can show that the Lebesgue constant for Chebyshev points is given by,

$$\|P\| = O(\ln(N)). \tag{12.7}$$

The convergence rate is clearly x-dependent—even for equispaced interpolation points the error in the Runge example is always well-behaved in the interior of the interval. One can show (see for example Fornberg, 1996) that the error behaves like $O(\alpha(x)^N)$ where

$$\alpha(x) = e^{\frac{1}{2}(1-x)\ln(1-x)+\frac{1}{2}(1+x)\ln(1+x)\,+\,C}. \tag{12.8}$$

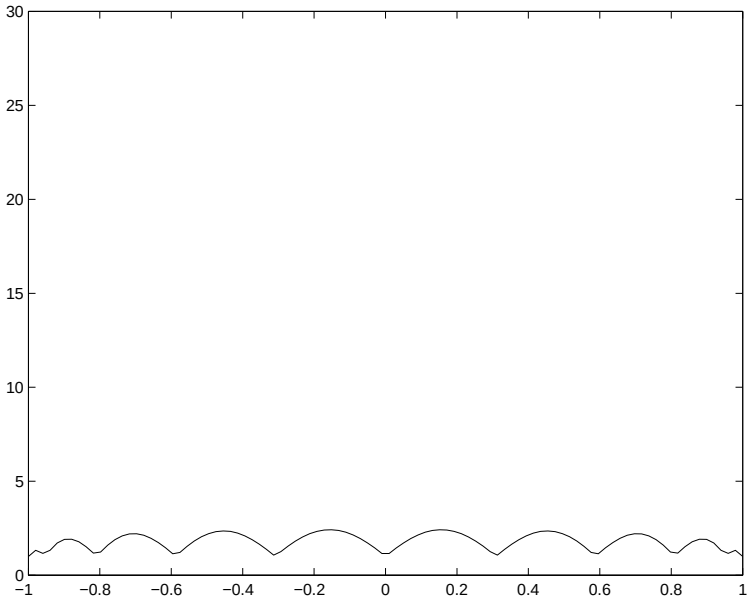


FIGURE 12.4.5. The Lebesgue constant: Chebyshev points.

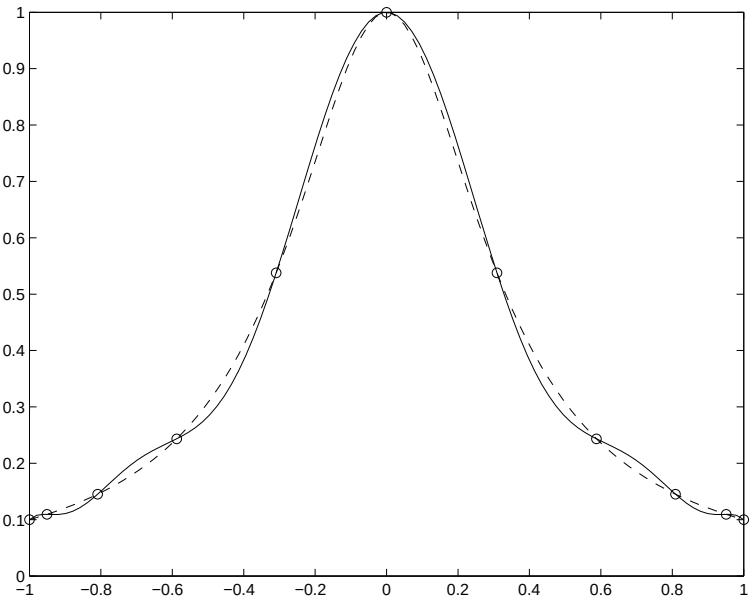


FIGURE 12.4.6. Runge example: Chebyshev points.

The function $f(x)$ affects only the value of the constant C. In the case of the Runge example, $f(x) = 1/(1 + x^2)$, one can find that $C = -\frac{1}{2}\ln\frac{17}{16} - 14\arctan 4$. It is then easy to verify that $\alpha(x)$ passes the value one at $x \approx \pm 0.7942$. We indeed notice in Figures 12.4.1 and Figure 12.4.2 a transition between convergence and divergence (as N increases) just at these locations.

In the Chebyshev case, the relation (12.8) simplifies all the way down to

$$\alpha(x) = e^C$$

where C is a non-positive constant. Uniform convergence is now assured for all continuous functions $f(x)$—and is faster (C is more negative) the smoother $f(x)$ is.

12.5. Finite difference formulas.

12.5.1. Some elementary ways to derive finite difference (FD) formulas. An FD formula approximates a derivative as a linear combination of some neighboring function values. We consider first a specific case, and note a few ways one might use to obtain the *weights* (coefficients) that the function values need to be multiplied with to make the derivative approximation exact for as high degree polynomials as possible.

Example: Approximate $f'(x_0)$ based on function values at the three points $x_0 - h$, x_0 and $x_0 + h$, i.e. determine the best weights c_{-1}, c_0 and c_1 in the approximation

$$f'(x_0) \approx c_{-1}f(x_0 - h) + c_0 f(x_0) + c_1 f(x_0 + h).$$

Solution 1: The value of x_0 has no influence, so we set $x_0 = 0$ in order to keep the notation simple. The Lagrange polynomial that takes the desired values at $x = -h, 0, h$ becomes

$$p_2(x) = f(-h)\frac{(x-0)(x-h)}{(-h-0)(-h-h)} + f(0)\frac{(x+h)(x-h)}{(0+h)(0-h)} + f(h)\frac{(x+h)(x-0)}{(h+h)(h-0)}$$

Differentiating with respect to x and then setting $x = 0$ gives

$$p_2'(0) = -\frac{1}{2h}f(-h) + 0\ f(0) + \frac{1}{2h}f(h) \ ,$$

i.e. the desired weights are $c_{-1} = -\frac{1}{2h}$, $c_0 = 0$, $c_1 = \frac{1}{2h}$.

Solution 2: We express $f(-h)$, $f(0)$ and $f(h)$ by means of Taylor expansions around $x = 0$

$$\left\{ \begin{array}{ll} f(-h) & = f(0) - \frac{h}{1!}f'(0) + \frac{h^2}{2!}f''(0) - \ldots \\ f(0) & = f(0) \\ f(h) & = f(0) + \frac{h}{1!}f'(0) + \frac{h^2}{2!}f''(0) - \ldots \end{array} \right. .$$

We want

$$c_{-1}f(-h) + c_0 f(0) + c_1 f(h) = 0\ f(0) + 1\ f'(0) + 0\ f''(0) + \ldots$$

Plugging in the expansions above and equating coefficients for $f(0)$, $f'(0)$ and $f''(0)$ gives a linear system for the three unknowns

$$\left\{ \begin{array}{rrrl} c_{-1} & +c_0 & +c_1 & = 0 \\ -\frac{h}{1!}\ c_{-1} & & +\frac{h}{1!}\ c_1 & = 1 \\ \frac{h^2}{2!}c_{-1} & & +\frac{h^2}{2!}c_1 & = 0 \end{array} \right. .$$

This has the same solution as before: $c_{-1} = -\frac{1}{2h}$, $c_0 = 0$, $c_1 = \frac{1}{2h}$.

Solution 3: We want the formula

$$f'(0) \approx c_{-1}f(-h) + c_0 f(0) + c_1 f(h)$$

to be exact for as high degree polynomials as possible. We enforce it in turn for $f = 1$, $f = x$ and $f = x^2$ and obtain essentially the same system to solve as in the previous solution.

$$\begin{array}{lcrrrl} f = 1 & \Rightarrow & c_{-1} & + c_0 & +c_1 & = 0 \\ f = x & \Rightarrow & c_{-1}(-h) & & +c_1(h) & = 1 \\ f = x^2 & \Rightarrow & c_{-1}(-h)^2 & & +c_1(h)^2 & = 0 \end{array} \ .$$

□

Tediously, one can use either of the three approaches above to generate tables of weights, such as seen in Table 12.5.2. In the first of these tables, we recognize the approximation we have just derived as the top line of weights $-\frac{1}{2}\ 0\ \frac{1}{2}$ (a division by

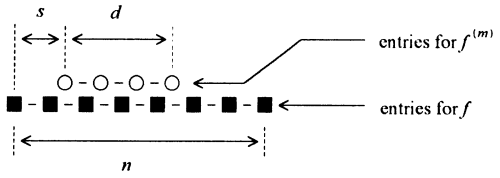


FIGURE 12.5.1. Illustration of the notation used for the Pade weight algorithm.

h^m is also assumed when calculating an m^{th} derivative). A much more convenient procedure is given next.

12.5.2. Padé-based routine for finding FD formulas on an equispaced grid. Symbolic algebra packages such as Mathematica or Maple can be very convenient for carrying out and simplifying many difficult or tedious tasks in calculus and algebra. In the case of finding weights for FD formulas, particular advantages of such packages include the availability of

- exact arithmetic—here in the form of rational numbers, and
- advanced commands, like analytical Taylor or Padé expansion of a function.

Table 1 gives five examples of equispaced FD approximations that are needed in different situations (to approximate PDEs in space, advance ODEs in time etc.) We can illustrate the general form of stencils like these in the way that is shown in Figure 12.5.1. The four numbers d, s, n and m entirely describe the FD stencil we are interested in. The five cases described in Table 1 amount to choosing these numbers as shown in Table 12.5.4. Given just these numbers, the task is to generate all the weights needed for the corresponding FD formulas. The following very short codes achieve this (first presented in Fornberg [**?**]):

Mathematica:

$t =$`Pade`$[x^s$`Log`$[x]^m, \{x, 1, n, d\}]$;
`{CoefficientList[Denominator[`t`],`x`],CoefficientList[Numerator[`t`],`x`]/`h^m`}`

Description	Finite difference formula	Error
centered, regular grid	$$f''(x) \approx \left[-\frac{1}{12}f(x-2h)+\frac{4}{3}f(x-h)-\frac{5}{2}f(x) + \frac{4}{3}f(x+h)-\frac{1}{12}f(x+2h)\right]/h^2$$	$O(h^4)$
staggered, regular grid	$$f'(x) \approx \left[\frac{1}{24}f(x-\frac{3}{2}h)-\frac{9}{8}f(x-\frac{1}{2}h) + \frac{9}{8}f(x+\frac{1}{2}h)-\frac{1}{24}f(x+\frac{3}{2}h)\right]/h$$	$O(h^4)$
one-sided, regular grid	$$f'(x+h) \approx \left[\frac{1}{4}f(x-3h)-\frac{4}{3}f(x-2h)+3f(x-h) - 4f(x)+\frac{25}{12}f(x+h)\right]/h$$	$O(h^4)$
implicit, regular grid	$$\frac{1}{12}f''(x-h)+\frac{5}{6}f''(x)+\frac{1}{12}f''(x+h) \approx [1f(x-h)-2f(x)+1f(x+h)]/h^2$$	$O(h^4)$
implicit, one-sided, regular grid	$$-\frac{3}{8}f'(x-3h)+\frac{37}{24}f'(x-2h)-\frac{59}{24}f'(x-h)+\frac{55}{24}f'(x) \approx [-f(x)+f(x+h)]/h$$	$O(h^4)$

TABLE 1. Some illustrative examples of FD formulas.

In the output, the variable h will denote the grid spacing—no explicit value needs to be entered for it in advance. This brief code uses Mathematica's routine Padé, which is not loaded automatically. So before

running the code, a package containing it must be loaded by the command <<Calculus‘Pade‘ . In the case illustrated in Figure 12.5.1 and choosing $m = 1$, the Mathematica output becomes

$$\{\{0,\ \tfrac{49}{288},\ \tfrac{95}{144},\ \tfrac{49}{288}\},$$
$$\{-\tfrac{64925}{110592\,h},\ \tfrac{64925}{110592\,h},\ \tfrac{78841}{552960\,h},\ -\tfrac{343}{110592\,h},\ \tfrac{43}{430080\,h}\}\}$$

It is interesting to note that best accuracy is achieved when the leftmost derivative entry is omitted (has weight zero).

Maple:

$t :=$`pade`$(x\hat{}s * \ln(x)\hat{}m, x = 1, [n, d])$:

`coeff` (`expand` (`denom`$(t)), x, i)$ $\$i = 0..d$;

`coeff` (`expand` (`numer`$(t)), x, i)/h\hat{}m$ $\$i = 0..n$;

The package containing the routine pade needs again to be pre-loaded; the syntax now is with (numapprox): .

12.5.3. Derivation of Padé routine for FD weights. One example suffices to illustrate why the Padé algorithm works. Consider for example case 4 from Tables 1 and 12.5.4. We want to find the coefficients which make a stencil

$$b_0 f''(x-h) + b_1 f''(x) + b_2 f''(x+h) \approx c_0 f(x-h) + c_1 f(x) + c_2 f(x+h)$$

accurate for as high degree polynomials as possible. Substituting $f(x) = e^{i\omega x}$ gives

$$-\omega^2 \left[b_0 e^{-i\omega h} + b_1 + b_2 e^{i\omega h}\right] e^{i\omega x} \approx \left[c_0 e^{-i\omega h} + c_1 + c_2 e^{i\omega h}\right] e^{i\omega x} \quad .$$

Case	d	s	n	m
1	0	2	4	2
2	0	3/2	3	1
3	0	4	4	1
4	2	0	2	2
5	3	-3	1	1

TABLE 2. Input parameters required for the symbolic algebra algorithm to generate the FD formulas in Table 12.5.4

The goal is to make the approximation as accurate as possible, if expanded locally around $\omega = 0$. Canceling the factors $e^{i\omega x}$ and substituting $e^{i\omega h} = \xi$, i.e. $i\omega h = \ln \xi$ gives

$$\left\{\frac{\ln \xi}{h}\right\}^2 \left[\frac{b_0}{\xi} + b_1 + b_2\xi\right] \approx \left[\frac{c_0}{\xi} + c_1 + c_2\xi\right] \quad ,$$

$$\left\{\frac{\ln \xi}{h}\right\}^2 \approx \frac{c_0 + c_1\xi + c_2\xi^2}{b_0 + b_1\xi + b_2\xi^2} \quad .$$

At this point, we want the best possible accuracy when expanded around $\xi = 1$. Padé approximation of $\left\{\frac{\ln \xi}{h}\right\}^2$ around $\xi = 1$ to order [2,2] will offer this:

$$\left\{\frac{\ln \xi}{h}\right\}^2 \approx \frac{(\xi-1)^2}{h^2(1 + (\xi-1) + \frac{1}{12}(\xi-1)^2)} = \frac{1 - 2\xi + 1\xi^2}{h^2(\frac{1}{12} + \frac{5}{6}\xi + \frac{1}{12}\xi^2)}.$$

The weights follow from just picking off the coefficients in the numerator and denominator.

12.5.4. Some convenient tables of FD weights.

12.5.5. FD formulas on arbitrarily spaced grids. The three methods in Subsection 12.5.4 to obtain FD weights generalize - tediously - to the case of non-equispaced grids, whereas the Padé method does not. A very simple algorithm for arbitrarily-spaced 1-D grids was discovered in 1988 [??], [??]. Starting from Lagrange's interpolation formula, it turns out that the weights can be calculated as is implemented in the following Matlab code:

```
function c=weights(z,x,m) % Calculates FD weights. The parameters are:
% z location where approximations are to be accurate,
% x vector with x-coordinates for grid points,
% m highest derivative that we want to find weights for
% c array size m+1,length(x) containing (as output) in
% successive rows the weights for derivatives 0,1,...,m.
n=length(x); c=zeros(m+1,n); c1=1; c4=x(1)-z; c(1,1)=1; for i=2:n
mn=min(i,m+1); c2=1; c5=c4; c4=x(i)-z; for j=1:i-1 c3=x(i)-x(j); c2=c2{*}c3;
if j==i-1 c(2:mn,i)=c1{*}((1:mn-1)'.{*}c(1:mn-1,i-1)-c5{*}c(2:mn,i-1))/c2;
c(1,i)=-c1{*}c5{*}c(1,i-1)/c2; end c(2:mn,j)=(c4{*}c(2:mn,j)-(1:mn-1)'.{*}c(1:mn-1,j)
```

TABLE 1

Weights for some centered FD formulas on an equi-spaced grid

Order of accuracy	Approximations at $x = 0$; x coordinates at nodes:								
	-4	-3	-2	-1	0	1	2	3	4
0th derivative									
∞					1				
1st derivative									
2				$-\frac{1}{2}$	0	$\frac{1}{2}$			
4			$\frac{1}{12}$	$-\frac{2}{3}$	0	$\frac{2}{3}$	$-\frac{1}{12}$		
6		$-\frac{1}{60}$	$\frac{3}{20}$	$-\frac{3}{4}$	0	$\frac{3}{4}$	$-\frac{3}{20}$	$\frac{1}{60}$	
8	$\frac{1}{280}$	$-\frac{4}{105}$	$\frac{1}{5}$	$-\frac{4}{5}$	0	$\frac{4}{5}$	$-\frac{1}{5}$	$\frac{4}{105}$	$-\frac{1}{280}$
2nd derivative									
2				1	-2	1			
4			$-\frac{1}{12}$	$\frac{4}{3}$	$-\frac{5}{2}$	$\frac{4}{3}$	$-\frac{1}{12}$		
6		$\frac{1}{90}$	$-\frac{3}{20}$	$\frac{3}{2}$	$-\frac{49}{18}$	$\frac{3}{2}$	$-\frac{3}{20}$	$\frac{1}{90}$	
8	$-\frac{1}{560}$	$\frac{8}{315}$	$-\frac{1}{5}$	$\frac{8}{5}$	$-\frac{205}{72}$	$\frac{8}{5}$	$-\frac{1}{5}$	$\frac{8}{315}$	$-\frac{1}{560}$
3rd derivative									
2			$-\frac{1}{2}$	1	0	-1	$\frac{1}{2}$		
4		$\frac{1}{8}$	-1	$\frac{13}{8}$	0	$-\frac{13}{8}$	1	$-\frac{1}{8}$	
6	$-\frac{7}{240}$	$\frac{3}{10}$	$-\frac{169}{120}$	$\frac{61}{30}$	0	$-\frac{61}{30}$	$\frac{169}{120}$	$-\frac{3}{10}$	$\frac{7}{240}$
4th derivative									
2			1	-4	6	-4	1		
4		$-\frac{1}{6}$	2	$-\frac{13}{2}$	$\frac{28}{3}$	$-\frac{13}{2}$	2	$-\frac{1}{6}$	
6	$\frac{7}{240}$	$-\frac{2}{5}$	$\frac{169}{60}$	$-\frac{122}{15}$	$\frac{91}{8}$	$-\frac{122}{15}$	$\frac{169}{60}$	$-\frac{2}{5}$	$\frac{7}{240}$

FIGURE 12.5.2. FD weights.

TABLE 2
Weights for some FD formulas centered half-way between grid points

Order of accuracy	Approximations at $x = 0$; x coordinates at nodes:							
	$-\frac{7}{2}$	$-\frac{5}{2}$	$-\frac{3}{2}$	$-\frac{1}{2}$	$\frac{1}{2}$	$\frac{3}{2}$	$\frac{5}{2}$	$\frac{7}{2}$
0th derivative								
2				$\frac{1}{2}$	$\frac{1}{2}$			
4			$-\frac{1}{16}$	$\frac{9}{16}$	$\frac{9}{16}$	$-\frac{1}{16}$		
6		$\frac{3}{256}$	$-\frac{25}{256}$	$\frac{75}{128}$	$\frac{75}{128}$	$-\frac{25}{256}$	$\frac{3}{256}$	
8	$-\frac{5}{2048}$	$\frac{49}{2048}$	$-\frac{245}{2048}$	$\frac{1225}{2048}$	$\frac{1225}{2048}$	$-\frac{245}{2048}$	$\frac{49}{2048}$	$-\frac{5}{2048}$
1st derivative								
2				- 1	1			
4			$\frac{1}{24}$	$-\frac{9}{8}$	$\frac{9}{8}$	$-\frac{1}{24}$		
6		$-\frac{3}{640}$	$\frac{25}{384}$	$-\frac{75}{64}$	$\frac{75}{64}$	$-\frac{25}{384}$	$\frac{3}{640}$	
8	$\frac{5}{7168}$	$-\frac{49}{5120}$	$\frac{245}{3072}$	$-\frac{1225}{1024}$	$\frac{1225}{1024}$	$-\frac{245}{3072}$	$\frac{49}{5120}$	$-\frac{5}{7168}$

FIGURE 12.5.3. FD weights.

```
c(1,j)=c4{*}c(1,j)/c3; end c1=c2; end
```

To reproduce for ex. the formulas that extend over the 9 points [-4:4] in Table 1 in the previous section, we would call this routine weights as follows:

```
 function c = weights(0,-4:4,4)
```

giving the output

```
 c = 0 0 0 0 1.0000
0 0 0 0 0.0036 -0.0381 0.2000 -0.8000 -0.0000 0.8000 -0.2000 0.0381
-0.0036 -0.0018 0.0254 -0.2000 1.6000 -2.8472 1.6000 -0.2000 0.0254
-0.0018 -0.0292 0.3000 -1.4083 2.0333 -0.0000 -2.0333 1.4083 -0.3000
```

TABLE 3

Weights for some one-sided FD formulas on an equi-spaced grid

Order of accuracy	Approximations at $x = 0$; x coordinates at nodes								
	0	1	2	3	4	5	6	7	8
0th derivative									
∞	1								
1st derivative									
1	-1	1							
2	$-\frac{3}{2}$	2	$-\frac{1}{2}$						
3	$-\frac{11}{6}$	3	$-\frac{3}{2}$	$\frac{1}{3}$					
4	$-\frac{25}{12}$	4	-3	$\frac{4}{3}$	$-\frac{1}{4}$				
5	$-\frac{137}{60}$	5	-5	$\frac{10}{3}$	$-\frac{5}{4}$	$\frac{1}{5}$			
6	$-\frac{49}{20}$	6	$-\frac{15}{2}$	$\frac{20}{3}$	$-\frac{15}{4}$	$\frac{6}{5}$	$-\frac{1}{6}$		
7	$-\frac{363}{140}$	7	$-\frac{21}{2}$	$\frac{35}{3}$	$-\frac{35}{4}$	$\frac{21}{5}$	$-\frac{7}{6}$	$\frac{1}{7}$	
8	$-\frac{761}{280}$	8	-14	$\frac{56}{3}$	$-\frac{35}{2}$	$\frac{56}{5}$	$-\frac{14}{3}$	$\frac{8}{7}$	$-\frac{1}{8}$
2nd derivative									
1	1	-2	1						
2	2	-5	4	-1					
3	$\frac{35}{12}$	$-\frac{26}{3}$	$\frac{19}{2}$	$-\frac{14}{3}$	$\frac{11}{12}$				
4	$\frac{15}{4}$	$-\frac{77}{6}$	$\frac{107}{6}$	-13	$\frac{61}{12}$	$-\frac{5}{6}$			
5	$\frac{203}{45}$	$-\frac{87}{5}$	$\frac{117}{4}$	$-\frac{254}{9}$	$\frac{33}{2}$	$-\frac{27}{5}$	$\frac{137}{180}$		
6	$\frac{469}{90}$	$-\frac{223}{10}$	$\frac{879}{20}$	$-\frac{949}{18}$	41	$-\frac{201}{10}$	$\frac{1019}{180}$	$-\frac{7}{10}$	
7	$\frac{29531}{5040}$	$-\frac{962}{35}$	$\frac{621}{10}$	$-\frac{4006}{45}$	$\frac{691}{8}$	$-\frac{282}{5}$	$\frac{2143}{90}$	$-\frac{206}{35}$	$\frac{363}{560}$

FIGURE 12.5.4. FD weights.

```
0.0292 0.0292 -0.4000 2.8167 -8.1333 11.3750 -8.1333 2.8167 -0.4000
0.0292
```

Although in this example, the grid was equi-spaced and the point of approximation (z = 0) happened to coincide with a grid point, neither of this was required. This algorithm is flexible, fast, and numerically stable. Especially when working in Matlab, it is generally the preferable one to use for calculating FD weights (including weights to use for interpolation).

12.5.6. FD formulas in 2-D.. The two main tasks are

(1) approximate some derivative (say $\frac{\partial^3}{\partial x \partial y^2}$) at a grid point, or
(2) approximate some derivative but now also including interpolation which we can see as $\frac{\partial^0}{\partial x^0 \partial y^0}$ at arbitrary in-between grid point locations. This of course, includes interpolation.

The procedure is essentially the same in both cases. Starting with the first case (just to keep the notation marginally simpler); let h and k be the grid spacings in the x– and y–directions respectively. From the tables in the previous subsection, we see that we can approximate to fourth order

$$\frac{\partial}{\partial x} \approx \left[\frac{1}{12} \;\; -\frac{2}{3} \;\; 0 \;\; \frac{2}{3} \;\; -\frac{1}{12}\right]/h \text{ and } \frac{\partial^2}{\partial y^2} \approx \left[-\frac{1}{12} \;\; \frac{4}{3} \;\; -\frac{5}{2} \;\; \frac{4}{3} \;\; -\frac{1}{12}\right]/k^2$$

The corresponding 2-D stencil can then be represented by the matrix

$$\frac{\partial^3}{\partial x \partial y^2} \approx \begin{bmatrix} -\frac{1}{12} \\ \frac{4}{3} \\ -\frac{5}{2} \\ \frac{4}{3} \\ -\frac{1}{12} \end{bmatrix} \begin{bmatrix} \frac{1}{12} & -\frac{2}{3} & 0 & \frac{2}{3} & -\frac{1}{12} \end{bmatrix}/(k^2 h) = \begin{bmatrix} -\frac{1}{144} & \frac{1}{18} & 0 & -\frac{1}{18} & \frac{1}{144} \\ \frac{1}{9} & -\frac{8}{9} & 0 & \frac{8}{9} & -\frac{1}{9} \\ -\frac{5}{24} & \frac{5}{3} & 0 & -\frac{5}{3} & \frac{5}{24} \\ \frac{1}{9} & -\frac{8}{9} & 0 & \frac{8}{9} & -\frac{1}{9} \\ -\frac{1}{144} & \frac{1}{18} & 0 & -\frac{1}{18} & \frac{1}{144} \end{bmatrix}/(k^2 h) \,.$$

This is best implemented without ever forming the product above (there is no need to spend n^2 storage locations on keeping the elements of the rank 1 matrix). We can just as well apply the two 1-D stencils in succession to the data. For example, we can apply the x-derivative stencil at five consecutive y-levels, and then weigh together these results according to the weights in the y-stencil. Or reverse the order and

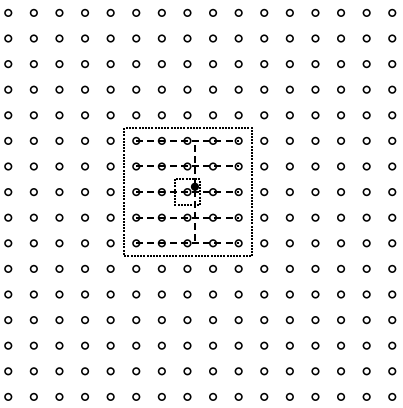


FIGURE 12.5.5. Illustration of how to apply 1-D FD formulas to calculate a 2-D derivative at an in-between grid point location.

implement five y-derivatives first and conclude with the x-derivative on the results. In either case, the result would be the same as if we had used the full $n \times n$ matrix.

If we need to approximate a derivative in-between grid points, as shown in Figure 12.5.5, the procedure would be essentially identical. The Figure illustrates how one might approximate x-derivatives first at five grid lines and conclude with an approximation in the y-direction. For this, we need to find the FD weights for approximation at a location other than a grid point. The Padé algorithm does this just fine, but gets a bit slow if one needs to do this for a very large number of places. A computationally much faster method (which also works for non-equispaced grids) is given in the exercises.

12.6. Splines.

In a previous section we noted that one has to take care while increasing the degree of the interpolation polynomial. There is however, a way of increasing the accuracy of the approximation without increasing the order of the interpolation polynomial. The idea is to use a low order interpolation polynomial and apply this polynomial

over short intervals of the approximation range. Thus, if we choose piecewise linear interpolation polynomials, we simply connect (interpolate) the interpolation points with straight lines, as in our example of the Introduction. Since the error for piecewise linear polynomials, interpolating $f(x), \quad x \in [x_j, x_{j+1}]$ is given by,

$$e(x) = \frac{1}{2}(x - x_j)(x - x_{j+1}) f^{(2)}(\xi) \tag{12.1}$$

one finds that

$$\max_{x \in [x_j, x_{j+1}]} |e(x)| \leq \frac{1}{8} h_j^2 M_j \tag{12.2}$$

where $h_j = x_{j+1} - x_j$, and $|f^{(2)}(x)| \leq M_j$ for $x \in [x_j, x_{j+1}]$. Thus provided that $|f^{(2)}(x)|$ is bounded, one can decrease the error by decreasing h_j, perhaps by increasing the number of interpolation points. On the other hand, since (12.2) shows that the error is larger in areas where the second derivative is large one can also try and reduce the overall error by using the same number of interpolation points, but placing them according to

$$h_j^2 M_j \approx \text{constant}, \ j = 0, \ldots, N-1,$$

i.e. by placing more interpolation points in areas where the 'curvature' is higher. For one dimensional problems this strategy works really well. Unfortunately its generalization to higher dimensions is not so easy. This topic is further explored in the Chapter on Radial Basis Functions, Chapter 14.

The main problem with piecewise linear interpolation is that the interpolation function is not smooth—it has jumps in its first derivative where the pieces join, and it shows. One can do better by using splines which are designed to have continuous higher order derivatives at the joints.

12.6.1. Splines on uniformly spaced interpolation points. Anticipating the development of subdivision schemes in the next section, we slightly change direction. Instead of using different functions, like the Lagrange functions, to interpolate a given function $f(x)$ at the equidistant points $x_j = j, \quad j = 0, \ldots N$, we now use a

single function, say $B(x)$, to construct the interpolation polynomial $p(x)$ as a linear combination of shifted versions of $B(x)$ itself,

$$p(x) = \sum_{j=0}^{N} c_j B(x-j),$$

where the coefficients are again chosen to satisfy the interpolation condition, $p(j) = f(j), \;\; j = 0, \ldots N$.

It is necessary to point out two facts:

(1) The particular choice of equidistant points is not special. Any set of $N+1$ equidistant points can be rescaled to the interval $[0, N]$. Also we'll shortly remove the restriction to equidistant points.
(2) The coefficients c_j are no longer necessarily equal to $f(x)$ at the interpolation points $x = j$—the coefficients need to be chosen so that the interpolation condition is satisfied.

EXAMPLE 38. Piecewise linear interpolation is obtained by choosing,

$$B(x) = \begin{cases} 1+x, & x \in [-1, 0] \\ 1-x, & x \in [0, 1] \\ 0, & x \notin [-1, 1] \end{cases}.$$

□

A particularly useful set of interpolation functions, the so-called B-splines (B for *Bell*-shaped), are recursively constructed, starting with

$$B_1(x) = \begin{cases} 1, & x \in [-\frac{1}{2}, \frac{1}{2}) \\ 0, & x \notin [-\frac{1}{2}, \frac{1}{2}) \end{cases}. \tag{12.3}$$

Higher order B-splines are then defined as

$$B_m(x) = \int_{-\frac{1}{2}}^{\frac{1}{2}} B_{m-1}(x-t)dt, \tag{12.4}$$

more conveniently written as the convolution

$$B_m(x) = B_{m-1} \star B_1(x), \tag{12.5}$$

where the convolution between two functions, $f(x)$ and $g(x)$ is given by,

$$f \star g(x) = \int_{-\infty}^{\infty} f(t)g(x-t)dt. \tag{12.6}$$

We list the following important properties of B-splines; there is one more, refinability, that we'll get to in the next section:

(1) Over each interval $[j, j+1]$, $B_m(x)$ is a polynomial of degree $m-1$.
(2) B-splines are positive and have finite support: $B_m(x) > 0, \quad x \in (-\frac{1}{2}m, \frac{1}{2}m)$ and $B_m(x) = 0, \quad x \notin (-\frac{1}{2}m, \frac{1}{2}m)$.
(3) Symmetry: $B_m(-x) = B_m(x)$.
(4) Normalization: $\int_{-\infty}^{\infty} B_m(x)dx = 1$.
(5) Partition of unity: $\displaystyle\sum_{k=-\infty}^{\infty} B_m(x-k) = 1$.
(6) Smoothness:

$$B'_m(x) = B_{m-1}(x + \frac{1}{2}) - B_{m-1}(x - \frac{1}{2}). \tag{12.7}$$

This tells us that $B_m(x)$ has $m-2$ continuous derivatives, and $B^{(m-1)}(x)$ is piecewise continuous.
(7) Recurrence relation:

$$B_m(x) = \frac{\frac{1}{2}m + x}{m-1} B_{m-1}(x + \frac{1}{2}) + \frac{\frac{1}{2}m - x}{m-1} B_{m-1}(x - \frac{1}{2}). \tag{12.8}$$

With the exception of (12.8) these properties are more or less straightforwardly derived from its definition (12.4).

Figure 12.6.1 shows the first four B-splines. Incidentally, by looking at the graphs, can you guess what the limiting shape is if we continue the process to higher order splines? (See problem section.)

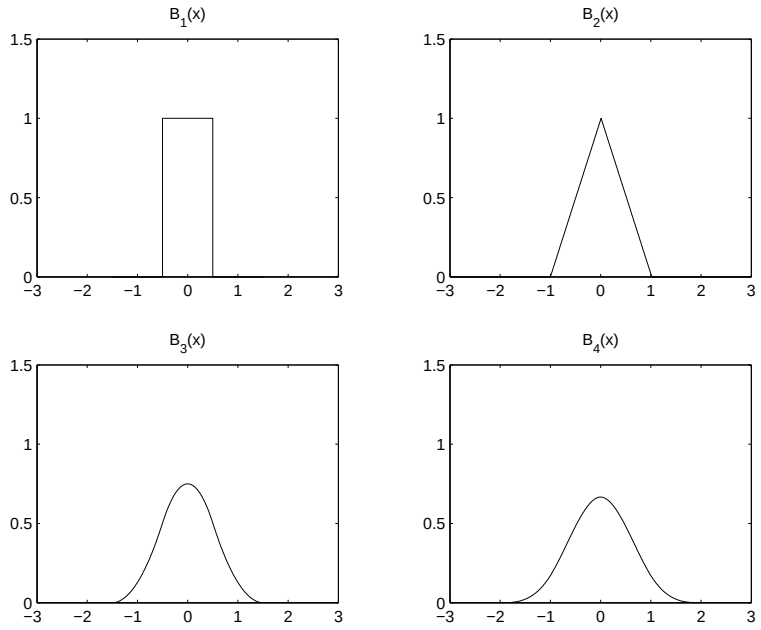


FIGURE 12.6.1. The first four B-splines.

EXAMPLE 39. *Runge example (continued).*

Let us return to the Runge example, $f(x) = 1/(1+9x^2), \quad x \in [-1,1]$ and approximate it with piecewise cubic splines at uniformly spaced interpolation points. If we use $N = 10$ uniformly spaced points it is the easiest to map $[-1,1]$ to $[0,10]$ using $t = 5x + 5$. This amounts to interpolating the function

$$F(x) = \frac{25}{9x^2 - 90x + 250},$$

at the integers, $j = 0, \ldots, 10$. The cubic spline approximation is given by

$$S_3(x) = \sum_{k=0}^{10} c_k B_4(x-k),$$

with $S_3(j) = F(j), \quad j = 0, \ldots, 10$. Since $B_4(x)$ is supported in the interval $(-2,2)$, this gives rise to the following tri-diagonal system of equations for the c_k's,

		$B_1(0)=1$		
$B_3(-1)=\frac{1}{8}$ $B_4(-1)=\frac{1}{6}$	$B_3(-\frac{1}{2})=\frac{1}{2}$	$B_2(0)=0$ $B_3(0)=\frac{3}{4}$ $B_4(0)=\frac{2}{3}$	$B_3(\frac{1}{2})=\frac{1}{2}$	$B_3(1)=\frac{1}{8}$ $B_4(1)=\frac{1}{6}$

TABLE 3. Calculating the splines.

$$F(j)=\sum_{k=j-1}^{j+1} c_k B_4(j-k). \tag{12.9}$$

Thus we need the values of $B_4(x)$ at the integers. From, $B_1(0)=1$ it is easy to compute the following table recursively using (12.8),

The tri-diagonal system (12.9) now becomes,

$$F(0) = \frac{1}{6}c_{-1}+\frac{2}{3}c_0+\frac{1}{6}c_1 \tag{12.10}$$

$$F(j) = \frac{1}{6}c_{j-1}+\frac{2}{3}c_j+\frac{1}{6}c_{j+1},\ j=1,\dots,N-1\,(=9) \tag{12.11}$$

$$F(N) = \frac{1}{6}c_{N-1}+\frac{2}{3}c_N+\frac{1}{6}c_{N+1} \tag{12.12}$$

There is clearly a problem at the boundaries; we need two additional equations in order to determine c_{-1} and c_{11}. There are three popular choices, natural splines, clamped splines, and the preferred, not-a-knot choice.

Natural spline. For this choice one specifies, $S''(0)=0=S''(N)$, to obtain

$$c_{-1}B_4''(1)+c_0B_4''(0)+c_1B_4''(-1)=0=c_{N-1}B_4''(1)+c_NB_4''(0)+c_{N+1}B_4''(-1).$$

Repeated use of the smoothness condition (12.7) gives,

$$\begin{aligned} B_4''(1) &= B_2(2)-2B_2(1)+B_2(0)=1\\ B_4''(0) &= B_2(1)-2B_2(0)+B_2(-1)=-2\\ B_4''(-1) &= B_2(0)-2B_2(-1)+B_2(-2)=1, \end{aligned}$$

so that

$$c_{-1} - 2c_0 + c_1 = 0 = c_{N-1} - 2c_N + c_{N+1}.$$

Substituting this into the tri-diagonal system (12.12), we arrive at

$$\frac{1}{6}\begin{bmatrix} 6 & & & & & \\ 1 & 4 & 1 & & & \\ & \ddots & \ddots & \ddots & & \\ & & 1 & 4 & 1 & \\ & & & & & 6 \end{bmatrix}\begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{N-1} \\ c_N \end{bmatrix} = \begin{bmatrix} F(0) \\ F(1) \\ \vdots \\ F(N-1) \\ F(N) \end{bmatrix}.$$

This is efficiently solved for the coefficients c_j using the specialized tri-diagonal solver of the previous chapter. Since we know the B-splines, one can therefore evaluate $S_3(x)$ at any desired value.

Clamped Spline. For this choice the derivatives of the splines are specified at the endpoints, i.e. $B'(0)$ and $B'(N)$ are specified. The detail are left as an exercise.

Not-a-knot. For this choice one demands that the third derivative is also continuous at 1 and $N-1$. This means that the cubic splines over the first two intervals are one and the same cubic polynomial, and the same for the last two intervals. Thus the second and second last 'knots' have disappeared, hence the name, not-a-knot.

The result of using a natural spline is shown in Figure 12.6.2 where the cubic spline approximation as well as the original function is shown; the two are virtually indistinguishable. □ Redo Figure!

12.6.2. Splines on non-uniformly spaced grids. If the node points are not uniformly spaced, there are two main options:

(1) Introduce a variable spacing $h_j = x_{j+1} - x_j$ in the previous algorithm,
(2) Use parametric splines.

In the case (1), very little changes from before. There will again exist cubic B-splines, again representing the most compact deviation from identical zero of any cubic spline (also now extending over four sub-intervals only). The equivalent linear system to (12.12) will again be tri-diagonal and diagonally dominant. Instead of pursuing the minor differences the deviation from non-uniformity causes in the previous algebra (for references, see [??], [??]), we will instead follow the option (2) - *Parametric*

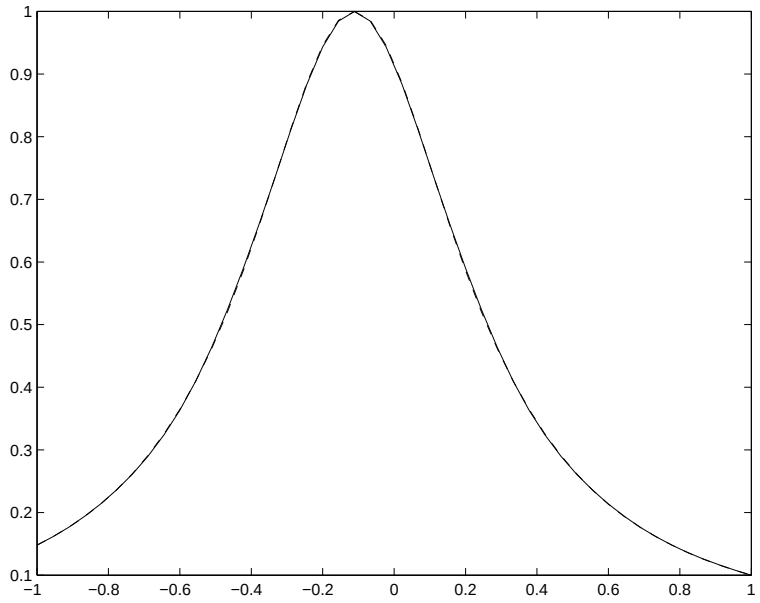


FIGURE 12.6.2. The Runge example using cubic splines.

splines in the next section. However, before doing so, we will show a remarkable result which indicates that cubic splines indeed are very special in providing good approximations to data in 1-D.

THEOREM 40. *Given data u_i at locations x_i, $i = 0, 1, 2, \ldots, n$, the natural cubic spline $u(x)$ (i.e. satisfying $u'' = 0$ at both ends) minimizes $\int_{x_0}^{x_n} (u''(x))^2 dx$ over all possible interpolants to the data.*

Since $u''(x)$ roughly measures the local curvature, minimizing the integral of the square of this quantity will lead to a smooth interpolant which feature the least possible amount of spurious extra oscillations (such as the Gibbs' phenomenon or the Runge phenomenon).

Proof: Let $u(x)$ be the natural cubic spline interpolant to the data. Suppose $s(x) = u(x) + q(x)$ is an interpolant which makes the integral even smaller (i.e. the only constraint on $q(x)$ is that $q(x_i) = 0$, $i = 0, 1, 2, \ldots, n$). Consider $f(\varepsilon) =$

$\int_{x_0}^{x_n}(u''(x)+\varepsilon\, q''(x))^2dx$. This function $f(\varepsilon)$ is a parabola in ε. If we can show that $f\,'(0)=0$, then we are finished. A parabola with a positive coefficient for the quadratic term can have only one minimum, and this result would show that this is taken when the spline interpolant $u(x)$ is not modified in any way.

Differentiation gives

$$\frac{df}{d\varepsilon}=2\int_{x_0}^{x_n}(u''(x)+\varepsilon\, q''(x))\; q''(x)\; dx$$

and

$$\left.\frac{df}{d\varepsilon}\right|_{\varepsilon=0}=2\int_{x_0}^{x_n}u''(x)q''(x)dx=2\,[u''(x)q'(x)]|_{x_0}^{x_n}-2\int_{x_0}^{x_n}u'''(x)q'(x)dx.$$

The term that came out of the integration by parts vanishes since $u(x)$ was a natural spline. Since it is a cubic spline, $u'''(x)$ is constant within each sub-interval. Factoring out these different constants gives the result

$$\left.\frac{df}{d\varepsilon}\right|_{\varepsilon=0}=-2c_0\int_{x_0}^{x_1}q'(x)dx-2c_1\int_{x_1}^{x_2}q'(x)dx-\ldots-2c_n\int_{x_{n-1}}^{x_n}q'(x)dx.$$

Each of the integrals is of the form $\int_{x_k}^{x_{k+1}}q'(x)dx=q(x_{k+1})-q(x_k)=0-0=0$. Hence, the proof is finished. □

12.6.3. Parametric splines. Non-uniformly spaced interpolation points are also easily accommodated by the development of the previous section. The idea is to use a parametric representation of the function or curve and to use uniformly spaced interpolation points in the parameter.

We are asked to interpolate a function $y(x)$ at $x_j,\quad j=0\ldots,N$. If we write $y_j=y(x_j)$, we form the vectors,

$$\mathbf{a}_j=\begin{bmatrix}x_j\\ y_j\end{bmatrix},\quad j=0\ldots \tag{12.13}$$

and form the vector interpolation polynomial

$$\mathbf{S}(t):=\begin{bmatrix}S_x(t)\\ S_y(t)\end{bmatrix}=\sum_{k=0}^{N}\mathbf{c}_jB_m(t-k) \tag{12.14}$$

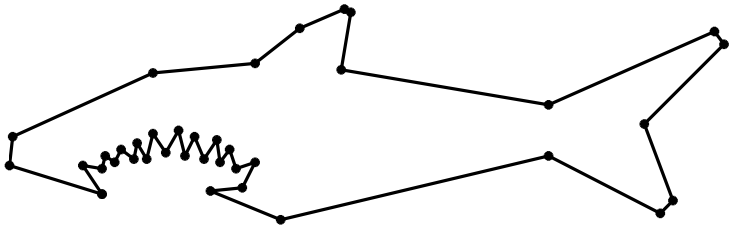


FIGURE 12.6.3. Control points and quadratic splines.

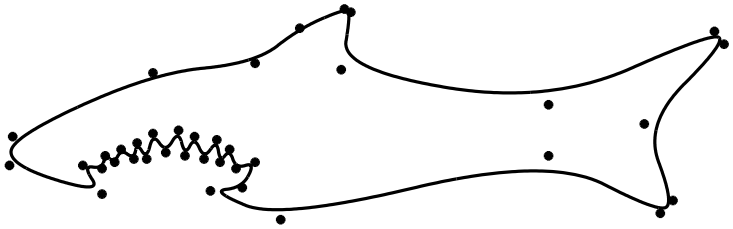


FIGURE 12.6.4. Control points and quadratic splines.

where the $\mathbf{c}_j$ are again obtained from the interpolating conditions, $\mathbf{S}(j) = \mathbf{a}_j, \quad j = 0, \ldots, N$. Note that this is simply a parametric curve drawn in the $x - y$ plane and is completely general.

There is another useful way of constructing a spline approximation. Thinking about it, one realizes that in many situations it does not make much sense to insist on the interpolatory condition. If the original values are for example, contaminated by measurement errors a curve that follows the data-points, without necessarily passing through them, may actually be more advisable. Let us therefore view the data-points (12.13) as so-called *control* points that determine the shape of the approximating spline without the necessity of being interpolatory. Thus, given the control points (12.13) we form the spline of order m,

$$\left[\begin{array}{c} p_x(t) \\ p_y(t) \end{array} \right] =: \mathbf{p}(t) = \sum_{j=0}^{N} \mathbf{a}_j B_m(t - j). \tag{12.15}$$

Let us illustrate this using $m = 3$ (quadratic spline).

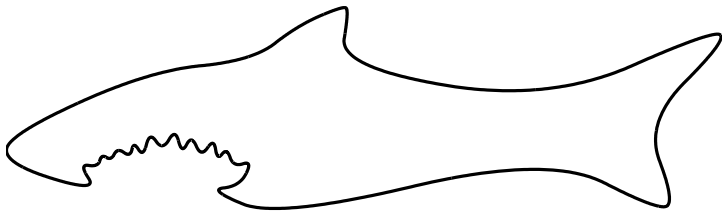


FIGURE 12.6.5. The shark with blunt teeth.

EXAMPLE 41. Figure 12.6.3 shows the control points, connected with straight line (second order splines). If we now form the quadratic spline according to (12.15) with the $\mathbf{a}_j$ as the control points and $m = 3$, we obtain Figure 12.6.4.

Note that the spline follows, but does not pass through the control points. The result is a much smoother curve, with continuous first derivative as shown in Figure 12.6.5. Unfortunately this turned out to be quite disastrous for the shark. Although its body has been nicely streamlined, blunt teeth are no good to a shark at all. One possibility is to insist that the spline passes through (interpolates) the control points defining the teeth. Fortunately there is a very simple mechanism for doing just that: one can force a quadratic spline $m = 3$ to pass through a control point by simply doubling it. For example, if the sequence of control points are given by

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 & 1.5 & 3 & 4 & 4.5 \\ 0 & 3 & 1 & 3.5 & 0.5 \end{bmatrix}$$

we can force the curve through $\begin{bmatrix} 3 \\ 1 \end{bmatrix}$ by simply changing the sequence of control points to

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 & 1.5 & 3 & 3 & 4 & 4.5 \\ 0 & 3 & 1 & 1 & 3.5 & 0.5 \end{bmatrix}.$$

Doing this for all the teeth, a much happier shark is the result, see Figure 12.6.6. Note that the price we pay for forcing the quadratic spline through the control points is a loss of one order of smoothness. At these points we loose the continuity in the first derivative—exactly what is needed to sharpen the teeth. □

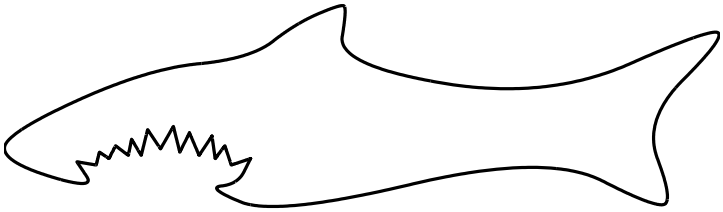


FIGURE 12.6.6. The same shark with sharp teeth.

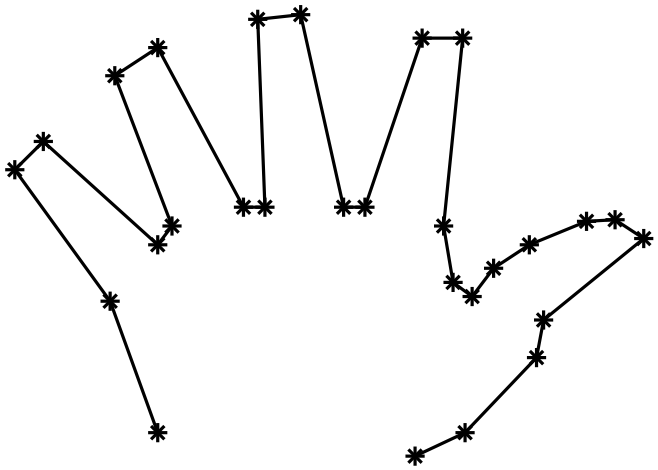


FIGURE 12.6.7. Example of a finite curve

In these examples the curve is closed and periodic boundary conditions are appropriate. If, however, we are given a control sequence as shown in Figure 12.6.7, one has to do something about the endpoints.

In this case it is natural to again force the spline to pass through the endpoints, easily achieved by doubling them as before. The result, again for $m = 3$ is shown in Figure 12.6.8.

All that remains is to explain why the doubling of a control point forces the spline through them. Let us double up on any of the control points of (12.15), say the second one,

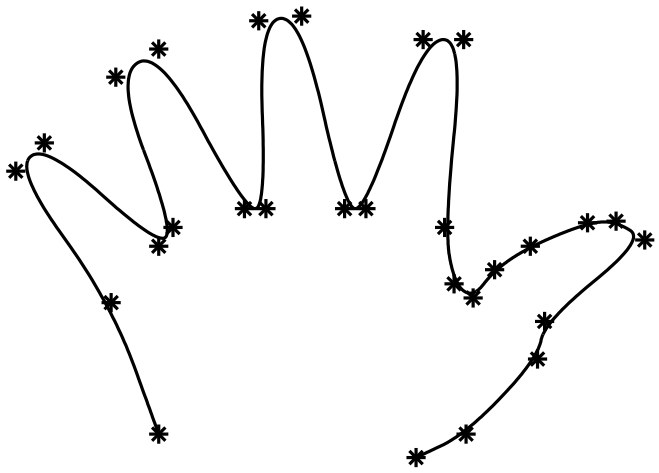


FIGURE 12.6.8. Finite curve and quadratic splines

$$\begin{bmatrix} p_x(t) \\ p_y(t) \end{bmatrix} =: \mathbf{p}(t) = \mathbf{a}_0 B_3(t) + \mathbf{a}_1(B_3(t-1)+B_3(t-2)) + \sum_{j=3}^{N+1} \mathbf{a}_{j-1} B_3(t-j), \quad 0 \le t \le N+1.$$

Since the support of $B_3(t)$ is inside the interval $[-\frac{3}{2}, \frac{3}{2}]$ it follows that

$$\mathbf{p}(\frac{3}{2}) = \mathbf{a}_1(B_3(-\frac{1}{2}) + B_3(\frac{1}{2})) = \mathbf{a}_1,$$

where the last step follows from the partition of unity. In order to see why we loose the smoothness we take the derivative,

$$\frac{d}{dt}\mathbf{p}(\frac{3}{2}) = \mathbf{a}_1 \frac{d}{dt}(B_3(-\frac{1}{2}) + \frac{d}{dt}B_3(\frac{1}{2})) = \mathbf{0},$$

where the last step follows from the symmetry of the B-splines. Thus we find that the tangent of the curve is not defined at the double control points, indicating a discontinuity of the first derivative.

These results depend on the partition of unity—only two shifted quadratic splines are nonzero at $t = \frac{3}{2}$— and for higher order splines more shifted splines are nonzero at any given point. This means that one has to use multiple control points in order to force the spline through it—three points for order 4 splines, etc. Interpolation is enforced at the cost of the smoothness of the curve.

It is remarkable that these splines can be easily computed without any reference to the B-splines. This is the topic of the next section.

12.7. Subdivision schemes for curve fitting.

The technical quality of the graphics in modern animated movies has shown remarkable improvement over the older generations. Modern mathematical developments, in particular the so-called subdivision schemes have played an important role. Let us say we want to do an animation of a person called Geri. At first Geri consists of only a number of control points that we possibly obtained by picking up the coordinates of reflective markers on a body suit worn by a real person. From these $3D$ coordinates we want to build a surface that is starting to resemble Geri. Among other things the surface should be dense so that we can later on map some texture onto it. We also need the representation to be local. We'll probably want to give Geri its own identity and that means that we have to remodel the original surface. This has to be done locally—if we change the shape of Geri's nose for example, the eyes should not be affected. Since we most likely will have to experiment a lot, the scheme has to be fast. Subdivision schemes do exactly this. Not only do companies specializing in animation such as Pixar, use these schemes extensively, they are also widely used for modeling objects in engineering and construction. In this section we explain the basic ideas of the subdivision of curves. We show that it is an efficient way of constructing some of the spline approximants of the previous section. The same ideas are the ones used for modeling surfaces in $3D$, there is just a little more technical detail to think about.

Suppose we are given the following bi-infinite sequence of points, $\mathbf{c}^{(0)} = \{c_j\}$, $j \in \mathbb{Z}$ where each point $c_j \in \mathbb{R}^2$. We are interested in approximating these points with a smooth curve in $\mathbb{R}^2$. (Actually there is nothing that prevents us from considering curves in $\mathbb{R}^n$; curves in $\mathbb{R}^2$ are just simpler to display.) Since we are not necessarily interested in curves passing through these points, we refer to them as *control points* rather than interpolation points, exactly as we did in the previous section

The schemes described in the previous sections consisted of a two-step procedure. First the approximating polynomial was constructed and then evaluated at the desired values. The basic idea behind subdivision schemes is to do away with the first

(a) (b)

FIGURE 12.7.1. (a) The original control points $\star$. The new control points $\circ$ after one iteration.

step—we describe procedures for evaluating the approximation *function* without having to construct it first. Note the subtle change in wording: in the previous section we already moved from an *interpolation* polynomial to an *approximating* polynomial. In this section we go one step further, we'll need to consider approximating *functions.* The main ideas are illustrated by an example.

EXAMPLE 42. Suppose we are given the control points $\mathbf{c}^{(0)}$ indicated by $\star$'s in Figure 12.7.1(a), conveniently connected by straight lines. The idea is to fill in the space between the control points without having to explicitly construct an approximating function. We need a prescription to proceed from the given $\mathbf{c}^{(0)}$ to a new set $\mathbf{c}^{(1)}$ where $\mathbf{c}^{(1)}$ start to fill in the space between the original control points. A very simple rule is the following linear combination of the original control points,

$$c_{2j}^{(1)} = c_j^{(0)} \text{ and } c_{2j+1}^{(1)} = \frac{1}{2}(c_j^{(0)} + c_{j+1}^{(0)}), \quad j \in \mathbb{Z}. \tag{12.1}$$

Note that the odd-indexed new points are generated halfway between the old ones while the even-indexed new points are simply the original points, indicated by $\circ$'s in Figure 12.7.1(b). This step can of course be repeated indefinitely, roughly doubling the number of points at each step. In this case the points fill in or converge to the straight lines connecting the initial control points. This simple rule provides us with the familiar piecewise linear interpolation polynomial. The question is whether we can construct higher order approximating functions. $\square$

Both the existence and smoothness of this limit curve depend on the choice of the coefficients of the linear combination. In general, given a sequence of coefficients

(only a finite number being nonzero) $\mathbf{a} = \{a_j\}$—the *mask* of the subdivision scheme—and an initial sequence of control points $\mathbf{c}$, we define the corresponding subdivision operator S by

$$(S\mathbf{c})_j = \sum_{k\in\mathbf{Z}} a_{j-2k}c_k, \quad j \in \mathbb{Z}. \tag{12.2}$$

The resulting subdivision scheme is then defined, for a given initial sequence of control points c, by

$$\mathbf{c}^{(0)} = \mathbf{c}, \quad \mathbf{c}^{(r)} = S\mathbf{c}^{(r-1)}, \quad r = 1, \ldots, \tag{12.3}$$

or, equivalently,

$$\mathbf{c}^{(0)} = \mathbf{c}, \quad \mathbf{c}^{(r)} = S^r\mathbf{c}, \quad r = 1, \ldots. \tag{12.4}$$

Note that the even– and odd-indexed points decouple. When j is even we use the even-indexed elements of the mask $\mathbf{a}$, and when j is odd we use the odd-indexed elements of $\mathbf{a}$. This means that we have two different maps combined into one.

EXAMPLE 43. Let us illustrate the use of the mask by choosing $a_0 = 1, a_1 = \frac{1}{2} = a_{-1}$, and the rest all zeros. It is then straightforward to see that
$c_{2j}^{(1)} = c_j^{(0)}$ and $c_{2j+1}^{(1)} = \frac{1}{2}\left(c_j^{(0)} + c_{j+1}^{(0)}\right)$,
i.e. this is just the scheme discussed above. □

There is no unique or best way of obtaining a mask. Different choices lead to different approximating functions, provided the resulting scheme converges. We investigate two different types of schemes. The first one, the so-called Dubuc-Deslauriers subdivision **[?, ?]**, is an interpolatory scheme where the approximating function interpolates the initial, and by implication, all subsequent, control points. The second scheme, the so-called Lane-Riesenfeld schemes, do not interpolate the control points—they smooth the data.

The key therefore is to find an appropriate mask. The choice of the mask determines (i) whether the subdivision scheme is interpolatory or smoothing (corner-cutting), (ii) the convergence of the scheme, and (iii) the smoothness of the final curve.

These issues are closely related to the existence of a *refinable function* which interestingly, also provides a link with wavelets [**?**].

Below we discuss both interpolatory as well as corner-cutting subdivision schemes, for infinite as well as finite sequences of control points.

12.7.1. Interpolatory Subdivision . In this section we introduce Dubuc-Deslauriers subdivision as an iterative procedure that reproduces polynomials of a given odd degree, i.e. if the original control points fall on any $p(x) \in \mathbf{P}^{2N+1}$, then all subsequent points also fall on $p(x)$. We also indicate how the limit curve for the Dubuc-Deslauriers scheme depends on the existence of an associated refinable function.

12.7.1.1. *Construction of the mask.* Suppose the original control points fall on a polynomial $p \in \mathbf{P}^{2N+1}$, i.e. $c_j^{(0)} = p(j),\ p \in \mathbf{P}^{2N+1}, \quad j \in \mathbb{Z}$. The idea is to find the shortest possible mask $\mathbf{a}$ such that all the subsequent iterations $c^{(r)}$ also fall on $p(x)$. This implies that

$$\sum_{k\in\mathbf{Z}} a_{j-2k}p(k) = p\left(\frac{j}{2}\right), \forall p(x) \in \mathbf{P}^{2N+1}, \quad j \in \mathbb{Z}. \tag{12.5}$$

Since the interpolation polynomial is unique, it follows that

$$\sum_{k=-N}^{N+1} p(k)L_k(x) = p(x), \quad x \in \mathbb{R}, \tag{12.6}$$

for any polynomial of degree not greater than $2N+1$, where the Lagrange polynomials $L_k(x)$ of degree $2N+1$, are uniquely defined by (see Section **??**),

$$L_k(j) = \delta_{k,j}, \quad k, j = -N, \ldots, N+1. \tag{12.7}$$

Suppose that j is *even* in (12.5), i.e. $j = 2n$ implies that

$$\sum_{k\in\mathbf{Z}} a_{2(n-k)}p(k) = p(n), \forall\ p(x) \in \mathbf{P}^{2N+1}, \quad j \in \mathbf{Z}$$

leading to the choice

$$a_{2j} = \delta_{j,0},\ j \in \mathbb{Z}.$$

The odd-indexed coefficients are obtained by choosing $x = \frac{1}{2}$ in (12.6) and $j = 1$ in (12.5), so that

$$\sum_{k=-N}^{N+1} p(k) L_k(\frac{1}{2}) = p(\frac{1}{2}),$$

and

$$\sum_{k \in \mathbf{Z}} a_{1-2k} p(k) = p\left(\frac{1}{2}\right),$$

respectively. Comparing these two expressions give

$$a_{1-2k} = L_k(\frac{1}{2}) \text{ or } a_{2j+1} = L_{-j}(\frac{1}{2}). \tag{12.8}$$

Thus we find that

$$c_{2j}^{(r+1)} = c_j^{(r)}, \quad j \in \mathbf{Z}, \;\; r = 0, 1, ... \tag{12.9}$$

and that the mask is symmetric,

$$a_j = a_{-j}, \quad j \in \mathbb{Z}. \tag{12.10}$$

Note that the subdivision scheme can now be written as

$$\begin{aligned} c_{2j}^{(r)} &= c_j^{(r-1)} \\ c_{2j+1}^{(r)} &= \sum_{k=j-N}^{j+N+1} a_{2j+1-2k} c_k^{(r-1)}. \end{aligned}$$

EXAMPLE 44. For $N = 0$, we have $L_0(x) = \frac{x-1}{0-1}$ and $L_1(x) = \frac{x-0}{1-0}$. The mask is therefore given by $\mathbf{a} = \begin{bmatrix} \frac{1}{2} & 1 & \frac{1}{2} \end{bmatrix}$, and it simply connects the control points with straight lines, as seen in Figure 12.7.1. □

(a) (b)

FIGURE 12.7.2. (a) Original control points $\star$ and one step of subdivision. (b) Original control points with limit curve.

EXAMPLE 45. **Example:** For $N = 1$, (12.8) give

$$a_{2j+1} = \begin{cases} -\frac{1}{16}, & j = -2, \\ \frac{9}{16}, & j = -1, \\ \frac{9}{16}, & j = 0, \\ -\frac{1}{16}, & j = 1, \\ 0, & \text{otherwise.} \end{cases}$$

Our previous example showed that for $N = 0$, the scheme converges to a continuous, piecewise linear function, connecting the original control points. In contrast, subdivision with the mask for $N = 1$ in converges to a smoother function, while still interpolating the original control points, as shown in Figure 12.7.2. Incidentally, this example is not completely arbitrary. In fact Knuth based his construction of TEX fonts on ideas remarkably similar to subdivision schemes [**?**, Chapter 2], more than 10 years before the Dubuc-Deslauriers scheme was introduced in [**?**].

□

Figure 12.7.2 indicates that the Dubuc-Deslauriers subdivision converges to a smooth function. This limiting curve is described in terms of a *refinable* function which we now proceed to discuss.

12.7.1.2. *Refinable functions.* In general a refinable function is a function that can be written as a linear combination of squashed and shifted versions of itself. The refinable functions $\phi(x)$ in which we are interested are the ones where the linear combination is determined by the mask,

$$\phi(x) = \sum_{j \in \mathbf{Z}} a_j \phi(2x - j), \quad x \in \mathbb{R}. \tag{12.11}$$

Let us fix the ideas with two examples.

EXAMPLE 46. If

$$\phi(x) = \begin{cases} 1 & x \in [0, 1) \\ 0 & \text{elsewhere} \end{cases},$$

then $\phi(x) = \frac{1}{2}\phi(2x) + \phi(2x - \frac{1}{2}) + \frac{1}{2}\phi(2x - 1)$. This function is also known as the Haar wavelet. □

EXAMPLE 47. If

$$\phi(x) = \begin{cases} 2x & x \in \left[0, \frac{1}{2}\right] \\ 2(1-x) & x \in \left[\frac{1}{2}, 1\right] \\ 0 & \text{elsewhere} \end{cases},$$

then $\phi(x) = \frac{1}{2}\phi(2x) + \frac{1}{2}\phi(2x - 1)$.□

These two examples provide a powerful hint—they are shifted versions of the first two B-splines of Section 12.6. This is no accident. All the B-splines of Section 12.6 has this property. It is also the only explicitly known examples of refinable functions. In general, given a mask $\mathbf{a}$, one has to prove that the refinable function $\phi(x)$ defined by (12.11) exists. This is often done by proving a contraction mapping based on (12.11). Although this can provide a way of actually constructing the refinable function, there is a better way, as well explain shortly.

We now pursue the hint provided by our two examples and show that all the B-splines of Section 12.6, or rather, shifted versions of them, are refinable.

First note that $B_1(x)$ satisfies,

$$B_1(x) = B_1(2x + \frac{1}{2}) + B_1(2x - \frac{1}{2}).$$

Taking the Fourier transform,

$$\widehat{B}_1(\omega) = \int_{-\frac{1}{2}}^{\frac{1}{2}} \exp(-i\omega x)dx \tag{12.12}$$

$$= \frac{i}{\omega}\left(\exp(-\frac{1}{2}i\omega) - \exp(+\frac{1}{2}i\omega)\right) \tag{12.13}$$

$$= \frac{1}{2}\left(\exp(-\frac{1}{4}i\omega) + \exp(+\frac{1}{4}i\omega)\right)\frac{i}{\omega/2}\left(\exp(-\frac{1}{4}i\omega) - \exp(+\frac{1}{4}i\omega)\right) \tag{12.14}$$

$$= \frac{1}{2}\left(\exp(-\frac{1}{4}i\omega) + \exp(+\frac{1}{4}i\omega)\right)\widehat{B}_1(\omega/2). \tag{12.15}$$

Higher order splines are defined in terms of convolutions (12.6), written as,

$$B_m(x) = B_1 \star \cdots \star B_1(x) \qquad (m \text{ times}),$$

or making use of the convolution theorem,

$$\widehat{B}_m(\omega) = \left(\widehat{B}_1(\omega)\right)^m.$$

Combining this result with (12.15), we find that

$$\widehat{B}_m(\omega) = \left(\frac{\exp(-\frac{1}{4}i\omega) + \exp(+\frac{1}{4}i\omega)}{2}\right)^m \widehat{B}_m(\omega/2).$$

The inverse Fourier transform returns us to $B_m(x)$,

$$
\begin{aligned}
B_m(x) &= \frac{1}{2\pi}\int_{-\infty}^{\infty} \exp(ix\omega)\widehat{B}_m(\omega)d\omega && (12.16)\\
&= \frac{1}{2\pi}\int_{-\infty}^{\infty} \exp(ix\omega)\left(\frac{\exp(-\frac{1}{4}i\omega)+\exp(+\frac{1}{4}i\omega)}{2}\right)^m \widehat{B}_m(\omega/2)d\omega && (12.17)\\
&= \frac{1}{2^{m-1}}\frac{1}{2\pi}\int_{-\infty}^{\infty} \exp(i2x\omega)\left(\exp(-\frac{1}{2}i\omega)+\exp(+\frac{1}{2}i\omega)\right)^m \widehat{B}_m(\omega)d\omega && (12.18)\\
&= \frac{1}{2^{m-1}}\sum_{k=0}^{m}\begin{pmatrix} m \\ k \end{pmatrix}\frac{1}{2\pi}\int_{-\infty}^{\infty} \exp\left(i(2x-k+\frac{1}{2}m)\omega\right)\widehat{B}_m(\omega)d\omega && (12.19)\\
&= \frac{1}{2^{m-1}}\sum_{k=0}^{m}\begin{pmatrix} m \\ k \end{pmatrix} B_m(2x-k+\frac{1}{2}m) && (12.20)
\end{aligned}
$$

This is close, but not quite what we want—we still have to shift the B-splines through $\frac{1}{2}m$. Therefore, if we define $B_m(t) =: \widetilde{B}_m(t+\frac{1}{2}m)$, it follows that,

$$\widetilde{B}_m(x) = \frac{1}{2^{m-1}}\sum_{k=0}^{m}\begin{pmatrix} m \\ k \end{pmatrix}\widetilde{B}_m(2x-k). \tag{12.21}$$

This result will be significant when we discuss the convergence of the Lane-Riesenfeld subdivision schemes.

12.7.1.3. *Convergence of Dubuc-Deslauriers subdivision.* For a given mask $\mathbf{a}$, the refinable function is defined by (12.11). Micchelli proved, using concepts outside the scope of this book, the existence of a refinable function for the Dubuc-Deslauriers

mask. Moreover, it has the following properties,

$$\text{(12.22)} \qquad \textit{(i)} \qquad \phi(j) = \delta_{j,0}, \quad j \in \mathbf{Z}$$

(12.23)

$$\text{(12.24)} \qquad \textit{(ii)} \qquad \phi(x) = 0,\ x \notin (-2N-1, 2N+1)$$

(12.25)

$$\text{(12.26)} \qquad \textit{(iii)} \qquad \sum_{j\in\mathbf{Z}} p(j)\phi(x-j) = p(x),\ p(x) \in \mathbf{P}^{2N+1},\ x \in \mathbf{R}$$

(12.27)

$$\text{(12.28)} \qquad \textit{(iv)} \qquad \phi(x) = \phi(-x), \quad x \in \mathbf{R}$$

(12.29)

$$\text{(12.30)} \qquad \textit{(v)} \qquad \phi\left(\frac{j}{2}\right) = a_j, \quad j \in \mathbf{Z}.$$

Now we are finally in a position to describe the limiting curve for Dubuc-Deslauriers subdivision: For the given initial control points c, the limiting curve $f(x)$ of the Dubuc-Deslauriers subdivision scheme is given by,

$$\text{(12.31)} \qquad f(x) = \sum_{j\in\mathbf{Z}} c_j\phi(x-j), \quad x \in \mathbf{R}.$$

Can you see how this formula allows us to compute the refinable functions for the Dubuc-Deslauriers subdivision? (**Hint:** If we want the limiting curve to be a shifted version of the refinable function $\phi(x)$, how do we choose the initial control points, $\mathbf{c}$?)

12.7.2. A modified subdivision scheme for finite sequences. The algorithms for bi-infinite sequences as described in the previous sections, are applied mainly in the case of periodic sequences. For finitely supported sequences these algorithms must be modified to accommodate the boundaries; away from the boundaries, we use the schemes as derived in Sub-section 12.7.1.1, of course. To be more precise, let us say we have a finite initial sequence given by $\mathbf{c} = [c_0, \ldots, c_K]$, and that we wish to apply a Dubuc-Deslauriers scheme of order N, i.e. we want to reproduce polynomials $p(x) \in \mathbf{P}^{2N+1}$. It is easy to understand the problem if we write out the

subdivision scheme (12.2) for $j = 1$,

$$c_1^{(1)} = \sum_{k=-N}^{N+1} a_{-2k+1} c_k^{(0)}.$$

Since the formula requires the values $c_{-N}^{(0)}, c_{-N+1}^{(0)}, \ldots, c_{-1}^{(0)}$, it cannot be used. The problem persists for all values near the boundary, up to

$$c_{2N-1}^{(1)} = \sum_{k=-1}^{2N} a_{2(N-k)-1} c_k^{(0)}$$

Concentrating just on the left-hand boundary—the right-hand boundary is treated in exactly the same way—the problem is for $j = 1, 3, \ldots, 2N-1$. For $j = 2N+1$, and higher, things are fine until we run into the right-hand boundary. It is for these N values (at each boundary) that we need an alternative formula.

The idea is very simple. We still want to reproduce polynomials of degree $2N+1$, and therefore simply base our formula on polynomials defined inside the finite interval. Let $\ell_k(x),\ k = 0, \ldots, 2N+1$ be the Lagrange polynomials of degree $2N+1$, uniquely defined by the $2N+2$ interpolation points adjacent to the left-hand boundary,

$$\ell_k(j) = \delta_{k,j}, \quad k, j = 0, \ldots, 2N+1. \tag{12.32}$$

For any polynomial, $p(x) \in \mathbf{P}^{2N+1}$ we can again write

$$\sum_{k=0}^{2N+1} p(k)\ell_k(x) = p(x).$$

For our subdivision scheme to reproduce polynomials $p(x) \in \mathbf{P}^{2N+1}$ we choose $p(k) = c_k$ and evaluate the polynomial at $x = j + \frac{1}{2}$ for $j = 0, \ldots, N-1$,

$$p(j+\frac{1}{2}) = \sum_{k=0}^{2N+1} p(k)\ell_k(j+\frac{1}{2}) \tag{12.33}$$

$$= \sum_{k=0}^{2N+1} c_k \ell_k(j+\frac{1}{2}) \quad j = 0, \ldots, N-1. \tag{12.34}$$

Noting that the $\ell_k(x)$ are just shifted versions of the $L_k(x)$ defined by (12.7),

$$\ell_k(x) = L_{k-N}(x - N), \quad k = 0, \ldots, 2N + 1,$$

the modified scheme for the left hand boundary ($j = 0, \ldots, N - 1$) becomes

$$c_{2j}^{(r+1)} = c_j^{(r)},$$

$$c_{2j+1}^{(r+1)} = \textstyle\sum_{k=0}^{2N+1} a_{j,k} c_k^{(r)},$$

where for $j = 0, 1, \ldots, N - 1$,

$$a_{j,k} = \begin{cases} L_{k-N}\left(j + \frac{1}{2} - N\right), & k = 0, \ldots, 2N + 1, \\ \\ 0, & \text{otherwise.} \end{cases} \tag{12.35}$$

For interior points $j \geq N$ (avoiding the right hand boundary) the scheme is the same as the one we derived above. To wit, if we write

$$c_{2j+1}^{(r+1)} = \sum_{k=j-N}^{j+N+1} a_{j,k} c_k^{(r)}$$

then $a_{j,k} = a_{2(j-k)+1}$ so that

$$a_{j,k} = \begin{cases} L_{k-j}(\frac{1}{2}), & k = -N + j, \ldots, N + 1 + j, \\ \\ 0, & \text{otherwise.} \end{cases} \tag{12.36}$$

Note that the subdivision coefficients for the boundary formulas (12.35) have two indexes, j and k. Thus the mask changes for all the boundary points. This is marked contrast with the interior scheme (12.36), which is identical to the scheme for bi-infinite sequences (12.8).

EXAMPLE 48. For $N = 1$, (12.35) gives

$$a_{0,k} = \begin{cases} \frac{5}{16}, & k = 0, \\ \frac{15}{16}, & k = 1, \\ -\frac{5}{16}, & k = 2, \\ \frac{1}{16}, & k = 3, \\ 0, & k = 4, \ldots . \end{cases}$$

The basic procedure is exactly the same at the left and right hand boundaries.

Given the modified mask for finite sequences, it can be shown that a set of refinable functions exist with respect to the modified mask (defined similarly to (12.11)). The boundary modifications of the refinable function is illustrated in Figure 12.7.3. □

The existence of a refinable function for the boundary modified subdivision scheme allows one to construct wavelets for finite intervals. It is remarkable that these wavelets have finite decomposition and reconstruction sequences [**?**].

Next we discuss the so-called corner-cutting subdivision schemes.

12.7.3. Corner-cutting Subdivision. The problem with interpolatory subdivision schemes is that the control points themselves may contain noise in which case a certain amount of smoothing might be appropriate. We are for a subdivision scheme where the limit curve remains close to the control points but does not necessarily go through them. In this section we discuss one class of corner-cutting subdivision schemes, namely the de Rham-Chaikin scheme and its generalization, the Lane-Riesenfeld scheme. The essential difference between the corner-cutting and interpolatory schemes lies in the choice of the mask of equation (12.2).

EXAMPLE 49. The de Rham-Chaikin mask is given by,

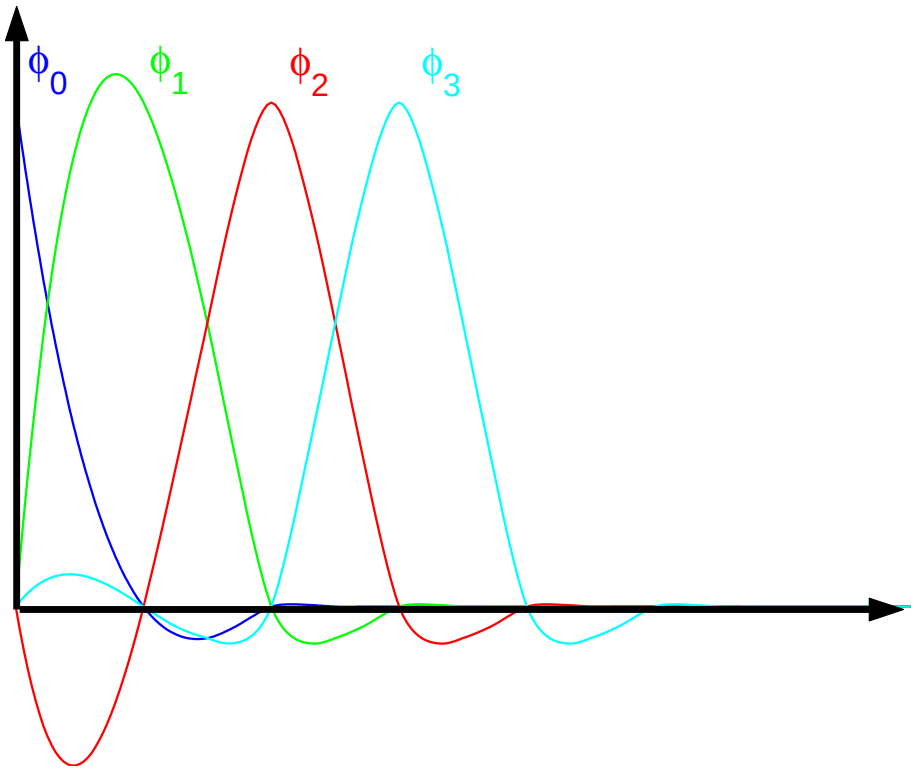


FIGURE 12.7.3. The refinable functions at the boundaries.

$$a_j = \frac{1}{4}\begin{pmatrix} 3 \\ j \end{pmatrix}, \quad j = 0, \ldots, 3 \tag{12.37}$$

$$= \frac{1}{4}\begin{bmatrix} 1 & 3 & 3 & 1 \end{bmatrix}. \tag{12.38}$$

Since this mask is not interpolatory, two new control points are created between any two given control points. According to prescription (12.38), the two points are inserted on the straight line connecting the two points, $\frac{1}{4}$ of the distance away from the points, as illustrated in Figure 12.7.4(a). The original control points are indicated by $\star$ and the new ones by $\circ$. This procedure is repeated iteratively, at each step two new points are inserted between two existing ones. The result is shown in Figure 12.7.4(b). Note the smoothness of the limit curve. $\square$

(a) (b)

FIGURE 12.7.4. De Rham-Chaikin corner cutting. (a) The original $\star$ and new $\circ$ control points ofter one iteration. (b) The limit curve.

Lane-Riesenfeld subdivision is a straightforward generalization of the de Rham-Chaikin scheme, with mask given by

$$a_j = \frac{1}{2^{m-1}} \begin{pmatrix} m \\ j \end{pmatrix}, \quad j = 0, \ldots, m. \tag{12.39}$$

The Rham-Chaikin scheme is therefore a Lane-Riesenfeld scheme of order $m = 3$.

The existence of a refinable function for Lane-Riesenfeld is straightforward; we only need to recall from Section 12.6 that the B-splines satisfy a refinable equation with coefficients given by (12.39). The limiting curve is therefore the spline,

$$p(x) = \sum_j c_j^{(0)} \widetilde{B}_m(x - j).$$

Thus the limiting curve becomes smoother with increasing values of m. Also note that it is precisely the parametric spline constructed in Section 12.6.

All that remains to be done is to modify the scheme in the presence of boundaries. The problem is the same as before—we need to supply missing values at the boundary. A very simple procedure is to repeat the boundary values as many times as needed. This should remind the reader of the procedure used in a previous section of forcing the parametric spline to pass through the control points. In fact, it is exactly the same procedure. It turns out that this again leads to a refinable function. Thus the boundary modified scheme again converges to a spline of the same degree as defined by the interior mask. The de Rham-Chaikin boundary modified refinable

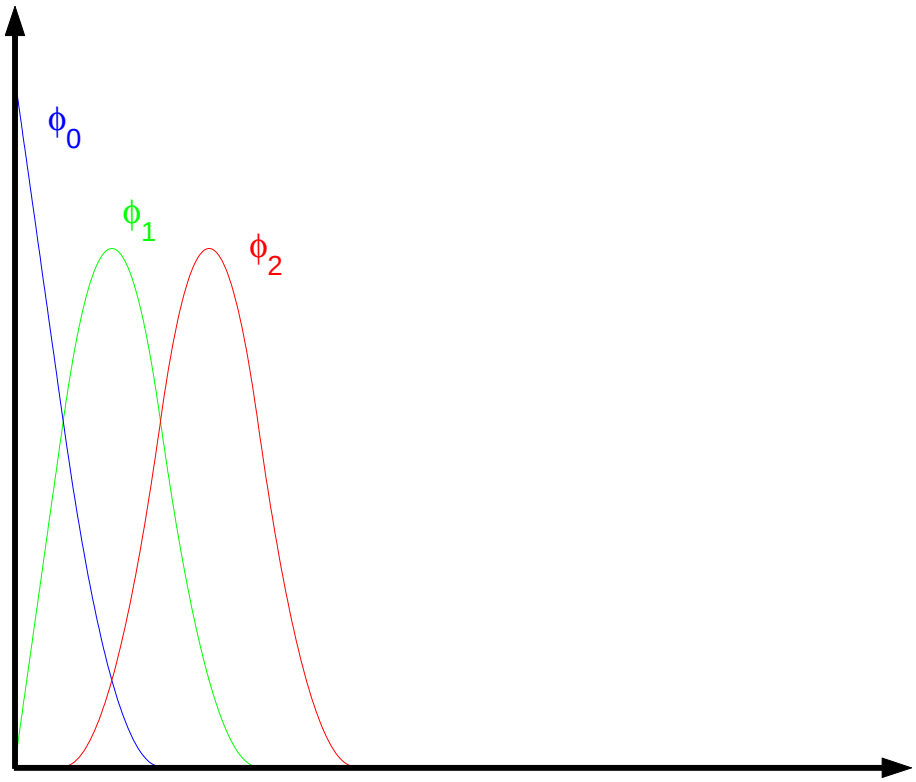
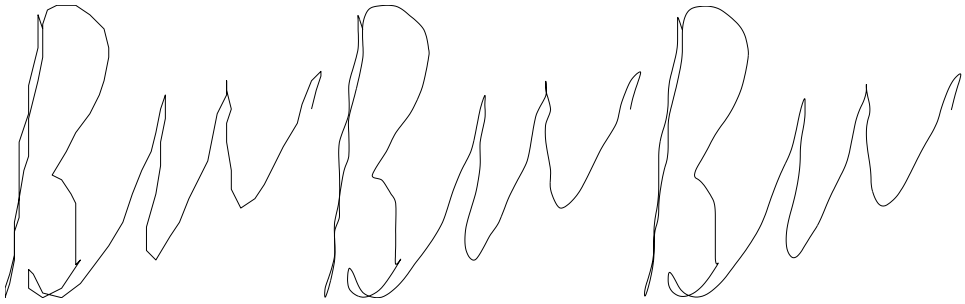


FIGURE 12.7.5. Boundary modifications of the de Rham-Chaikin refinable function.

functions are shown in Figure 12.7.5. In this case we need a doubling of the boundary points, as explained in detail in Section 12.6.3spline. In fact, all the examples in section 12.6.3spline were calculated using the de Rham-Chaikin subdivision scheme.

The pointed out above, all the examples shown in Sub-section 12.6.3spline} were computed using the de Rham-Chaikin subdivision scheme. In this section we compare the interpolatory and corner-cutting subdivision schemes using two problems that also occur in practical applications and both examples are defined on a finite interval, necessitating the boundary modifications.

EXAMPLE 50. The first example is a signature which was obtained from a digitizing tablet as part of a signature verification system. This particular tablet is not of the highest quality (but cheap!) and the coarseness of the underlying grid is clearly visible as shown in Figure 12.7.6(a). In its present form the signature is not suitable



(a) (b) (c)

FIGURE 12.7.6. Smoothing a signature. (a) Original. (b) Interpolatory subdivision. (c) Corner cutting subdivision.

for verification purposes and needs to be smoothed. Figure 12.7.6(b) and (c) show the limit curves of our two types of subdivision schemes—interpolatory and corner cutting.

Note that both schemes smooth the signature. Interpolatory subdivision keeps the original points, although they are themselves inaccurate. The smoothing is therefore limited to newly generated points. Since the corner cutting algorithms also smooth the original points, for this application they appear to be more appropriate.□

EXAMPLE 51. In our second example we illustrate the same properties on an image. For these applications the subdivision is first applied the each row (or column) of the image and then on each column (or row) of the result. Figure 12.7.7(a) shows some detail of a larger image and the blockiness of the individual pixels is clearly visible. Again the two different types subdivision smooth the image and both images shown in Figures 12.7.7(b) and (c) are significant improvements of the original. We leave it to the reader to decide which is the more pleasing result. □

(a) (b) (c)

FIGURE 12.7.7. Smoothing an image. (a) Original. (b) Interpolatory subdivision. (c) Corner cutting subdivision.

CHAPTER 13

ZEROS OF FUNCTIONS

13.1. Introduction.

Many different numerical methods have been proposed for solving scalar equations of the form

$$f(x) = 0.$$

In applications there is usually greater interest (and certainly a greater challenge) in solving nonlinear *sytems* of the form

$$\mathbf{f}(\mathbf{x}) = \mathbf{0}$$

where $\mathbf{f} : \mathbb{R}^N \longrightarrow \mathbb{R}^N$ is a vector valued function of a vector variable. Written in component form it becomes,

$$\begin{aligned} f_1(x_1, \ldots, x_N) &= 0 \\ &\vdots \\ f_N(x_1, \ldots, x_N) &= 0. \end{aligned}$$

Closely related to the problem of solving nonlinear systems is that of finding the optimum (maximum or minimum) of a real valued function of several variables. This is discussed in Chapter ?

The discussion of this chapter starts with the problem of finding the zeros of a scalar equation. Four methods will be discussed in detail, the bisection method, fixed-point interation, Newton's method, and the secant method. Subsequently we turn to the problem of solving systems of nonlinear equations. Two methods will be discussed, the always-important Newton's method, and Broyden's method.

13.2. Four Iterative Methods for the Scalar Case.

We illustrate the four methods on the following example,

$$f(x) = 2 - x - e^{-x}, \tag{13.1}$$

see Figure ? Note the two distinct roots, $x = -1.148193$, and $x = 1.841406$.

13.2.1. Bisection. If we have two guesses $x_{\rm l}$and x_h of a root such that $x_{\rm l} < x_{\rm h}$ such that $f(x_{\rm l})f(x_{\rm h}) < 0$, we know (assuming that $f(x)$ is continuous) that there is a root $x^\star$ somewhere in between, i.e. $x^\star \in (x_{\rm l}, x_{\rm h})$. The simple idea of the bisection method is to check the function value in the middle, $x_{\rm n} = \frac{1}{2}(x_{\rm l}+x_{\rm h})$. If $f(x_{\rm n})f(x_{\rm l}) < 0$, then $x_{\rm h}$ is replaced with $x_{\rm n}$, and the procedure repeated. If, on the other hand, $f(x_{\rm n})f(x_{\rm h}) < 0$, then $x_{\rm l}$ is replaced with $x_{\rm n}$, and the procedure repeated.

Since we repeated half the interval in which the root is located, the error is halved at each iteration. More precisely, if $e_{n+1} = x_{\rm h} - x_{\rm l}$after n iterations, then

$$e_{n+1} = \frac{1}{2}e_n.$$

although convergence is guaranteed, it tends to be slow. If we write the iterates x_n in binary format, we gain one binary digit of accuracy per iteration. It is possible to do much better. In practice however, it is standard practice to first apply a robust algorithm like bisection until we get close enough to the root, in which case the algorithms switches to one of the faster methods to be discussed below.

Table 1 shows the convergence of the bisection method on our example (13.1).

13.2.2. Fixed Point iteration. For this approach we rewrite the equation as

$$x = F(x), \tag{13.2}$$

choose an initial guess x_0 and iterate

$$x_{n+1} = F(x_n), \;\; n = 0, \ldots. \tag{13.3}$$

In order to apply Fixed Point (FP) iteration to our example (13.1), we rewrite it as

$$x = 2 - e^{-x}.$$

Iteration number	x_{l}	x_{h}
0	***1.8***000000000	2.0000000000
1	***1.8***000000000	***1.8***500000000
2	***1.8***250000000	***1.8***500000000
3	***1.8***375000000	***1.8***500000000
4	***1.8***375000000	***1.84***37500000
⋮	⋮	⋮
10	***1.841***2109375	***1.84140***62500
⋮	⋮	⋮
20	***1.841405***4871	***1.8414056***778

TABLE 1. Convergence of the bisection method. The bold, italicized digits are correct.

Iteration Number	$x_{n+1} = 2 - e^{-x_n}$	$x_{n+1} = -\ln(2 - x_n)$
0	2.000000000	***1.8414***000000
1	***1.8***646647168	***1.841***3699698
2	***1.84***50518473	***1.841***1806421
3	***1.841***9828721	***1.8***399878362
⋮	⋮	⋮
10	***1.8414056***19	***-1.***0241369097
⋮	⋮	⋮
20	***1.8414056604***	***-1.14619***18984

TABLE 2. Fixed Point examples (13.4) and (13.5). The bold, italicized digits are correct.

Another possibility is

$$x = -\ln(2 - x).$$

Written in this form the iteration becomes

$$x_{n+1} = 2 - e^{-x_n}, \tag{13.4}$$

and

$$x_{n+1} = -\ln(2 - x_n). \tag{13.5}$$

The results for the two cases are shown in Table 2.

The table shows an interesting fact. The first iteration (13.4) converges steadily to the solution. The second iteration however, although starting close to the solution, strays away and eventually settles on the *second* root in the vicinity of -1.14619. This is something to be investigated in more detail.

Let us denote the root by α, i.e.

$$\alpha = F(\alpha).$$

If the error in the n-th iteration is given by $e_n := x_n - \alpha$ it follows that

$$\begin{aligned} e_{n+1} &= x_{n+1} - \alpha \\ &= F(x_n) - F(\alpha) \\ &= \left[F(\alpha) + F'(\alpha)(x_n - \alpha) + O\left(|x_n - \alpha|^2\right)\right] - F(\alpha) \\ &= F'(\alpha)e_n + O\left(|e_n|^2\right) \end{aligned}$$

To leading order we therefore have that the error at each iteration is reduced/amplified by a factor of $F'(\alpha)$. Clearly if we want the error to reduce during each iteration we require

$$|F'(\alpha)| < 1. \tag{13.6}$$

Let us return to our two examples. In the case of iteration (13.4),

$$F'(\alpha) = e^{-\alpha},$$

therefore the scheme converges for the positive root, and should diverge for the negative root (try it!).

In the case of iteration (13.5),

$$F'(\alpha) = \frac{1}{2 - \alpha}.$$

In this case it will clearly converges for the negative root, and diverges for the positive root. That is exactly the behavior observed in Table 2.

13.2.3. Newton's method. One way of writing $f(x) = 0$ in Fixed Point form, is to write it as

$$x = x - \beta f(x) =: \ F(x)$$

Iteration number	x_n
0	2.0
1	***1.84***3482357250334348
2	***1.8414***06066157926438
3	***1.841405660436976***121

TABLE 3. The convergence of Newton's method. The bold, italicized digits are correct.

where β can be any constant. We have just seen that the convergence of fixed point iterations depend on

$$F'(\alpha) = 1 - \beta f'(\alpha).$$

Since β is an arbitrary parameter, we may be able to use it to get the best possible rate of convergence. That happens if we choose

$$F'(\alpha) = 0 = 1 - \beta f'(\alpha),$$

or

$$\beta = \frac{1}{f'(\alpha)}.$$

There is just one problem, we don't know the value of α—it is the root that we are trying to estimate. So instead of using the unknown α, we use our best current estimate of it, and choose

$$\beta = \frac{1}{f'(x_n)}.$$

Newton's method therefore becomes

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}. \tag{13.7}$$

Let us see how it behaves in practice. Table 3 show the results.

After just three iterations, the result is correct to 14 digits. More significantly, the number of correct digits seem to about double at each iteration. Let us confirm this.

Since Newton's method is in the form of FP iteration (13.3) with

$$F(x) = x - \frac{f(x)}{f'(x)},$$

Iteration Number	x_n
0	***1***.0000000000000000000
1	2.0000000000000000000
2	***1.8***2365723756505020 03
3	***1.841***55501830482185 65
4	***1.84140***60821125598868
5	***1.8414056604***270182076
6	***1.84140566043696063 7***

TABLE 4. Convergence of the secant method. The bold, italicized digits are correct.

we can analyze its convergence as before. In this case however, we need to keep one more term in the Taylor expansion. Therefore

$$\begin{aligned} e_{n+1} &= x_{n+1} - \alpha \\ &= F(x_n) - F(\alpha) \\ &= F'(\alpha)e_n + \frac{1}{2}F''(\alpha)e_n^2 + O(e_n^3). \end{aligned}$$

We have derived Newton's method so that $F'(\alpha) = 0$. To leading order we therefore have *second* order convergence. In practice this means that the error is roughly squared at each iteration, i.e. the number of accurate digits is roughly doubled at each iteration, as observed in Table 3.

13.2.4. Secant method. The one drawback of Newton's method is the fact that the derivative is required. Not only can it be complicated, but it increases the computational cost. This is significant, especially if one wants to solve systems of nonlinear equations, as we'll see below. The secant method replaces the derivative $f'(x_n)$ with $\frac{f(x_n)-f(x_{n-1})}{x_n-x_{n-1}}$ so that the secant method is given by

$$x_{n+1} = x_n - f(x_n)\frac{x_n - x_{n-1}}{f(x_n) - f(x_{n-1})}. \tag{13.8}$$

In order to start the scheme two initial guesses are required. After that the method continues with only one function evaluation per step; roughly of that of Newton. Its convergence properties are illustrated in Table 4

We have apparently sacrificed speed of convergence in favor of one less function evaluation per iteration. But how much less? Before we analyze the secant method in more detail, let us compute the rate of convergence numerically. Accordingly, assume that

$$e_{n+1} = Ke_n^q$$

where K and q are constants for sufficiently many iterations, with q the rate of convergence of course. If we also write

$$e_n = Ke_{n-1}^q$$

we divide the two expressions to obtain

$$\frac{e_{n+1}}{e_n} = \left(\frac{e_n}{e_{n-1}}\right)^q$$

so that

$$q = \ln\left(\frac{e_{n+1}}{e_n}\right) / \ln\left(\frac{e_n}{e_{n-1}}\right)$$

Taking x_6 in Table 4to be the exact solution, we find that $q =$. Although apparently slower that Newton's method, it is still super linear.

Using the secant equation (13.8), it folows that

$$\begin{aligned} e_{n+1} &= x_{n+1} - \alpha \\ &= x_n - \alpha - f(x_n)\frac{x_n - \alpha - (x_{n-1} - \alpha)}{f(x_n) - f(x_{n-1})} \\ &= \frac{e_n\left(f(x_n) - f(x_{n-1})\right) - f(x_n)\left(e_n - e_{n-1}\right)}{f(x_n) - f(x_{n-1})} \\ &= \frac{f(x_n)e_{n-1} - f(x_{n-1})e_n}{f(x_n) - f(x_{n-1})}. \end{aligned}$$

If we now do a Taylor expansion

$$f(x_n) = f(\alpha) + f'(\alpha)e_n + \frac{1}{2}f''(\alpha)e_n^2 + O(e_n^3),$$

with $f(\alpha) = 0$, and similarly for $f(e_{n-1})$, we find

$$\begin{aligned} e_{n+1} &= \frac{\frac{1}{2}f''(\alpha)\left(e_{n-1}e_n^2 - e_n e_{n-1}^2\right) + O(e_n^2 e_{n-1}^2)}{f'(e_n)\left(e_n - e_{n-1}\right) + O(e_n^2 + e_{n-1}^2)} \\ &= \frac{f''(\alpha)}{2f'(\alpha)} e_n e_{n-1} + O(e^4). \end{aligned}$$

The fact that e_{n+1} is now proportional to the *product* of the error at the two preceeding iterations reduces the convergence rate to something a little less than quadratic. In order to see what it is, assume $e_{n+1} = Ke_n^q = K(Ke_{n-1}^q)^q$. Balancing the powers on the left– and right-hand sides give

$$q^2 = q + 1$$

with

$$q = \frac{1}{2}\left(1 + \sqrt{5}\right).$$

13.3. Nonlinear Systems.

13.3.1. Newton's method. We now turn to solving nonlinear *sytems* of the form

$$\mathbf{f}(\mathbf{x}) = \mathbf{0}, \tag{13.1}$$

where $\mathbf{f} : \mathbb{R}^N \longrightarrow \mathbb{R}^N$ is a vector valued function of a vector variable, as promised in the Introduction. Written in component form it becomes,

$$\begin{aligned} f_1(x_1, \ldots, x_N) &= 0 \\ &\vdots \\ f_N(x_1, \ldots, x_N) &= 0. \end{aligned} \tag{13.2}$$

The basic idea is the same as for a scalar equation. Assuming we have an approximate solution $\mathbf{x}_k$, we are looking for a direction $\mathbf{p}$ such that $\mathbf{x}_k + \mathbf{p}$ is closer to the solution. Ons wayof finding $\mathbf{p}$ is to replace $\mathbf{f}(\mathbf{x}_k + \mathbf{p})$ with its linear approximation

$$\mathbf{m}(\mathbf{p}) = \mathbf{f}_k + J_k\mathbf{p} \tag{13.3}$$

where $\mathbf{f}_k$ and J_k are hte function values and Jacobian matrix evaluated at $\mathbf{x}_k$. In component form we have

$$J_k := \left[\begin{array}{ccc} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_N} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_N}{\partial x_1} & \cdots & \frac{\partial f_N}{\partial x_N} \end{array} \right]_{\mathbf{x}=\mathbf{x}_0} .$$

Ideally we would like $\mathbf{f}(\mathbf{x}_k + \mathbf{p}) = \mathbf{0}$, since this is not possible, we set $\mathbf{m}(\mathbf{p}) = \mathbf{0}$ instead, to find

$$J_k \mathbf{p}_k = -\mathbf{f}_k. \tag{13.4}$$

The update is then given by

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{p}_k.$$

Starting with an initial guess, $\mathbf{x}_0$ this procedure is then iterated to convergence.

Using arguments along similar lines as the scalar equation, one can again show that Newton's method for systems is quadratically convergent.

The main difficulty, even more so than in the scalas case is the requirement of the Jacobian matrix.

13.3.2. Broyden's method. In Broyden's method the Jacobian J_k is replaced by an approximation B_k and the linearization (13.3) becomes,

$$\mathbf{m}(\mathbf{p}) = \mathbf{f}_k + B_k \mathbf{p}.$$

If B_k is nonsingular, the iterations become

$$B_k \mathbf{p}_k = -\mathbf{f}_k.$$

We need B_k to mimic the behavior of the true Jacobian. The idea is if we are given B_k how can we update it in such a way that the behavior of the true Jacobian is mimicked. If we let $\mathbf{s}_k := \mathbf{x}_{k+1} - \mathbf{x}_k$ and $\mathbf{y}_k := \mathbf{f}_{k+1} - \mathbf{f}_k$, it follows from Taylor's theorem that

$$\begin{aligned} \mathbf{y}_k &= \int_0^1 J(\mathbf{x}_k + t\mathbf{s}_k)\mathbf{s}_k dt + O\left(\|\mathbf{s}_k\|^2\right) \\ &\approx J_{k+1}\mathbf{s}_k + O\left(\|\mathbf{s}_k\|^2\right). \end{aligned}$$

This is the behavior we require from B_{k+1}, known as the *secant* condition

$$\mathbf{y}_k = B_{k+1}\mathbf{s}_k. \tag{13.5}$$

Note that the secant condition only requires how B_{k+1} should behave in the direction of $\mathbf{s}_k$. In fact an operations count shows considerable freedom in the choice of B_{k+1}. (B_{k+1} has N^2 degrees of freedom, and (13.5) provided N conditions.) If $N = 1$, B_{k+1} is completely determined by (13.5), reducing it to the scalar secant method. For $N > 1$, the most successful choice for updating B_{k+1} is Broyden's method,

$$B_{k+1} = B_k + \frac{(\mathbf{y}_k - B_k\mathbf{s}_k)\mathbf{s}_k^T}{\mathbf{s}_k^T\mathbf{s}}. \tag{13.6}$$

EXERCISE. Show that B_{k+1} as defined by (13.6) satisfies the secant condition (13.5).

Broyden's method has an attractive property: It makes the smallest possible change to the approximate Jacobian, as measured by the Euclidean norm $\|B_{k+1} - B_k\|$, consistent with the secant condition (13.5). This can be seen through a direct calculation. Let B be any matrix satisfying the secant constraint, $B\mathbf{s}_k = \mathbf{y}_k$, and consider

$$\begin{aligned}
\|B_{k+1} - B_k\| &= \left\| \frac{(\mathbf{y}_k - B_k\mathbf{s}_k)\mathbf{s}_k^T}{\mathbf{s}_k^T\mathbf{s}} \right\| \\
&= \left\| \frac{(B - B_k)\mathbf{s}_k\mathbf{s}_k^T}{\mathbf{s}_k^T\mathbf{s}} \right\| \\
&\leq \|B - B_k\| \left\| \frac{\mathbf{s}\mathbf{s}_k^T}{\mathbf{s}_k^T\mathbf{s}} \right\| \\
&= \|B - B_k\|,
\end{aligned}$$

where the final step follows from the fact that the rank one matrix $\mathbf{s}\mathbf{s}^T/\mathbf{s}^T\mathbf{s}$ has singular value $\sigma_N = 1$. We therefore have that

$$\|B_{k+1} - B_k\| \leq \|B - B_k\|,$$

for any matrix B that satisfies the secant constraint.

Not all our problems have been resolved. There is still the matter of choosing the initial B_0. As the performance of Broyden's method depends critically on this choice, careful attention should be given to its choice. Since one should choose it to

mimic the behavior of $J(\mathbf{x}_0)$, one possibility is to choose $B_0 = J_0$, if this feasible. If not, one can try some finite difference approximation of J_0.

CHAPTER 14

Radial Basis Functions

14.1. Introduction.

This chapter focuses on the application of RBF to the numerical solution of PDEs. The RBF methodology was originated by Rolland Hardy around 1970 in connection with a cartography application that required multivariate scattered-node interpolation [?]. In a much noted 1982 survey [?], this approach, using a certain type of basis functions known as multiquadrics (MQ), was found to be the preferable one of about 30 then known methods (scoring the best in 13 of 18 tests, and second best in 3 of the remaining 5 tests). Although unconditional non-singularity of the interpolation problem was known early in some special cases [?], it was the breakthrough discovery in 1986 of guaranteed non-singularity also for MQ [?] which propelled the development of RBF into one of the most promising areas in modern computational mathematics.

In this chapter, we first introduce RBF as a generalization of standard cubic splines to multiple dimensions, and then summarize some results concerning accuracy and non-singularity. We note that one of the most striking developments over the last several decades regarding numerical solutions of PDEs has been the increased use of high-order methods. At the beginning of the era of digital computing, first or second order of accuracy was the norm. Although this sufficed for many pioneering calculations, and is still sometimes used, higher order methods can be far more efficient. Their primary limitation so far has been difficulties in cases with non-trivial domain shapes. The main aspect of RBF that we will focus on in this chapter is how they are well on the way to become the long sought after tool for generalizing high-order finite difference (FD) and pseudospectral (PS) methods to such situations. For the first time, freedom from mesh generation, ability to do local refinements, and easy handling of irregular geometries can all be combined with spectral accuracy.

After providing some general RBF background in Sections 14.2— 14.3, we discuss in Section 14.4 a major major computational issues: Stable algorithms. Following this, our attention will turn to RBF for PDEs, starting in Section 14.5 with a very brief overview of pseudospectral (PS) methods. This is followed in Section 14.6 by a derivation of finite difference methods using RBFs. Section 14.7 addresses more briefly a large number of additional RBF topics.

14.2. Introduction to RBF via cubic splines.

Figure 14.2.1(a) shows a function that is sampled at equispaced nodes over $[-1, 1]$, and the interpolating cubic spline (using Matlab's default not-a-knot end conditions). Part (b) displays the error of this interpolation.

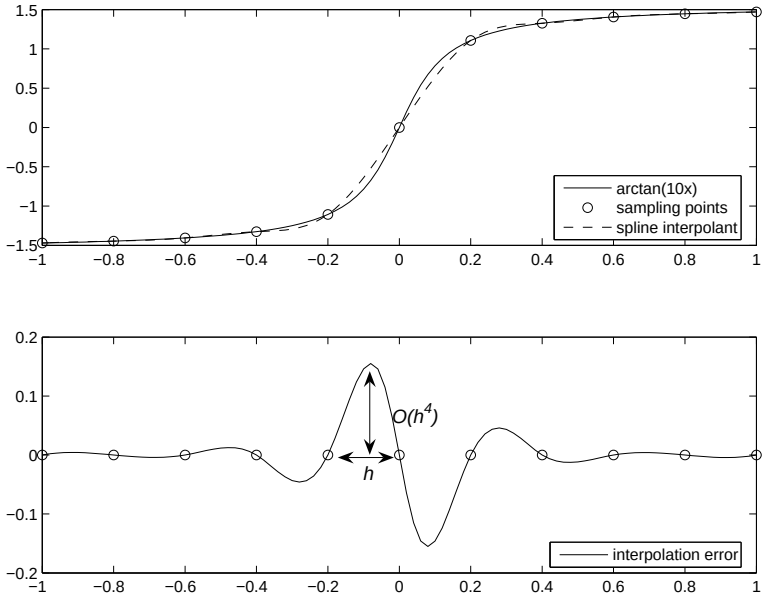


FIGURE 14.2.1. (a) The function $\arctan(10x)$ and its cubic spline interpolant. (b) The interpolation error.

A standard cubic spline is made up of a different cubic polynomial between each pair of adjacent node points, and it may at these points feature a jump in the *third* derivative (the function, and its first two derivatives are continuous everywhere). The standard approach for computing the coefficients of the different cubics which form the spline requires only the solution of a tridiagonal linear system [**?**]. If the

spacing between the sample points is h, it is well known that the size of the error will decrease like $O(h^4)$. It makes almost no difference in the algorithm if the nodes are not equally spaced. However, generalizations to more space dimensions have in the past been practical only if the nodes are lined up in the coordinate directions.

Another way to approach the problem of finding the 1-D cubic spline (for now omitting to address the issue of end conditions) is the following: At each data location x_j, $j = 1, \ldots, n$, place a translate of the function $\phi(x) = |x|^3$, i.e. at location x_j the function $\phi(x-x_j) = |x-x_j|^3$. We then ask if it is possible to form a linear combination of all these functions

$$s(x) = \sum_{j=1}^{n} \lambda_j \, \phi(x - x_j) \tag{14.1}$$

such that this combination takes the desired function values f_j at the data locations x_j, $j = 1, \ldots, n.$, i.e. enforcing $s(x_j) = f_j$. This amounts to asking for the coefficients λ_j to satisfy the linear system of equations

$$\begin{bmatrix} \phi(x_1 - x_1) & \phi(x_1 - x_2) & \cdots & \phi(x_1 - x_n) \\ \phi(x_2 - x_1) & \phi(x_2 - x_2) & & \phi(x_2 - x_n) \\ \vdots & & & \vdots \\ \phi(x_n - x_1) & \phi(x_n - x_2) & \cdots & \phi(x_n - x_n) \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_n \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix}. \tag{14.2}$$

Assuming that this system is non-singular, it can be solved for the coefficients λ_j. The interpolant $s(x)$, as given by (14.1), will then become a cubic function between the nodes and, at the nodes, have a jump in the third derivative. We have thus found another way to create an interpolating cubic spline. This time, it looks like that we have to solve a full (although symmetric) linear system. However, as we will see next, this formulation opens up powerful opportunities for generalizing the form of the interpolant, and also for extending the methodology to scattered data in any number of space dimensions.

14.2.1. Generalization to multiple dimensions. Figure 14.2.2(a) illustrates the RBF idea in 1-D. At each data location x_j, we centered a translate of our symmetric function $\phi(x)$.

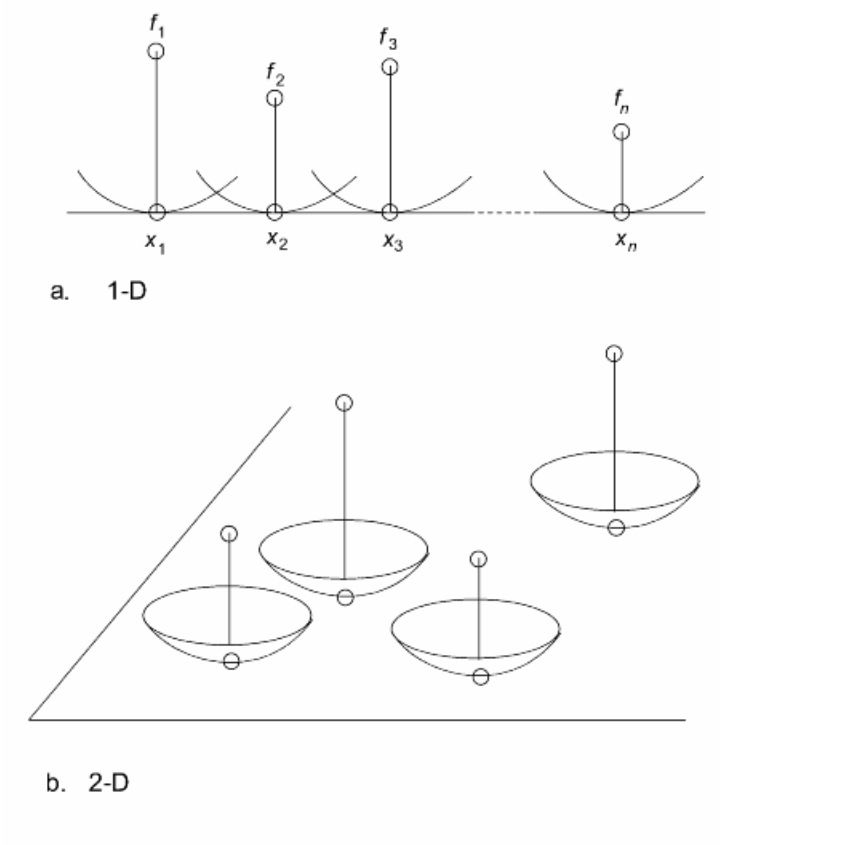


FIGURE 14.2.2. Illustration of the RBF concept in 1-D and in 2-D.

In 2-D, as illustrated in Figure 14.2.2(b), we instead use a *rotated* version of the same radial function. In d dimensions, we can write these rotated basis functions as $\phi(||\underline{x} - \underline{x}_j||)$, where $\|\cdot\|$ denotes the standard Euclidean norm. The form of the RBF interpolant and of the linear system that is to be solved has hardly changed from the 1-D case. Instead of (14.1) and (14.2), we now use as interpolant

$$s(\underline{x}) = \sum_{j=1}^{n} \lambda_j \, \phi(\left\|\underline{x} - \underline{x}_j\right\|) \tag{14.3}$$

with the collocation conditions

$$(14.4)\qquad \begin{bmatrix} \phi(\|\underline{x}_1-\underline{x}_1\|) & \phi(\|\underline{x}_1-\underline{x}_2\|) & \cdots & \phi(\|\underline{x}_1-\underline{x}_n\|) \\ \phi(\|\underline{x}_2-\underline{x}_1\|) & \phi(\|\underline{x}_2-\underline{x}_2\|) & & \phi(\|\underline{x}_2-\underline{x}_n\|) \\ \vdots & & & \vdots \\ \phi(\|\underline{x}_n-\underline{x}_1\|) & \phi(\|\underline{x}_n-\underline{x}_2\|) & \cdots & \phi(\|\underline{x}_n-\underline{x}_n\|) \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_n \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix}.$$

In particular, we note that the algebraic complexity of the interpolation problem has not increased with the number of dimensions - we will always end up with a square symmetric system of the same size as the number of data points. Cubic splines have thus been generalized to apply also to scattered data in any number of dimensions.

A generalized version of (14.3) will be introduced at the end of this chapter (equations (14.9) and (14.10)).

14.2.2. Different types of radial functions. The error $O(h^4)$ for cubic splines in 1-D will become $O(h^6)$ in the case of quintic splines. And it falls to $O(h^2)$ for linear splines (corresponding to $\phi(x) = |x|$). In general, if we take the RBF approach as outlined above, and use $\phi(x) = |x|^{2m+1}$, the error will become $O(h^{2m+2})$ (even powers in $\phi(x)$ will not work; for ex. if $\phi(x) = x^2$, the interpolant (14.1) will reduce to a single quadratic polynomial, no matter the value of n, and attempting to interpolate more than three points will have to give rise to a singular system). The sizes of these errors correspond directly to which derivative of $\phi(x)$ it is that features a jump. This leads to the 'obvious' question: why not choose a $\phi(x)$ which is infinitely differentiable everywhere, such as $\phi(x) = \sqrt{1+x^2}$? This idea is an excellent one, which can be applied to good advantage in any number of dimensions. If we still ignore boundary issues (possibly leading to some counterpart for RBF of the Runge phenomenon (RP) for polynomials, to be discussed in Section 14.7), the accuracy will become spectral: better than any polynomial order, and generally of the form $O(e^{-c\,n})$, where $c > 0$ and n is the number of points. Some precise statements and proofs in this regard (however without describing in-between node point oscillations as a manifestation of RP) have been given in [**?**], [**?**].

Table 1 lists a number of possible choices of radial functions, with illustrations shown in Figure 14.2.3.

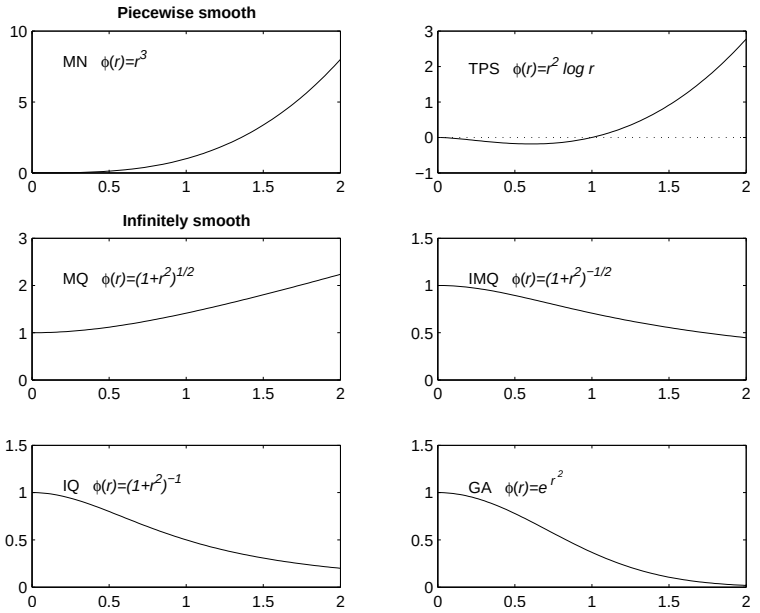


FIGURE 14.2.3. Illustrations of the eight different radial functions in Table 1.

In the piecewise smooth category (where we so far have discussed only MN: $\phi(r) = |r|^{2m+1}$), other interesting choices include TPS $\phi(r) = |r|^{2m} \ln\ |r|$, and also the WE and MT classes. The TPS RBF are commonly used in 2-D, especially then with $m = 1$. Just like how the natural cubic spline (obeying the end conditions $s''(a) = s''(b) = 0$) minimizes $\int_a^b [s''(x)]^2 dx$ over all possible interpolants [**?**], the RBF interpolant using $\phi(r) = r^2 \log r$ achieves an equivalent minimization for 2-D scattered data [**?**], [**?**]. The most notable feature of the Wendland functions is their *compact support*, i.e. being nonzero only over a local region of radius $1/\varepsilon$. Unless ε is very small, the matrix in (14.4) will be sparse, and (14.3) will contain well less than n terms. This can greatly speed up RBF calculations. However, as discussed further in Section 14.4, this advantage may be offset by a significant loss in accuracy. The MT functions turn out to be of particular interest in many statistical applications [??].

The spectral accuracy noted above for smooth radial functions holds in any number of spatial dimensions. For the non-smooth ones, the order accuracy curiously improves with the number of dimensions. For example, for $\phi(r) = |r|^{2m+1}$ and for

$\phi(r) = |r|^{2m} \ln\ |r|$, the order of accuracy (in the ...-norm) becomes ..... and ..... respectively [?]. Intuitively, this can be understood from the fact that ..... (?). Check

For all the types of RBF, apart from MN and TPS, we have also introduced a *shape parameter*, which is commonly denoted by ε. For small values of ε, the basis functions become very flat and, for large values, they become sharply spiked (e.g. for IQ, IMQ, GA) or, in the case of MQ, it approaches the piecewise linear case. Although the extremes (ε very small, and ε very large) would at first both seem to be unsuitable, we will soon see that the former case will be of particular interest in connecting RBF with pseudospectral (PS) methods.

14.2.3. Fourier representation of radial functions. In some of the subsequent analysis, the Fourier transforms of the RBF will be of interest. Table 1 also shows these (or their generalized Fourier transform, if the integral that defines the regular version would be divergent, cf. [?], [?]). We use in this chapter the 1-D convention $u(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \widehat{u}(\omega)\ e^{i\omega x} d\omega$; $\widehat{u}(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} u(x)\ e^{-i\omega x} dx$, and its generalization to higher dimensions. When a function in d dimensions is radially symmetric (i.e. depends on $r = \sqrt{x_1^2 + x_2^2 + \ldots + x_d^2}$ only), the Fourier transform will similarly depend only on $\rho = \sqrt{\omega_1^2 + \omega_2^2 + \ldots + \omega_d^2}$. The computation of the d-dimensional transforms is then known as a Hankel transform (identical to its own inverse):

$$\begin{aligned} \widehat{\phi}(\rho) &= \tfrac{1}{(2\pi)^{d/2}} \int_{-\infty}^{\infty} \cdots \int_{-\infty}^{\infty} \phi(\|\underline{x}\|)\ e^{-i\,\underline{\omega}\,\cdot\,\underline{x}}\, d\underline{x} \\ &= \tfrac{1}{\rho^{(d-2)/2}} \int_0^{\infty} \phi(r)\ r^{d/2}\ J_{(d-2)/2}(r\ \rho)\ dr \end{aligned} \quad .$$

Alternatively, the Hankel transform can be computed by the formulas

$$\begin{aligned} &d = 2m+1 \text{ odd:} && \widehat{\phi}(\rho) = (-2)^m \sqrt{\tfrac{2}{\pi}} \tfrac{d^m}{d(\rho^2)^m} \int_0^{\infty} \phi(r) \cos(r\rho)\ dr \\ &d = 2m+2 \text{ even:} && \widehat{\phi}(\rho) = (-2)^m \tfrac{d^m}{d(\rho^2)^m} \int_0^{\infty} \phi(r)\ r\ J_0(r\rho)\ dr \end{aligned} \quad ,$$

involving derivatives, but no Bessel functions of higher than zeroth order.

14.2.4. Non-singularity of the RBF system. We denote the coefficient matrix in (14.4) by A. The issue whether this matrix can ever become singular is of critical importance for the utility and robustness of RBF in approximations of the

Type of radial function			Fourier transform $\widehat{\phi}(\rho)$ in d-D
Piecewise smooth			
MN	monomial	$\lvert r\rvert^{2m+1}$	$\frac{(-1)^{m+1}2^{2m+\frac{d}{2}+1}\Gamma(m+\frac{d+1}{2})\Gamma(m+\frac{3}{2})}{\pi}\frac{1}{\lvert\rho\rvert^{2m+d+1}}$
TPS	thin plate spline	$\lvert r\rvert^{2m}\ln\ \lvert r\rvert$	$(-1)^{m+1}2^{2m+\frac{d}{2}-1}m!\Gamma(m+\frac{d}{2})\frac{1}{\lvert\rho\rvert^{2m+d}}$
WE	Wendland	.....	.....
MT	Matérn	.....	.....
Infinitely smooth			
MQ	multiquadric	$\sqrt{1+(\varepsilon r)^2}$	.....
IQ	inverse quadratic	$\frac{1}{1+(\varepsilon r)^2}$	.....
IMQ	inverse MQ	$\frac{1}{\sqrt{1+(\varepsilon r)^2}}$	.....
GA	Gaussian	$e^{-(\varepsilon r)^2}$	$\frac{e^{-\rho^2/(4\varepsilon^2)}}{(\sqrt{2}\varepsilon)^d}$

TABLE 1. Definition and Fourier transforms for some cases of radial functions.

form (14.3). We will show in Theorem 53 that a sufficient condition for unconditional non-singularity (no matter how any number of points are distributed in any number of dimensions) is that the radial function's Fourier transform $\widehat{\phi}(\rho)$ is positive. This result will immediately guarantee the non-singularity for the WE, MT, IQ, IMQ and GA cases. The result holds also for IQ, as was shown by Micchelli in 1986 [**?**], with the first elementary proof only quite recent [..]. We will give the proof of non-singularity for all cases with $\widehat{\phi}(\rho)$ positive in two stages. Lemma 52 carries it out in the special case of GA, and we note in the proof of Theorem 53 that the change is minor when generalizing to other cases with $\widehat{\phi}(\rho) > 0$. These results were obtained (although in a somewhat different context) already in the early 1930s ([**?**], see also [**?**]):

THEOREM 52. *The RBF matrix A is positive definite (in particular, nonsingular) in the case of GA RBF.*

PROOF. The standard proof is carried out in two steps:

1. Show that A is positive semidefinite, and
2. If a set of points $\underline{x}_m$ are distinct, show that the function

$$f(\underline{x}) = \sum_{m=1}^{n} \alpha_m \, e^{-i \, \underline{x}\cdot\underline{x}_m} \tag{14.5}$$

cannot be identically zero unless all the coefficients α_m are all zero. *Step 1:* Every

entry of A is then of the form $e^{-\varepsilon^2\|\underline{\omega}\|^2}$ for some vector $\underline{\omega}$ (i.e. for $A_{j,k}$ we choose $\underline{\omega} = \underline{x}_j - \underline{x}_k$). The exponent can be simplified to be just linear in $\underline{\omega}$ if we note (cf. the Fourier transform of Gaussians, as listed in Table 1)

$$e^{-\varepsilon^2\|\underline{\omega}\|^2} = \tfrac{1}{(4\pi)^{d/2}\varepsilon^d} \int_{R^d} e^{-\|\underline{x}\|^2/4\varepsilon^2} \;\; e^{-i \, \underline{x}\cdot\underline{\omega}} \, d\underline{x} \tag{14.6}$$

If $\underline{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_n)^T$ is an arbitrary column vector, we thus find that

$$\begin{aligned} \underline{\alpha}^T \, A \, \underline{\alpha} \;\; &= \textstyle\sum_{j=1}^{n} \sum_{k=1}^{n} \;\alpha_j\alpha_k \; e^{-\varepsilon^2\left\|\underline{x}_j - \underline{x}_k\right\|^2} \\ &= \textstyle\sum_{j=1}^{n} \sum_{k=1}^{n} \;\alpha_j\alpha_k \; \tfrac{1}{(4\pi)^{d/2}\varepsilon^d} \int_{R^d} e^{-\|\underline{x}\|^2/4\varepsilon^2} \;\; e^{-i \, \underline{x}\cdot\,(\underline{x}_j - \underline{x}_k)} \, d\underline{x} \\ &= \tfrac{1}{(4\pi)^{d/2}\varepsilon^d} \int_{R^d} e^{-\|\underline{x}\|^2/4\varepsilon^2} \textstyle\sum_{j=1}^{n} \sum_{k=1}^{n} \;\alpha_j\alpha_k \; e^{-i \, \underline{x}\cdot\,(\underline{x}_j - \underline{x}_k)} \, d\underline{x} \end{aligned} \tag{14.7}$$

The double sum in the integrand can be written as

$$\left(\sum\nolimits_{j=1}^{n} \alpha_j \; e^{-i \, \underline{x}\cdot\,\underline{x}_j}\right) \left(\sum\nolimits_{k=1}^{n} \alpha_k \; e^{i \, \underline{x}\cdot\,\underline{x}_k}\right) = \left|\sum\nolimits_{m=1}^{n} \alpha_m \; e^{-i \, \underline{x}\cdot\,\underline{x}_m}\right|^2 \geq 0. \tag{14.8}$$

Since $e^{-\|\underline{x}\|^2/4\varepsilon^2}$ (in the integral) is everywhere positive, this shows that $\underline{\alpha}^T A \underline{\alpha} \geq 0$, i.e. A is a positive semidefinite matrix. It remains to show that we can't have equality if any component of $\underline{\alpha}$ is non-zero. *Step 2:* Since the points $\underline{x}_m$ are assumed to be

distinct, we can choose m^* such that $\underline{x}_{m^*} \cdot \underline{x}_{m^*} > \underline{x}_{m^*} \cdot \underline{x}_k$, $k \neq m^*$. Let $\underline{x} = \sigma \, \underline{x}_{m^*}$. Assuming further that $f(\underline{x}) \equiv 0$ (cf. (14.5)), so is $g(\sigma) = \sum_{m=1}^{n} \alpha_m \; e^{-i \, \sigma\underline{x}_{m^*}\cdot\underline{x}_m} \equiv 0$. If $\alpha_{m^*} \neq 0$, the term $\alpha_{m^*} \; e^{-i \, \sigma\underline{x}_{m^*}\cdot\underline{x}_{m^*}}$ will outgrow all other terms if we differentiate $g(\sigma)$ increasingly many times. This is impossible in view of $g(\sigma) \equiv 0$. Hence $\alpha_{m^*} = 0$. The argument can then be repeated to show that all coefficients α_m are equal to zero.

□

Other proofs for the Step 2 can be found for ex. in [**?**], [..]. A slight modification of Theorem 52 is particularly useful when considering RBF of compact support (such as Wendland-type radial functions):

THEOREM 53. *The RBF matrix A is positive definite if* $\int_{R^d} \phi(\|\underline{x}\|)^2 d\underline{x}$ *is finite* $\widehat{\phi}(\rho)$ *is positive.*

PROOF. Instead of (14.6), we start with the relation

$$\phi(\|\underline{\omega}\|) = \frac{1}{(2\pi)^{d/2}} \int_{R^d} \widehat{\phi}(\|\underline{x}\|) \;\; e^{i\,\underline{x}\cdot\underline{\omega}}\; d\underline{x}\;,$$

and proceed similarly. We will again arrive at (14.8); the only (insignificant) difference being that the double sum within the integral in the last line of (14.7) will be preceded by another positive function than $e^{-\|\underline{x}\|^2/4\varepsilon^2}$. □

The two Theorems 52 and 53 rely on $\int_{R^d} \phi(\|\underline{x}\|)^2 d\underline{x}$ to be finite, greatly limiting the types of radial functions they can be applied to. The concept of *completely monotone* (CM) functions leads to non-singularity proofs for many more types of radial functions:

DEFINITION 54. A $C^\infty(0,\infty)$ function $\psi(r)$, which has a bounded first derivative at the origin, is said to be completely monotone if $(-1)^k \frac{d^k}{dr^k \psi(r)} \geq 0$ for $r > 0$ and $k = 0, 1, \ldots$

LEMMA 55. *A function* $\psi(r), r \geq 0$, *is completely monotone if and only if its inverse Laplace transform* $\gamma(s)$ *is nonnegative (i.e.* $\gamma(s) \geq 0$ *when* $\psi(r) = \int_0^\infty \gamma(s)\; e^{-sr} ds$*)*

Proofs of Lemma 55 and can be found for ex. in [..], [..].

THEOREM 56. *If* $\psi(r)$ *is completely monotone (but not constant), then the RBF matrix A using the radial function* $\phi(r) = \psi(r^2)$ *will be positive definite.*

PROOF. Let again $\underline{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_n)^T$ is an arbitrary column vector. We then obtain

$$\begin{aligned} \underline{\alpha}^T\ A\ \underline{\alpha} &= \sum_{j=1}^n \sum_{k=1}^n\ \alpha_j \alpha_k\ \psi(\left\|\underline{x}_j - \underline{x}_k\right\|^2) \\ &= \sum_{j=1}^n \sum_{k=1}^n\ \alpha_j \alpha_k\ \int_0^\infty \gamma(s)\ e^{-s\left\|\underline{x}_j - \underline{x}_k\right\|^2}\ ds \\ &= \int_0^\infty \gamma(s) \sum_{j=1}^n \sum_{k=1}^n\ \alpha_j \alpha_k\ e^{-s\left\|\underline{x}_j - \underline{x}_k\right\|^2}\ ds\ . \end{aligned}$$

The double sum is positive according to Theorems 52 and $\gamma(s) \geq 0$ (and not identically zero). Hence $\underline{\alpha}^T\ A\ \underline{\alpha} > 0$. □

EXAMPLE. With $\phi(r) = \frac{1}{(1+r^2)^\beta}$, i.e. $\psi(r) = \frac{1}{(1+r)^\beta}$, it will hold that $(-1)^k \frac{d^k}{dr^k}\psi(r) = \left(\sum_{i=0}^{k-1}(\beta+i)\right)\ /\ (1+r)^{\beta+k},\ k = 0, 1, \ldots$ Hence, by the Definition 54, $\psi(r)$ is completely monotone if $\beta > 0$. Lemma 55 confirms this, since $\gamma(s) \equiv e^{-s}s^{\beta-1}/\Gamma(\beta)$, which is positive for s positive, again when $\beta > 0$. Therefore, by Theorem 56, the A-matrix for $\phi(r) = \frac{1}{(1+r^2)^\beta}$ is always positive definite (for any point distributions in any number of space dimensions) when $\beta > 0$. ■

EXAMPLE. In the case of $\phi(r) = \text{sech}\ \varepsilon r$, the closed-form expressions for $\widehat{\phi}(\rho)$ become too difficult to work with if the dimension d is large, and Theorem 53 is therefore not easily applicable. Nevertheless, Theorem 56 will readily show that we again always get a positive definite A-matrix (for details, see [**?**]; another demonstration of the result can be found in [..]). ■

The implication in Theorem 56 goes both ways—$\psi(r)$ being CM is in fact both necessary and sufficient for the A-matrix to be guaranteed positive definite for any number of space dimensions. From this follows immediately that the A-matrix cannot be unconditionally positive definite if $\phi(r)$ has a zero for $r \geq 0$. In particular, any radial function with compact support (such as the Wendland functions) are always limited to some certain number of dimensions.

The MQ case of $\phi(r) = \sqrt{1 + (\varepsilon\ r)^2}$ is more difficult. In this case, the A-matrix is no longer positive definite. However, it is still non-singular, having one positive and $n - 1$ negative eigenvalues. Micchelli's celebrated proof of this result in 1986 [**?**] played a key role in confirming MQ RBF as a method of choice for multidimensional interpolation, and this formed the starting point for much of the recent interest in

RBF. A very much simplified proof of this result was found only recently (see [..] for details).

In the (piecewise smooth) cases of MN and TPS (or just the $m = 1$ cases?), check singularities of the A-matrix can arise if we use (14.3), but not if we instead use the almost equivalent form

$$s(\underline{x}) = \lambda_0 + \sum_{j=1}^{n} \lambda_j \, \phi(\|\underline{x} - \underline{x}_j\|), \tag{14.9}$$

together with the constraint $\sum_{j=1}^{n} \lambda_j = 0$. This extension, allowing constants to be interpolated exactly, can be advantageous also in cases smooth RBF. It can also be extended further still by letting $\{p_k(\underline{x})\}_{k=1}^{m}$ be a basis for the space of all d-variate polynomials that have degree $\leq Q$, and then use as interpolant

$$s(\underline{x}) = \sum_{j=1}^{n} \lambda_j \, \phi(\|\underline{x} - \underline{x}_j\|) + \sum_{k=1}^{m} \beta_k p_k(\underline{x}) \tag{14.10}$$

where the expansion coefficients λ_j and β_k are determined by enforcing $s(\underline{x}_j) = f_j$, $j = 1, \ldots, n$, together with the constraints $\sum_{j=1}^{n} \lambda_j \, p_k(\underline{x}_j) = 0$, $k = 1, \ldots, m$ (see for example [**?**] and [??] for results regarding non-singularity).

14.3. The shape parameter ε.

The literature on selecting a good (single) value for ε is extensive, e.g. [**?**], [**?**], [**?**], [**?**], [**?**], [**?**], [**?**], [**?**]. Most of these works focus on finding the minimal error in computations on various applications. For large values of ε, for ex. a GA interpolant consists of a many narrow spikes, reaching up to each function value that is to be interpolated, i.e. the interpolant in-between data points will be very inaccurate. Hence, a decrease in ε is initially beneficial. Tests tend to show that these improvements will cease at some point, after which errors will tend to increase again as $\varepsilon \to 0$. There are two main causes for this reversal in trend; numerical ill-conditioning, and a Runge-type phenomenon. They will be discussed in the next two subsections. We will find that both causes can be at least partly overcome, thereby making the low ε-regime of great practical interest.

14.3.1. Potential advantages of using near-flat basis functions. It was demonstrated in [**?**] that, in the limit of $\varepsilon \to 0$, the RBF interpolants in 1-D in general converge to the Lagrange interpolating polynomial. Since these (lowest degree) interpolating polynomials in turn form the basis for all classical pseudospectral (PS) methods, this implies that PS methods alternatively can be viewed as special cases of RBF methods [**?**]. Already in 1-D, this viewpoint can offer new opportunities because use of RBF with $\varepsilon > 0$ can be both more accurate and more stable than the 'classical' PS methods (which correspond to the $\varepsilon = 0$ limit in conjunction with certain very restricted types of node layouts, often based on zeros or extrema of orthogonal polynomials). However, the most striking advantages come in 2-D (and higher) with the new ability of then using scattered node layouts over irregularly shaped domains. This allows PS methods to be generalized from very restrictive domain shapes and tensor-type grids only over to fully irregular domains with scattered nodes.

In the $\varepsilon \to 0$ limit, the conditioning of the linear system (14.4) degrades rapidly. For example, with 41 scattered nodes in 2-D, $\det(A)$ is proportional to ε^{416} as $\varepsilon \to 0$ for all the infinitely smooth radial functions listed in Table 1 [**?**]. The expansion coefficients λ_i become oscillatory and grow rapidly in magnitude with decreasing ε (proportionally to $1/\varepsilon^{16}$ in this example). The subsequent evaluation of the interpolant by means of (14.3) will then involve large amounts of numerical cancellations. It was for some time believed that a trade-off between high accuracy and good conditioning was inevitable (expressed in terms of an "uncertainty principle" [**?**], see also Section 5.3.4 in [**?**]). Utilizing contour integration in a complex ε-plane, the Contour-Padé algorithm [**?**] became the first method able to bypass this ill-conditioning, thereby permitting stable computations of RBF interpolants all the way down to $\varepsilon = 0$. Maybe even more importantly, this algorithm demonstrated that the previously feared ill-conditioning barrier against the use of small ε merely amounted to equations (14.4) and (14.3) being an ill-conditioned approach for the well-conditioned task of evaluating the RBF interpolant $s(\underline{x})$ based on data values f_i at locations $\underline{x}_i,\ i = 1, 2, \ldots, n$. By means of the still more recent RBF-QR algorithm [**?**], [**?**], the Contour-Padé limitation on the number of nodes n (to be no more than around 100-200 nodes in 2-D) is essentially eliminated. Both of these methods will be described further in Section 14.4.

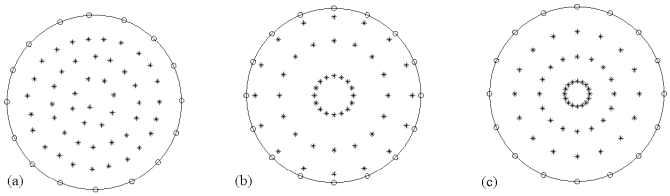


FIGURE 14.3.1. Three different node distributions, all with 16 nodes on the boundary and 48 nodes in the interior of the unit circle, used by the FD2, PS, and RBF methods, respectively, in the Poisson equation test case.

The potential benefits of computing in a very low ε-range were strikingly illustrated in [**?**]. One of the test cases considered there was to solve Poisson's equation $\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial x^2} = f(x, y)$ over the unit circle using straightforward RBF collocation, with a right hand side $f(x, y)$ and Dirichle boundary conditions chosen so that $u(x, y) = 65/(65 + (x - 0.2)^2 + (y + 0.1)^2)$ becomes the solution. With nodes for a second order finite difference scheme (FD2), a Fourier-Chebyshev PS scheme, and RBF laid out as illustrated in parts a-c respectively of Figure 14.3.1, the max norm errors become as seen in Figure 14.3.2(a) when calculated with the Contour-Padé algorithm. Direct solution of the RBF collocation equations by Gaussian elimination (Figure 14.3.2(b)) gives the same results for large ε, but the accuracy is lost due to ill-conditioning for low ε.

The RBF approach features much more flexibility than the FD2 and PS alternatives in that the results depend very little on how the nodes are scattered over domains, which furthermore need not be circular but can be arbitrarily shaped. The RBF errors are seen to be many orders of magnitude smaller (than those of FD2 and PS) as soon as ε is chosen sufficiently small. For the FD2 scheme, halving the typical space step will increase the accuracy by a factor of four - and (in 2-D) require 4 times as many nodes. Increasing the accuracy by a factor of 10^6 (to make the FD2 calculation as accurate as the RBF ones) would thus require approximately $64 \cdot 10^6$ rather than 64 node points.

Figures 14.3.2(a) and (b) indicate that, even when ill-conditioning is eliminated for low ε values, there still remains some additional effect that breaks the trend of

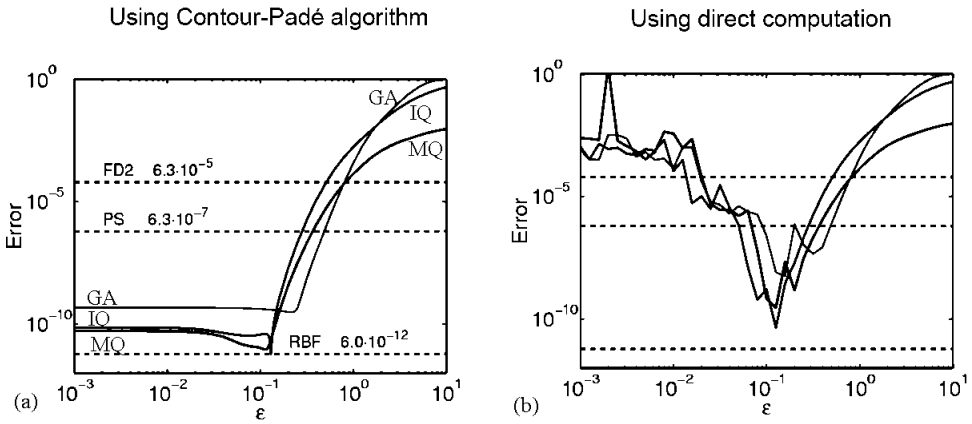


FIGURE 14.3.2. The max norm errors in the Poisson equation test case, as functions of ε when using MQ, IQ, and GA RBFs. The errors (not ε-dependent) for standard second order finite differences (FD2) and Fourier-Chebyshev pseudospectral (PS) method are also included for comparison: (a) Computation using Contour-Padé method, (b) Direct implementation via Gaussian elimination.

errors decreasing with decreasing ε. In the next subsection, we will focus on he cause for this.

14.3.2. The Runge phenomenon (RP).

14.3.2.1. *The RP for polynomials.* The best-known case of the RP occurs for increasing order polynomial interpolation on equispaced grids, and is illustrated in Figure 14.3.3. Convergence / divergence rates as the number of node points increases will depend on x-position, node distribution but only to a limited extent on properties of the interpolated function (the only relevant quantity being how far away from $[-1, 1]$ the function can be analytically continued without encountering any singularities). The theory for this is well understood [**?**], [**?**].

For example, in the case shown in Figure 14.3.3, the envelope of the oscillatory error varies proportionally to

$$E(z, n) = e^{n\,(\psi(z_0)-\psi(z))} \tag{14.1}$$

(both for $z = x$ real, and for z complex), where the logarithmic potential function

$$\psi(z) = -\frac{1}{2}\mathrm{Re}\left[(1-z)\ln(1-z) - (-1-z)\ln(-1-z)\right] \tag{14.2}$$

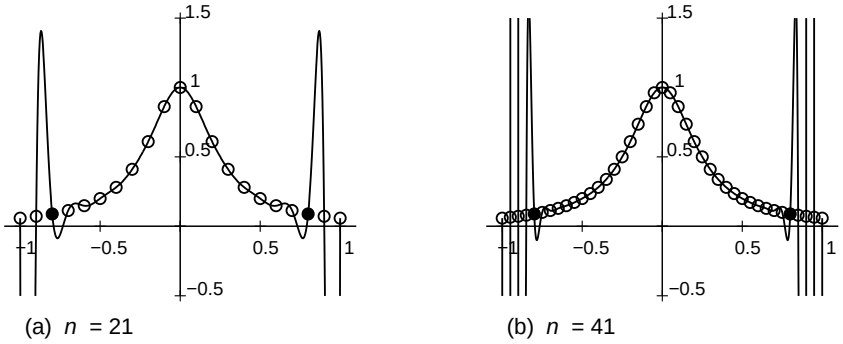


FIGURE 14.3.3. Equispaced polynomial interpolation of $f(x) = \frac{1}{1+16x^2}$ over [-1,1]. As n increases, there is spectral convergence for $|x| < 0.7942$ and exponential divergence otherwise. These transition points are marked by solid dots.

is generic for equispaced polynomial interpolation over [-1,1]. The function $f(x)$ that is interpolated enters only by setting $z_0 = 0.25i$ in (14.1); a singularity in the complex plane of $f(z) = 1/(1 + 16z^2)$.

The standard remedy against the RP is Chebyshev-type clustering of nodes towards the end of the interval, e.g.

$$x_j = -\cos(\frac{\pi(j-1)}{n-1}),\ j = 1, 2, \ldots, n. \tag{14.3}$$

In that case, one then obtains, in place of (14.2), $\psi(z) = -\ln|z+\sqrt{z^2-1}|$, which for $z = x$ real, $-1 \le x \le 1$, evaluates to zero. Because of (14.1), this corresponds to the well-known uniform Chebyshev interpolation accuracy across $[-1, 1]$ for all functions $f(x)$. However, while this particular node distribution resolves one difficulty (the RP), it introduces others. For example, in the context of time stepping PDEs, the CFL stability condition can become very severe.

14.3.2.2. *The RP for RBF approximations.* Figure 14.3.4 illustrates how the RBF interpolation error at first decreases and then increases when ε goes from very large to very small. For ε large, each GA basis function consists of a sharp spike, with a height such that it just reaches up to the corresponding function value. The errors this mechanism causes vanish quickly for decreasing ε, but another type of error is

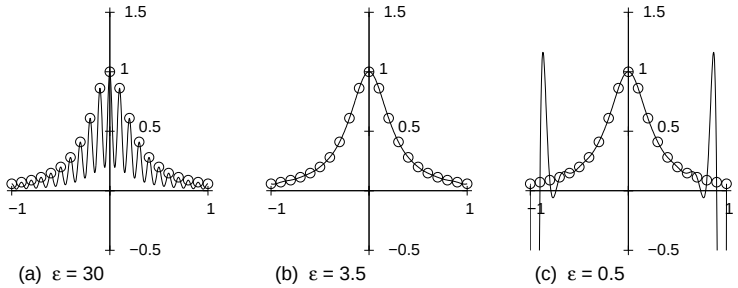


FIGURE 14.3.4. GA interpolants of $f(x) = \frac{1}{1+16x^2}$ for a wide range of ε-values: $\varepsilon = 30,\ 3.5$, and 0.5 respectively.

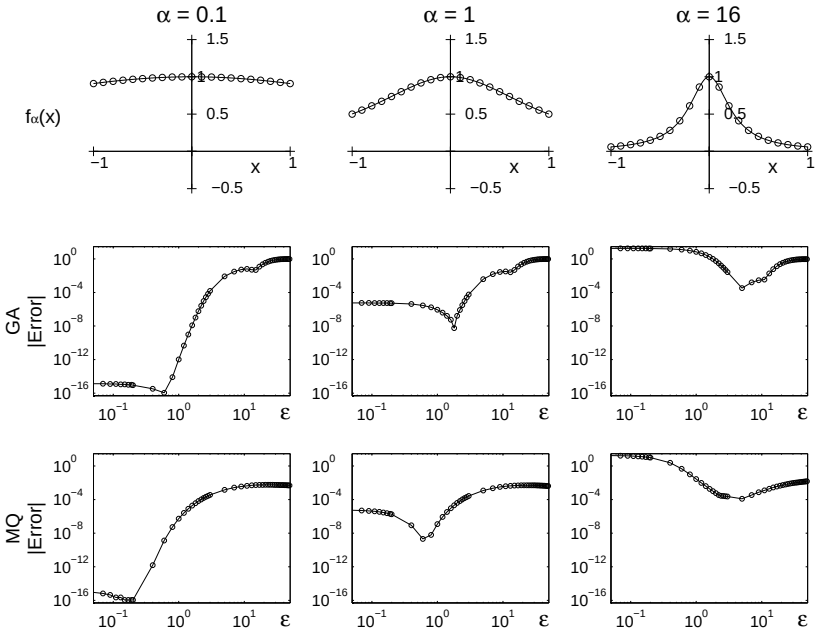


FIGURE 14.3.5. Top row: The function $f_\alpha(x) = \dfrac{1}{1+\alpha x^2}$ for three values of α. Next two rows of subplots show how the error varies with ε in the case of GA and MQ RBFs respectively.

seen to enter for very small ε - the RP, as an immediate consequence of the fact that the interpolant then approaches the polynomial one.

The six subplots in Figure 14.3.5 illustrate how the smoothness of the interpolant influences when the trend reversal occurs, and how strong this reversal will be. The

third case ($\alpha = \frac{1}{16}$) uses the same test function as is shown in Figures 14.3.3 and 14.3.4. The Runge phenomenon enters in all cases once ε is sufficiently small, and its level at $\varepsilon = 0$, as obtained from the polynomial RP theory quoted in Section 14.3.2.1, agrees completely with the lowest ε-results in Figure 14.3.5. Although higher accuracy can be reached if the data is smoother, the RP still in all cases breaks a very favorable improvement trend for decreasing ε.

The discussion above provides an understanding of the trend reversal that was illustrated earlier (in a slightly different context) in Figure 14.3.2 a. Although this trend reversal was described theoretically in [**?**], the present RP interpretation of it (first presented in [**?**]) is much more intuitive.

A few possible approaches for reducing the accuracy loss due to RP at boundaries have already been suggested in the literature:

(1) Node clustering at edges (or wherever needed to improve accuracy),
(2) Super Not-a-Knot (SNaK) generalization of cubic spline Not-a_Knot end conditions,
(3) Spatially variable shape parameter; use ε_j at node location $\underline{\underline{x}}_j$.

The first option is the only one available if one uses polynomial interpolants. The discussion in [**?**] suggests that SNaK is preferable to node clustering in the context of RBF. The idea of suppressing RP by use of spatially variable shape parameters is considered in [**?**], and is discussed briefly later in this chapter in Section 14.7.

While boundaries are a common trigger of RP, it is not the only one. It can also arise if one attempts to improve local accuracy by clustering nodes in select areas, as is a standard procedure with finite elements or splines.An extensive discussion of this effect (and a possible remedy) can be found in [**?**]. This effect is illustrated in Figure 14.3.6. In part a, there is no RP visible, but the equispaced RBF approximation lacks sufficient resolution near the center, where the data features a very sharp gradient. In part b, we have inserted two extra nodes in the critical area and, in part c, still two more points are inserted. The most striking result of this local refinement is a seemingly disastrous RP. We will next see how this can be brought under control.

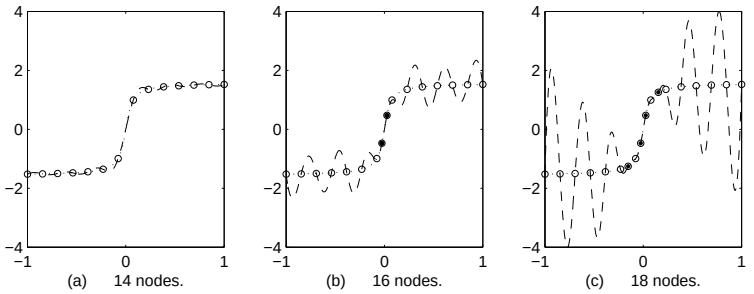


FIGURE 14.3.6. MQ RBF $\varepsilon = 2$ interpolants (dashed curves) of $f(x) = \arctan(20x)$ (dotted curves) over [-1,1] (a) 14 equispaced points, (b) two extra points inserted near the center, and (c) still two more points inserted near the center.

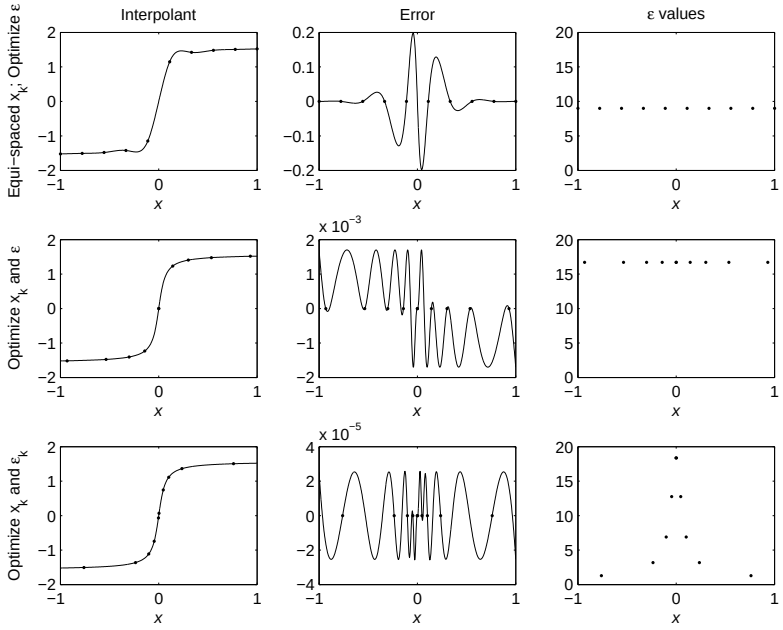


FIGURE 14.3.7. Ten-node MQ interpolations of $f(x) = \arctan 20x$. Top row: Equispaced nodes, ε (same at all nodes) optimized. Middle row: Node locations x_k and ε (same at all nodes) optimized. Bottom row: Both x_k and ε_k optimized.

14.3.2.3. *Example of RP control.* Figure 14.3.7 shows that one can obtain excellent accuracy in the $f(x) = \arctan 20x$ test case already with very few nodes if one just uses good choices for their locations x_j and, better still (bottom row of subplots),

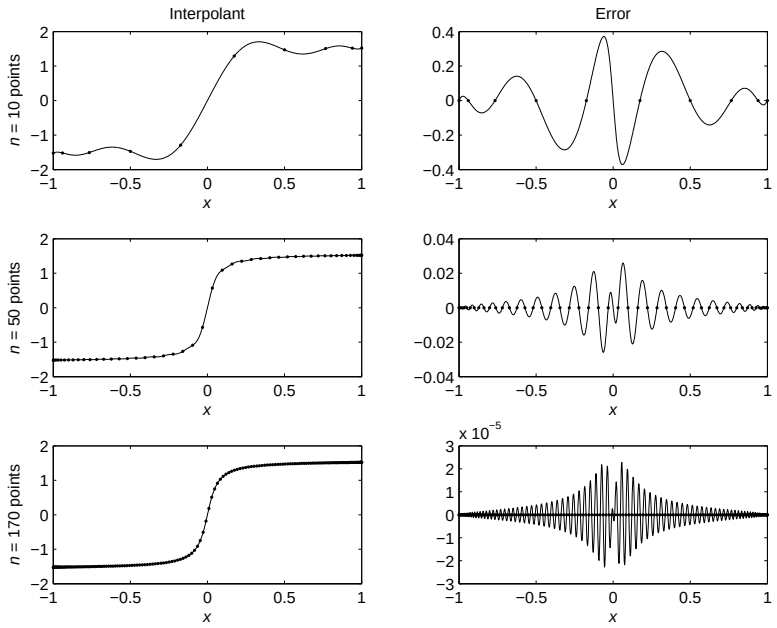


FIGURE 14.3.8. 10-point, 50-point and 170-point Chebyshev interpolants for $\arctan(20x)$ over [-1,1]; display of the interpolants and their errors.

if one also makes the shape parameter spatially variable (now taking the value ε_j for the RBF centered at the node $x_j \in [-1, 1]$). We note that it is favorable to let the ε_j-values be large where the nodes are dense, as will be discussed further in Section 14.7. Although the multivariate optimizer (function *ga* from Matlab's genetic algorithm toolbox) used in obtaining Figure 14.3.7 probably found only local optima (in contrast to the global ones), the error level that is reached is nevertheless spectacular in comparison with what can be achieved with, say, polynomial interpolation at the Chebyshev nodes (corresponding to a typical non-periodic PS method).

As Figure 14.3.8 shows, $n = 170$ nodes are needed to match the max norm accuracy of $2.5 \cdot 10^{-5}$ that RBF achieved using only $n = 10$ interpolation nodes. Hopefully, further numerical experiments in the style of Figure 14.3.7 will lead to fast and practical guidelines for effective choices for both nodes and shape parameter(s). The chances are excellent that especially the idea of spatially variable shape parameters will allow RBF interpolants to overcome the RP, as well as the Gibbs phenomenon as

discussed further in Section 14.7. Spatially variable shape parameters are discussed further in Section 14.7.

14.4. Stable computations in the flat RBF limit.

As we noted above, direct calculation of RBF interpolants by means of (14.4) and (14.3) lead to extreme ill-conditioning when $\varepsilon \to 0$. All the elements of the A-matrix (the coefficient matrix in (14.4)) then approach one. For example in 1-D, with n points, the condition number of the system will approach infinity at the rate $O(1/\varepsilon^n)$ if n is odd, and as $O(1/\varepsilon^{n-1})$ if n is even. High precision arithmetic (beyond the standard 64 bit) can be used to 'survive' some of the ill-conditioning, but the cost for this will increase without bound in the $\varepsilon \to 0$ limit. Preconditioning for the system (14.4) can be somewhat helpful [**?**], [**?**] but, once the coefficient matrix A has been formed numerically, irreversible conditioning damage has already been done. The Contour-Padé algorithm provided the first clear demonstration that direct use of (14.4), (14.3) merely amounts to an ill-conditioned numerical approach to a genuinely well-conditioned problem. More recently, a second algorithm, RBF-QR, has been developed which also entirely avoids the ill conditioning issue in the $\varepsilon \to 0$ limit. These two algorithms are briefly summarized next.

14.4.1. Contour-Padé method. In the discussion so far, the shape parameter ε has been a real-valued quantity. However, if one imagined solving (14.4) by Cramer's rule, followed by evaluating (14.3), it becomes clear that the resulting interpolant $s(\underline{x}, \varepsilon)$ (for fixed $\underline{x}$ and variable ε) is an analytic function of ε which, in the vicinity of the origin, has no other singularities than possibly some poles. Since $\lim_{\varepsilon \to 0} s(\underline{x}, \varepsilon)$ typically exists finite-valued [**?**], the origin ($\varepsilon = 0$) cannot even be a pole, and a removable singularity remains therefore as the only option. Furthermore, we know that the value of an analytic function at any point is the average value of the function taken around any circle centered at the point (assuming there are no singularities within the circles). In the simplest case, the Contour-Padé method will work as illustrated in Figure 14.4.1. With ill conditioning not allowing us to evaluate $s(\underline{x}, \varepsilon)$ directly at or near $\varepsilon = 0$, we simply evaluate it instead around a circle surrounding $\varepsilon = 0$, and then take the average.

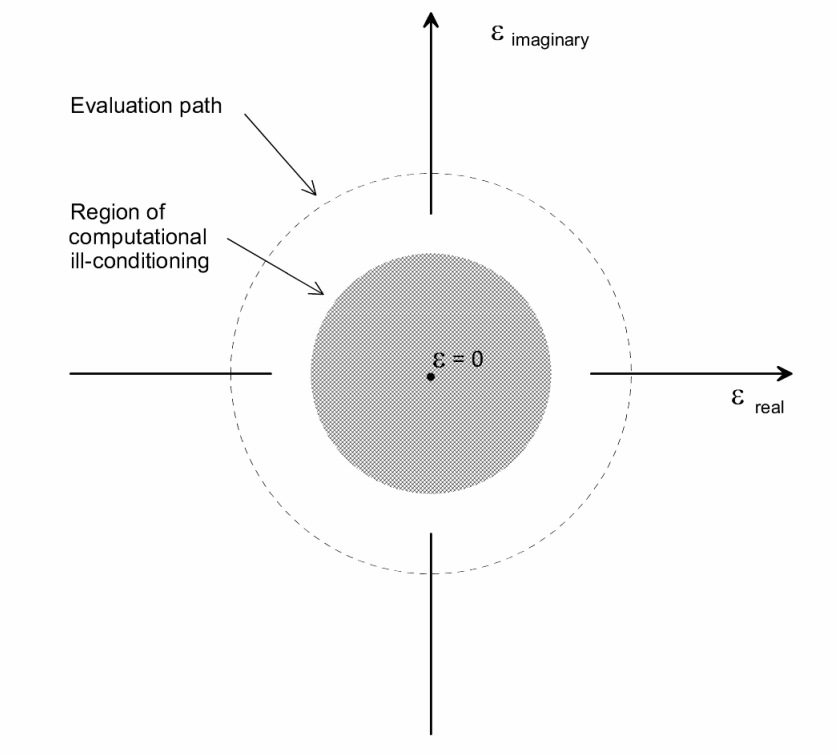


FIGURE 14.4.1. Schematic illustration of the Contour-Padé method in the simplest case of no singularities of $s(\underline{x}, \varepsilon)$ near the origin.

We know from the non-singularity theory in Section 14.2 that $s(\underline{x}, \varepsilon)$ cannot have any singularities along the real axis in the ε-plane. The only complication that might arise is that it sometimes turns out to have a few poles within the ill-conditioned central region. Instead of just averaging the values around the circle, the values are instead used as input to a complex FFT. This will produce (with exponentially increasing accuracy as the number of sample points are increased) the Laurent coefficients for an expansion of $s(\underline{x}, \varepsilon)$ that is valid in the largest singularity-free annulus that surround the contour. The terms with negative powers of ε originate purely from the poles inside the contour (whether located inside the ill-conditioned region, or outside it). Padé summation of these terms will convert them into a rational function $r(\varepsilon)$. Together with the remaining Laurent expansion terms (for

non-negative powers of ε), the interpolant takes the form

$$s(\underline{x}, \varepsilon) = \{r(\varepsilon)\} + \left\{d_0 + d_1\varepsilon + d_2\varepsilon^2 + d_3\varepsilon^3 + \ldots\right\},$$

which is well suited for numerical evaluation anywhere inside the evaluation path. Outside it, there are no ill-conditioning issues, and direct evaluation according to (14.4) and (14.3) work just fine. As n (the number of points in the data set) is increased, the number of poles in the vicinity of origin tends to remain very low. However, the central region of ill-conditioning grows, and the Contour-Padé method fails when this region becomes so large that it leaves no clear path between this region and where the branch points start along the imaginary axis (in case of MQ, arising from singularities of $\sqrt{1+(\varepsilon r)^2}$ in the ε-plane). In 2-D, this tend to happen when n is greater than around 100-200; the situation improves significantly if the number of dimensions increases. This causes no difficulties for the elliptic equation problem discussed in Section 14.3.1 or for generating scattered-node FD-type formulas as will be described in Section 14.6, but it is nevertheless desirable to have available a stable method which does not have this limitation on the number of points.

14.4.2. RBF-QR method. The key idea behind the RBF-QR method is to replace, in the case of small ε, the extremely ill conditioned RBF basis with a well conditioned one that spans exactly the same space, and to do this exchange in a way which does not at any stage involve any potentially dangerous numerical cancellations. The concept is somewhat reminiscent of how $\{1, x, x^2, ..., x^n\}$ forms a very ill conditioned basis over [-1,1], whereas the Chebyshev basis $\{T_0(x), T_1(x), T_2(x), ..., T_n(x)\}$ is a very well conditioned one. Since the spaces they span are identical, the results of interpolation using the two bases will also be identical, except for the fact that computations with the latter basis are vastly more stable against influence of rounding errors. The first implementation of RBF-QR was for the important special case of nodes on the surface of a sphere [**?**], followed by an implementation for general node distributions [**?**]. We limit the discussion here to the former case. The new equivalent bases that we introduce will be seen to converge to the spherical harmonics (SPH) basis as $\varepsilon \to 0$.

RBF	Definition	Expansion coefficients $c_{\mu,\varepsilon}$
MQ	$\sqrt{1+(\varepsilon r)^2}$	$\frac{-2\pi(2\varepsilon^2+1+(\mu+1/2)\sqrt{1+4\varepsilon^2})}{(\mu+3/2)(\mu+1/2)(\mu-1/2)}\left(\frac{2}{1+\sqrt{4\varepsilon^2+1}}\right)^{2\mu+1}$
IMQ	$\frac{1}{\sqrt{1+(\varepsilon r)^2}}$	$\frac{4\pi}{(\mu+1/2)}\left(\frac{2}{1+\sqrt{4\varepsilon^2+1}}\right)^{2\mu+1}$
IQ	$\frac{1}{1+(\varepsilon r)^2}$	$\frac{4\,\pi^{3/2}\,\mu!}{\Gamma(\mu+\frac{3}{2})(1+4\varepsilon^2)^{\mu+1}}\;{}_2F_1(\mu+1,\mu+1;2\mu+2;\frac{4\varepsilon^2}{1+4\varepsilon^2})$
GA	$e^{-(\varepsilon r)^2}$	$\frac{4\pi^{3/2}}{\varepsilon^{2\mu+1}}e^{-2\varepsilon^2}I_{\mu+1/2}(2\varepsilon^2)$

TABLE 2. SPH expansion coefficients corresponding to different choices of smooth RBFs.

14.4.2.1. *Relations between RBF and SPH.* A SPH expansion of a function defined over the surface of the unit sphere takes the form

$$s(x,y,z)=\sum_{\mu=0}^{\infty}\sum_{\nu=-\mu}^{\mu}c_{\mu,\nu}\;Y_\mu^{\;\nu}(\underline{x}).$$

Truncated SPH expansions ($\mu\le\mu_{\max}$) feature a completely uniform resolution over the surface of the sphere. Based on the works of Freeden et.al. [?] and of Hubbert and Baxter [?] we can find explicit formulas for the coefficients $c_{\mu,\varepsilon}$ when expanding a RBF centered on the surface of the sphere:

$$\phi(\|\underline{x}-\underline{x}_i\|)=\sum_{\mu=0}^{\infty}{\sum_{\nu=-\mu}^{\mu}}'\,\{c_{\mu,\varepsilon}\;\varepsilon^{2\mu}\;Y_\mu^{\;\nu}(\underline{x}_i)\}\;Y_\mu^{\;\nu}(\underline{x}). \tag{14.1}$$

(where the symbol $\sum'$ implies halving the $\nu=0$ term of the sum). The resulting coefficients in the cases of MQ, IMQ, IQ, and GA are shown in Table 2.

A key feature of these formulas is that, even for ε vanishingly small, all coefficients can be calculated without any danger of loosing significant digits.

14.4.2.2. *Matrix representation and QR-factorization.* The result when (14.1) is applied in turn to the n RBF, all centered on the surface of the sphere, can be

re-written in matrix×vector form:

$$\begin{bmatrix} \phi(\|\underline{x}-\underline{x}_1\|) \\ \phi(\|\underline{x}-\underline{x}_2\|) \\ \vdots \\ \phi(\|\underline{x}-\underline{x}_n\|) \end{bmatrix} =$$

$$= \begin{bmatrix} \frac{c_{0,\varepsilon}}{2}Y_0^0(\underline{x}_1) & \varepsilon^2 c_{1,\varepsilon}Y_1^{-1}(\underline{x}_1) & \varepsilon^2\frac{c_{1,\varepsilon}}{2}Y_1^0(\underline{x}_1) & \varepsilon^2 c_{1,\varepsilon}Y_1^1(\underline{x}_1) & \varepsilon^4\{.\} & . \\ \frac{c_{0,\varepsilon}}{2}Y_0^0(\underline{x}_2) & \varepsilon^2 c_{1,\varepsilon}Y_1^{-1}(\underline{x}_2) & \varepsilon^2\frac{c_{1,\varepsilon}}{2}Y_1^0(\underline{x}_2) & \varepsilon^2 c_{1,\varepsilon}Y_1^1(\underline{x}_2) & \varepsilon^4\{.\} & . \\ \dots & \dots & \dots & \dots & \dots & . \\ \frac{c_{0,\varepsilon}}{2}Y_0^0(\underline{x}_n) & \varepsilon^2 c_{1,\varepsilon}Y_1^{-1}(\underline{x}_n) & \varepsilon^2\frac{c_{1,\varepsilon}}{2}Y_1^0(\underline{x}_n) & \varepsilon^2 c_{1,\varepsilon}Y_1^1(\underline{x}_n) & \varepsilon^4\{.\} & . \end{bmatrix} \begin{bmatrix} Y_0^0(\underline{x}) \\ Y_1^{-1}(\underline{x}) \\ Y_1^0(\underline{x}) \\ Y_1^1(\underline{x}) \\ Y_2^{-2}(\underline{x}) \\ Y_2^{-1}(\underline{x}) \\ Y_2^0(\underline{x}) \\ Y_2^1(\underline{x}) \\ Y_2^2(\underline{x}) \\ \dots \end{bmatrix}$$

$$= B \cdot Y$$

A QR factorization of B combines its rows in such a way that the result becomes upper triangular. Since the powers of ε are the same within each column, elements with different powers of ε do not mix in this process, and the upper triangular matrix is obtained stably, featuring the same pattern as for B in terms of the powers of ε. Factoring out to the left the lowest power of ε for each row of R gives the result

$$\begin{bmatrix} \phi(\|\underline{x}-\underline{x}_1\|) \\ \phi(\|\underline{x}-\underline{x}_2\|) \\ \phi(\|\underline{x}-\underline{x}_3\|) \\ \vdots \\ \phi(\|\underline{x}-\underline{x}_n\|) \end{bmatrix} = \begin{bmatrix} & & \\ & Q & \\ & & \end{bmatrix} \begin{bmatrix} 1 & & & & \\ & \varepsilon^2 & & & \\ & & \varepsilon^2 & & \\ & & & \varepsilon^2 & \\ & & & & \ddots \end{bmatrix} \times$$

$$\times \begin{bmatrix} * & . & . & . & . & . & . & . & \dots \\ & * & * & * & . & . & . & . & \dots \\ & & * & * & . & . & . & . & \dots \\ & & & * & . & . & . & . & \dots \\ & & & & * & * & * & * & \dots \\ & & & & & \ddots & \dots & \dots & \dots \\ & & & & & & & & \end{bmatrix} \begin{bmatrix} Y_0^0(\underline{x}) \\ Y_1^{-1}(\underline{x}) \\ Y_1^0(\underline{x}) \\ Y_1^1(\underline{x}) \\ Y_2^{-2}(\underline{x}) \\ Y_2^{-1}(\underline{x}) \\ Y_2^0(\underline{x}) \\ Y_2^1(\underline{x}) \\ Y_2^2(\underline{x}) \\ \vdots \end{bmatrix} \tag{14.2}$$

$$= (Q \cdot E \cdot R) \cdot \underline{Y}(\underline{x}) \tag{14.3}$$

where Q is a unitary $n \times n$ matrix, E is a $n \times n$ diagonal matrix and R is an upper triangular $n \times m$ matrix. The entries marked as "*" in the matrix R are of size $O(\varepsilon^0)$. All the other non-zero entries of R are of size $O(\varepsilon^2)$ (or higher powers). The new basis is given by the elements of $R \cdot \underline{Y}(\underline{x})$. It differs from the original basis only by having omitted the non-singular matrix $Q \cdot E$ from its left side, i.e. it forms a different basis for the same space. The key feature of the algorithm is that the E-matrix, which contains all the ill-conditioning, disappeared *analytically* from the problem, and has not in any way damaged the accuracy of the equivalent (but well-conditioned) base $R \cdot \underline{Y}(\underline{x})$. We can easily continue one step further and combine the new basis functions so that each one becomes a single spherical harmonic function, with a small perturbation. These perturbations fade away as $\varepsilon \to 0$ because all the entries denoted with "$\cdot$" contain a positive power of ε. RBF interpolation on the sphere in the $\varepsilon \to 0$ limit will thus agree with SPH interpolation. However, as we have seen already RBF are often more accurate for finite (or spatially variable) ε-values, thereby providing a new perspective on SPH-based methods.

14.5. Brief overview of high order FD methods and PS methods.

14.5.1. High order FD methods. On an equispaced grid with nodes located at $\{\dots -2h,\ -h,\ \ 0,\ \ h,\ \ 2h,\ \ \dots\}$, the simplest approximation for $f\,'(0)$ is Check FD sectio

$$f\,'(0) \approx -\frac{1}{2h}f(-h) + 0\ f(0) + \frac{1}{2h}f(h) \tag{14.1}$$

with an error of size $O(h^2)$. The coefficients in formulas such as this are uniquely determined by requiring that they be exact for polynomials of as high degree as possible. Numerous methods are available to very effectively compute coefficients also for formulas of higher orders of accuracy and which approximate higher derivatives, both on equispaced and nonuniform 1-D grids ([**?**], [..], [..]).

14.5.2. FD introduction to PS methods.

14.5.2.1. *Periodic case.* With any of the procedures to compute FD weights just mentioned, one readily determines the coefficients that are shown in Table 3. In this special case, they are also available in closed form

$$c_{j,k} = \begin{cases} \frac{(-1)^{k+1}(j!)^2}{i\ (j+k)!\ (j-k)!} & k = \pm 1, 2, \ldots, \pm j \\ 0 & k = 0 \end{cases}$$

for approximations of order $2j,\ j = 1, 2, \ldots$.

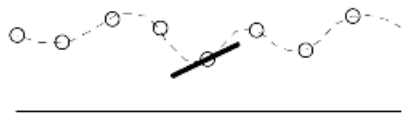
order											
2					$-\frac{1}{2}$	0	$\frac{1}{2}$				
4				$\frac{1}{12}$	$-\frac{2}{3}$	0	$\frac{2}{3}$	$-\frac{1}{12}$			
6			$-\frac{1}{60}$	$\frac{3}{20}$	$-\frac{3}{4}$	0	$\frac{3}{4}$	$-\frac{3}{20}$	$\frac{1}{60}$		
8		$\frac{1}{280}$	$-\frac{4}{105}$	$\frac{1}{5}$	$-\frac{4}{5}$	0	$\frac{4}{5}$	$-\frac{1}{5}$	$\frac{4}{105}$	$-\frac{1}{280}$	
⋮		↓	↓	↓	↓	⋮	↓	↓	↓	↓	
limit	$\cdots$	$\frac{1}{4}$	$-\frac{1}{3}$	$\frac{1}{2}$	-1	0	1	$-\frac{1}{2}$	$\frac{1}{3}$	$-\frac{1}{4}$	$\cdots$

TABLE 3. Weights for centered FD approximations of the first derivative on an equispaced grid (omitting the factor 1/h).

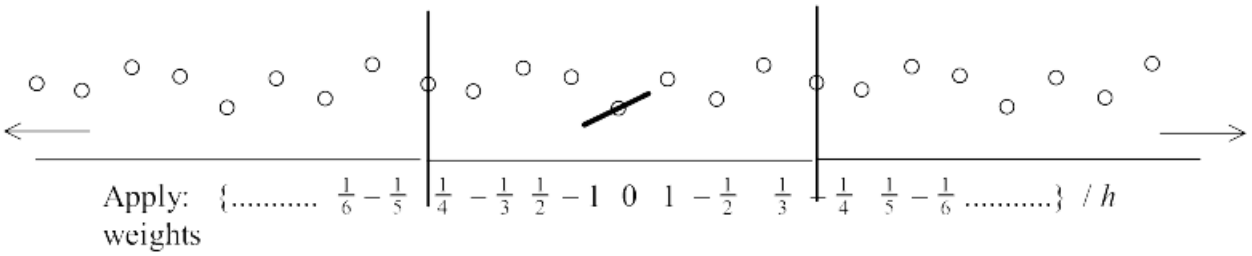
We can note that there exists a very simple limit for the coefficients when the order of the FD stencils go to infinity

$$\lim_{j\to\infty} c_{j,k} = \begin{cases} (-1)^{k+1}\ /\ k & k = \pm 1, 2, \ldots \\ 0 & k = 0 \end{cases} .$$

This limit provides a fundamental connection between FD and Fourier PS methods in the way that is illustrated in Figure 14.5.1.



a. Derivative calculated by analytic differentiation of interpolating trigonometric polynomial (for example obtained by FFT)



b. Derivative calculated by applying limiting FD stencil to periodic repetitions of the data.

FIGURE 14.5.1. Two different ways to wiev the approximation of a first derivative by the Fourier PS method. Both give the same result.

Collocation of equispaced periodic data with trigonometric functions and then differentiating these functions analytically gives exactly the same derivative approximations as one would get if the periodic data was extended to an infinite grid and the infinite order FD stencil was applied. Needless to say, the latter is a computationally less practical approach, but the equivalence is fundamental in connecting trigonometric function collocation with increasing order FD methods. The equivalence can be proved to hold for any order derivative [**?**], [**?**].

14.5.2.2. *Non-periodic case.* If we consider non-periodic data, for example with data points located at the Chebyshev nodes $x_k = -\cos\frac{(k-1)\pi}{n-1}$, $k = 1, 2, \ldots, n$, we have a very similar equivalence. For any node distribution, we can approximate the derivative at any node point (or at any in-between location) by easily obtained FD weights in stencils that extend across the full grid [**?**], [Weideman]. The resulting derivative approximations will be exactly the same as if the data was collocated (i.e. fitted by a linear combination of Chebyshev polynomials) and the resulting function

maybe clarify fu
ther with som
picture of Fouri
analysis of F
schemes

differentiated analytically. The FD approach reduces to the Chebyshev PS method if the node points happen to be located at Chebyshev node locations (14.3), but works just as well however the points were distributed—i.e. there is no need to limit oneself to node distributions that are suggested by any classical polynomial set.

The collocation methods we have just described - based on Fourier and Chebyshev basis functions - generalize in a completely straightforward manner to tensor-type grids (i.e. rectangular domains in 2-D, etc.). By use of domain decomposition ideas, single-domain PS methods lead to spectral element methods, gaining some level of geometric flexibility [??]. As we will see next, PS methods of the types described so far (using any fixed set of basis functions) can never allow the data to lie at scattered locations. One key theme of this chapter is how the RBF approach entirely overcomes this restriction.

The efficiency of PS methods, periodic (Fourier) or non-periodic is thoroughly documented in the literature. check

14.5.3. Possible singularity of scattered node PS collocation in more than 1-D.. To show that no fixed set of expansion functions can ever guarantee a non-singular interpolation problem in more that 1-D, we follow [**?**] and consider a set of data points $\underline{x}_k,\ k = 1, 2, \ldots, n$ in more than 1-D, with associated function values f_k. Let $\psi_k(\underline{x})$ be any fixed set of basis functions. The interpolant then takes the form

$$s(\underline{x}) = \sum_{k=1}^{n} \lambda_k \psi_k(\underline{x})$$

where the expansion coefficients are obtained by solving

$$\begin{bmatrix} \psi_1(\underline{x}_1) & \psi_2(\underline{x}_1) & \cdots & \psi_n(\underline{x}_1) \\ \psi_1(\underline{x}_2) & \psi_2(\underline{x}_2) & & \psi_n(\underline{x}_2) \\ \vdots & & & \vdots \\ \psi_1(\underline{x}_n) & \psi_2(\underline{x}_n) & \cdots & \psi_n(\underline{x}_n) \end{bmatrix} \begin{bmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_n \end{bmatrix} = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \end{bmatrix}.$$

In more than 1-D, it is possible to continuously move two data locations so they become interchanged, without them having had to coincide at any time along the way. Doing this, two rows of the coefficient matrix above have swapped positions,

i.e. the determinant has changed sign. Therefore, it must have been zero somewhere along that way.

In contrast, this particular exchange process will not lead to any difficulty in the case of RBF interpolation. Considering the corresponding coefficient matrix in (14.4), both a row and a column will have changed places, i.e. there is no resulting sign change. To show that no other movement of nodes can lead to singularities require the more general arguments given in Section 14.2.

14.6. RBF-generated finite differences.

As we noted in Section 14.5, weights in 1-D FD formulas are typically obtained by requiring them to be exact for polynomials of as high degree as possible. The same idea has been tried for scattered nodes in 2-D, with some success [?], [?], [?]. As an alternative to basing FD-type formulas on local polynomial interpolants, one can instead base them on local RBF interpolants. To illustrate this concept, as developed in [?], we first recall two classical FD stencils for the 2-D Laplacian $\Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$ [?], [?], [?]

Explicit (2nd order) *Compact* (4th order)

$$\begin{bmatrix} & 1 & \\ 1 & -4 & 1 \\ & 1 & \end{bmatrix} \frac{u}{h^2} = \Delta u \quad ; \quad \begin{bmatrix} 1 & 4 & 1 \\ 4 & -20 & 4 \\ 1 & 4 & 1 \end{bmatrix} \frac{u}{6h^2} = \begin{bmatrix} & 1 & \\ 1 & 8 & 1 \\ & 1 & \end{bmatrix} \frac{\Delta u}{12} \tag{14.1}$$

If one for example wants to solve Poisson's equation $\Delta u = f$, it is no disadvantage that a linear combination of Δu-values appear in the right hand side of the compact formula, as these would correspond to known f-values. The advantage is that we have reached 4^{th} order of accuracy, with the stencil for u still remaining small and diagonally dominant.

Figure 14.6.1 illustrates a scattered set of nodes, and how some of these have been selected out to form a similar stencil. To find weights $c_j,\ j = 1, \ldots, n_1$ and $b_k,\ k = \{\text{some subset of } 1, \ldots, n_1\}$, giving a formula accurate at location x_1, we demand it to be exact for the radial functions $\phi(\|\underline{x} - \underline{x}_j\|)$ and for $\Delta\phi(\|\underline{x} - \underline{x}_j\|),\ j =$

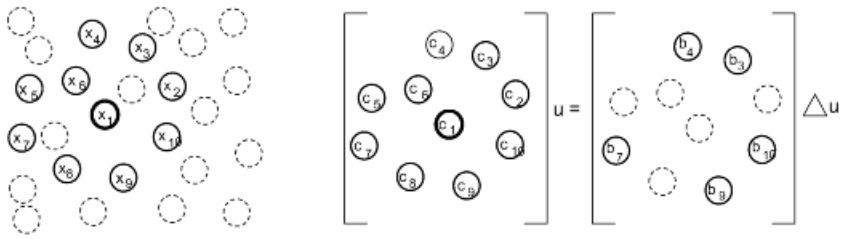


FIGURE 14.6.1. RBF-HFD concept: To the left, an example of how stencil points may be selected near a location at which the Laplacian is to be approximated. To the right, a schematic representation of how u-values and Δu-values (on a smaller subset still of the stencil points) will be linked through RBF-computed wights.

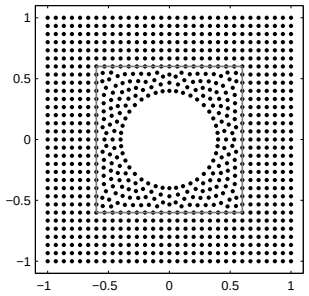


FIGURE 14.6.2. Computational domain in RBF-HFD test example; shown together with the hybrid node layout used in the example.

$1, \ldots, \{check\}$. Several issues relating to an efficient implementation are addressed in [**?**]. Key steps and issues include

- Form of the RBF interpolant (use of (14.9) turns out to be preferable to (14.3)),
- Given a center location x_1, select nearby nodes so the stencils with c_j - weights becomes diagonally dominant,
- Decide what radial function and shape parameter ε to use

We will here limit ourselves to give an example from [**?**] which illustrates how the concept may be used effectively in a case with irregular geometry:

Example: Numerically solve the nonlinear Poisson equation $\Delta u = e^{-2x} u^3$ over the domain between the large outer square and the inner circle as shown in Figure 14.6.2. Dirichlet boundary conditions are chosen so that $u(x, y) = e^x \tanh \frac{y}{\sqrt{2}}$ becomes an exact solution.

The hybrid node structure uses a regular grid with 4th order compact FD stencils over a large part of the domain, and then adds scattered-node stencils only as needed to accommodate irregular boundaries. In this example, diagonal dominance and 4^{th} order accuracy could be achieved everywhere by using 9 c-weights with 5 b-weights in the regular part (like in (14.1)) and 10 c-weights with 9 b-weights where nodes are irregular. Standard Newton-SOR iterations ([**?**], Section 7.4) converge very rapidly. The resulting error varies with ε as is shown in Figure 14.6.3. For low ε, the FD

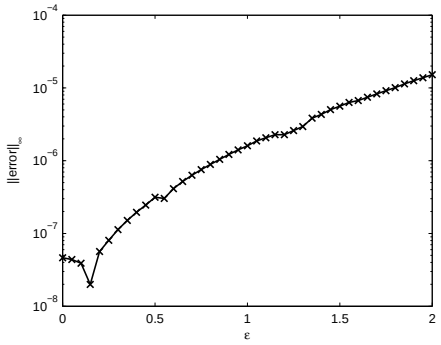


FIGURE 14.6.3. Error in RBF-HFD solution to nonlinear Poisson test problem; shown as function of ε.

weights were computed using the Contour-Padé method. In the $\varepsilon \to 0$ limit, they typically agreed with standard polynomial-based weights in cases of regular meshes.

14.7. Some other related RBF topics.

We have in this section collected a number of additional RBF topics that all are of importance when designing and analyzing RBF for PDEs.

14.7.1. Spatially variable shape parameters. We have already come across demonstrations of spatially variable shape parameters in Section 14.3, Figure 14.3.7). Many aspects of this topic are discussed in [**?**]. Following a literature survey, we will here focus on some surprisingly distinct patterns that arise in the eigenvalues of the A-matrix, as defined in (14.4), both when ε is the same at all nodes (denoted "ε constant") and when it varied from node point to node point (denoted "ε_k variable"). Although much analysis remains to be done (in particular with respect to conditions for non-singularity; the theorems in Section 14.2 assume the "ε constant" case), our observations nevertheless strongly indicate that spatially variable shape parameters offer major opportunities for improving both accuracy and conditioning of RBF interpolation.

14.7.1.1. *Some literature on choosing a good shape parameter values.* The literature on selecting a good (single) value for ε is extensive, as noted in the introduction to Section 14.3. The idea of using a spatially variable shape parameter in the RBF

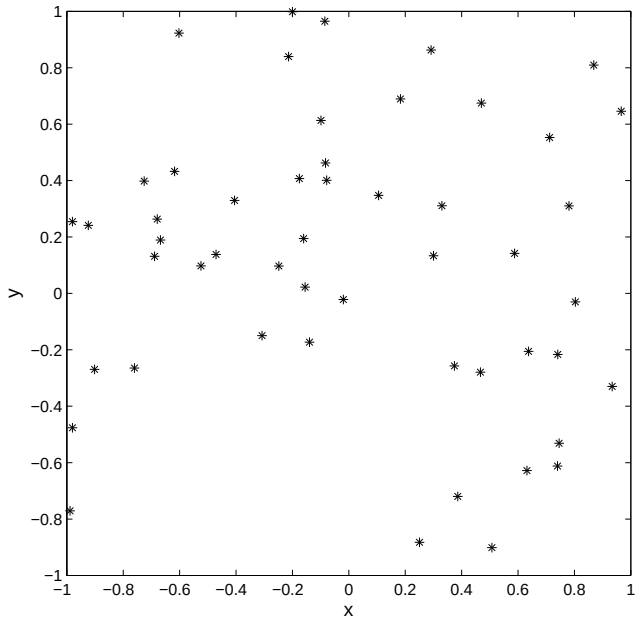


FIGURE 14.7.1. Random distribution of $n = 51$ nodes used in the eigenvalue calculations.

expansion (14.3) has been proposed numerous times. A limited version of the concept was proposed by Kansa already in 1990 [**?**]. The idea was generalized shortly afterwards by him and Carlson [**?**], using least squares optimization to find good ε_k distributions for certain test functions. More recently, spatially variable ε_k MQ interpolants in 1-D have been related to 1-D splines [**?**]. In [**?**], an adaptive algorithm is proposed in which node densities are varied according to a local error criterion, and variable ε_k-values are increased wherever the node layout has become denser. Numerical experiments reported in [**?**] led to a number of observations, several of which agree well with results in this study (such as the benefit of reducing ε_k at boundaries and that introducing oscillations in ε_k might improve both conditioning and accuracy).

14.7.1.2. *Eigenvalue analysis for the A-matrix.* Results in [**?**] (page 308) can be shown to imply that the eigenvalues of the A-matrix in the former case will scale with different powers of ε. Numerical calculations provide a much more detailed picture.

For example, with $n = 51$ scattered nodes in 2-D, as shown in Figure 14.7.1,

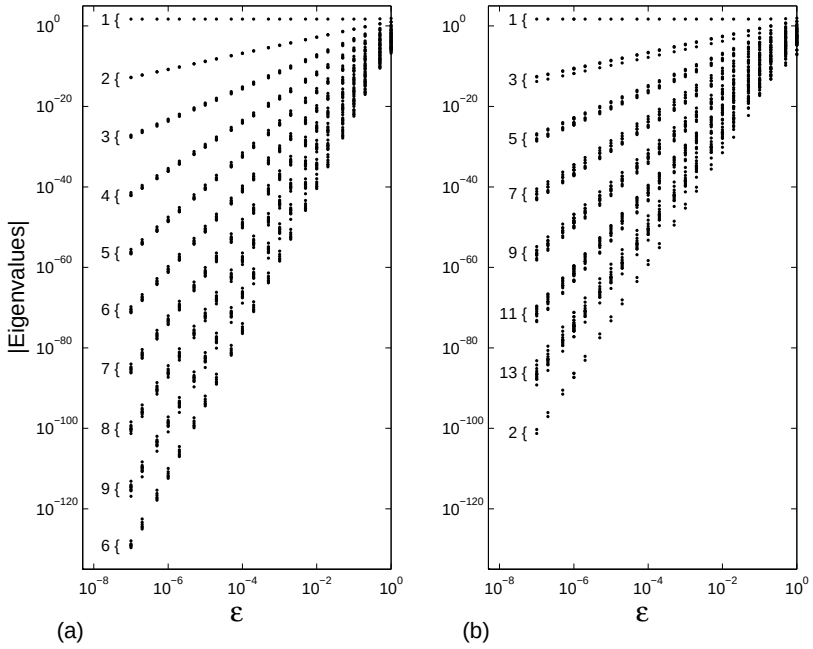


FIGURE 14.7.2. Eigenvalues of the MQ RBF A-matrix in the 2-D $n = 51$ scattered node case, as functions of ε.The number of eigenvalues in each of the different groups are also shown (easiest counted when shown numerically rather than graphically); (a) ε constant, (b) ε_k variable.

the eigenvalues vary with ε as seen in Figure 14.7.2 a (computed using Matlab's VPA - variable precision arithmetic). Irrespective of the choice of RBF type (IQ, MQ, or GA), the eigenvalues form very clear groups, following the pattern

$$\{O(1)\},\ \{O(\varepsilon^2), O(\varepsilon^2)\},\ \{O(\varepsilon^4), O(\varepsilon^4), O(\varepsilon^4)\},\ \{O(\varepsilon^6), O(\varepsilon^6), O(\varepsilon^6), O(\varepsilon^6)\},\ \ldots$$

until the last eigenvalue is reached (causing the last group to possibly contain fewer eigenvalues than the general pattern would suggest). Different choices of scattered node locations $\underline{x}_k$ make no difference in this regard. More concisely, we can write this eigenvalue pattern as

$$1, 2, 3, 4, 5, 6, \ldots \tag{14.1}$$

indicating how many eigenvalues there are of orders ε^0, ε^2, ε^4, ε^6, ε^8, ε^{10}, etc. Given such a pattern, one can immediately calculate the orders of both cond(A) and

		Power of ε =								
Geometry	shape param.	0	2	4	6	8	10	12	14	...
1-D non-periodic	ε constant	1	1	1	1	1	1	1	1	...
	ε_k variable	1	2	2	2	2	2	2	2	...
1-D on circle periph.	ε constant	1	2	2	2	2	2	2	2	...
(embedded in 2-D)	ε_k variable	1	2	2	2	2	2	2	2	...
2-D non-periodic	ε constant	1	2	3	4	5	6	7	8	...
	ε_k variable	1	3	5	7	9	11	13	15	...
On spherical surface	ε constant	1	3	5	7	9	11	13	15	...
(embedded in 3-D)	ε_k variable	1	3	5	7	9	11	13	15	...
3-D non-periodic	ε constant	1	3	6	10	15	21	28	36	...
	ε_k variable	1	4	9	16	25	36	49	64	...

TABLE 4. Numbers of eigenvalues of different sizes (powers of ε) for different geometries and types of shape parameter.

$\det(A) = \prod_{k=1}^{n} \lambda_k$. Doing so confirms the special case noted in Section 14.4 of $\det(A)$ being of size $O(\varepsilon^{416})$ when $n = 41$ (obtained in [**?**] by an entirely different approach involving contour integration) and also shows that in this same case, $\text{cond}(A) = O(\varepsilon^{-16})$. Corresponding results for different geometry types are shown in Table 4 on the lines labeled "ε constant".

Figure 14.7.2 b shows that choosing $\varepsilon_k = \varepsilon \cdot \{\text{random numbers on [0,1]}\}$ and letting $\varepsilon \to 0$ (for the figure using same random nodes in 2-D as seen in Figure 14.7.1) creates a different but equally distinct and clear eigenvalue pattern

$$1, 3, 5, 7, 9, 11, \ldots \tag{14.2}$$

In the $n = 41$ -case discussed above, we get for the spatially variable ε_k $\det(A) = O(\varepsilon^{310})$ and $\text{cond}(A) = O(\varepsilon^{-12})$ (i.e. a clear improvement). These same types of numerical studies can easily be extended to all the cases shown in Table 4 as "ε_k variable"

. In all cases that are shown, the results are verified for IQ, MQ, and GA in calculations extending to still higher values of n and also for numerous cases of different scattered node sets and random ε_k distributions.

		Number of nodes n =						
Geometry	shape param.	1	10	100	1000	10000	100000	...
1-D non-periodic	ε constant	0	18	198	1998	19998	199998	...
	ε_k variable	0	10	100	1000	10000	100000	...
1-D on circle periph.	ε constant	0	10	100	1000	10000	100000	...
(embedded in 2-D)	ε_k variable	0	10	100	1000	10000	100000	...
2-D non-periodic	ε constant	0	6	26	88	280	892	...
	ε_k variable	0	6	18	62	198	632	...
On spherical surface	ε constant	0	6	18	62	198	632	...
(embedded in 3-D)	ε_k variable	0	6	18	62	198	632	...
3-D non-periodic	ε constant	0	4	14	34	76	166	...
	ε_k variable	0	4	12	26	60	132	...

TABLE 5. Condition number $\text{cond}(A) = 1 \,/\, \varepsilon^{\alpha(n)}$ with $\alpha(n)$ displayed for various values of n in all the cases of Table 4.

The patterns seen in Table 4 show that "ε_k variable" in all the non-periodic cases is more favorable than "ε constant". Because $\text{cond}(A) = O(1/\{\text{smallest eigenvalue}\})$, we can readily convert the information in Table 4 to obtain $\text{cond}(A)$ as a function of n, as shown in some typical cases in Table 5

. For fixed n, conditioning is also seen to improve rapidly with increasing number of dimensions.

The data in Table 4 shows that, even with randomly scattered ε_k-values (or when the ε_k-values are chosen according to an 'inversely proportional to nearest neighbor' strategy; found to give exactly the same eigenvalue results), extremely distinct eigenvalue patterns hold. One might have expected that completely irregular variations in the shape parameters ε_k might have led to irregular variations in the eigenvalues of the A-matrix (compared to the constant ε situation), and that therefore some of the extremely small eigenvalues might have been perturbed enough to change sign (with the possibility of becoming zero). The fact that even the very smallest eigenvalues show no tendency whatsoever towards any irregularities suggests that singular systems are not likely to arise.

Still other eigenvalue patterns appear in 'intermediate' cases, such as all ε_k but one taking the same value, or the $\varepsilon_k$alternating between two values. For example, in the case "2-D general" (cf. Table 4), the patterns become as seen in Table 6. These

		Power of $\varepsilon =$										
Geometry	shape param.	0	2	4	6	8	10	12	14	16	18	...
2-D general	one ε_k different	1	3	2	5	4	7	6	9	8	11	...
	ε_k alternating	1	3	4	5	7	8	9	11	12	13	...

TABLE 6. Eigenvalue patterns (for IQ, MQ and GA) in two additional cases of scattered points in 2-D.

two cases are seen to feature conditionings that fall between the most favorable "ε_k variable" and least favorable "ε constant" cases shown in Table 4.

.

14.7.2. Analysis of RBF on lattices.

some results fro

[?]

14.7.2.1. *Accuracy of interpolation and derivative approximations.*

14.7.2.2. *Cardinal coefficient locality.* Many types of RBF (such as MN, TPS, MQ, etc.) are large across an entire domain. Yet, in contrast with expansions in orthogonal polynomials, RBF expansions exhibit strong locality with regard to their coefficients. That is, changing a single data value mainly affects coefficients of RBF that are centered in the immediate vicinity of that data location. This locality feature is advantageous for the development of fast and well conditioned iterative RBF algorithms. Although locality holds for scattered data in any number of dimensions, it has so far been analyzed successfully only on periodic lattices, and mainly in 1-D. Following [?], we can for any radial functions introduce a 2π-periodic function

$$\Xi(\xi) = \sum_{k=-\infty}^{\infty} \phi(k)\, e^{i\,k\,\xi} = \sum_{j=-\infty}^{\infty} \widehat{\phi}(\xi + 2\pi j). \tag{14.3}$$

The second sum above, following from Poisson's summation formula, will typically converge also in the cases where the first one diverges. With cardinal data, defined at the integer lattice points as $f_0 = 1$ and $f_k = 0$ at $x = k$ non-zero integer, the RBF expansion coefficients become [?], [?]

$$\lambda_k = \frac{1}{2\pi} \int_0^{2\pi} \frac{\cos k\xi}{\Xi(\xi)}\, d\xi \tag{14.4}$$

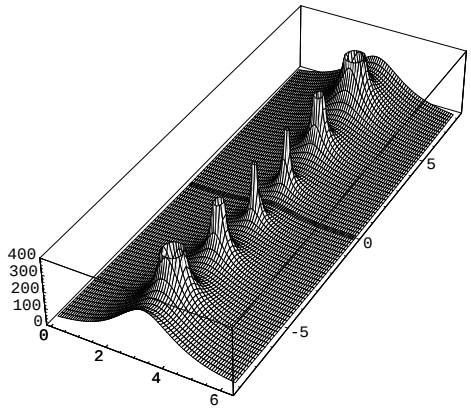


FIGURE 14.7.3. Magnitude of $h(\xi)$, as given by (14.6) over the domain $0 \leq \mathrm{Re}\xi \leq 2\pi$, $-8 \leq \mathrm{Im}\xi \leq 8$.

and the RBF cardinal interpolant becomes

$$s_C(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \frac{\widehat{\phi}(\xi)\ \cos x\xi}{\Xi(\xi)}\, d\xi. \tag{14.5}$$

By means of Cauchy's theorem and the calculus of residues, one can obtain very accurate approximations for the integral in (14.4). The case of MQ with $\varepsilon = 1$ is used below to illustrate this approach. The task then becomes to evaluate

$$\lambda_k = -\frac{1}{4\pi} \int_0^{2\pi} h(\xi)\, e^{i\,k\,\xi} d\xi$$

where

$$h(\xi) = \frac{1}{\sum_{j=-\infty}^{\infty} \frac{K_1(|2\pi j+\xi|)}{|2\pi j+\xi|}}\ .$$

The function $h(\xi)$ is 2π-periodic, and can over $\xi \in [0, 2\pi]$ (on the real axis) be written, without taking magnitudes, as

$$h(\xi) = \frac{1}{\sum_{j=0}^{\infty} \frac{K_1(2\pi j+\xi)}{2\pi j+\xi}\ +\ \sum_{j=1}^{\infty} \frac{K_1(2\pi j-\xi)}{2\pi j-\xi}}. \tag{14.6}$$

In this latter form, $h(\xi)$ can be extended as a single-valued analytic function throughout the strip $0 \leq \mathrm{Re}\xi \leq 2\pi$, $-\infty < \mathrm{Im}\xi < \infty$.

Figure 14.7.3 illustrates the magnitude of this function, and Figure 14.7.4 shows its schematic character. Figure needs r pair.

FIGURE 14.7.4. Character of the function $h(\xi)$ in the complex plane. The original and the modified integration paths are shown.

We change the integration path as is indicated in Figure 14.7.4, and note that the two leading contributions to the integral, when k increases, will come from (i) the first pole and (ii) from the (non-canceling) contributions from the vicinities of the branch points at $\xi = 0$ and $\xi = 2\pi$. Along the line $\xi = \pi + i\ t$, t real, the function $h(\xi)$ is purely real and $1/h(\xi)$ features decaying oscillations. The first pole of $h(\xi)$ appears near $\pi + 1.056109\ i$ and has a residue of approximately -34.866, thus contributing a term of $17.433\ (-1)^k\ e^{-1.056\ k}$ to λ_k. The singularity of $h(\xi)$ around the origin (repeated at $\xi = 2\pi$) comes from only one term in the denominator of (14.6), taking the form $\frac{\xi}{K_1(\xi)} = \xi^2 + (\frac{1}{4} - \frac{\gamma}{2} + \frac{\ln 2}{2} - \frac{\ln \xi}{2})\xi^4 + \ldots$ The branch singularity is to leading order of the form $-\frac{1}{2}\xi^4 \ln \xi = -\frac{1}{2}\xi^4(\ln|\xi| + i\ \arg \xi)$ (and similarly around $\xi = 2\pi$). What does not cancel between the two sides of the contour but instead adds up (hence the factor 2 below) amounts to $2(-\ \frac{1}{4\pi}) \int_0^{i \cdot \{\text{some } \delta > 0\}} (-\ \frac{1}{2})\xi^4 i\ \frac{\pi}{2} e^{-ik\xi}\, d\xi$. Letting $\xi = it$ and noting that, as $k \to \infty$, we can change the upper integration limit to infinity, this simplifies to $-\ \frac{1}{8} \int_0^\infty t^4 e^{-kt} dt = -\ \frac{3}{k^5}$. For increasing k, we thus obtain

$$\lambda_k \approx \underbrace{17.433\ (-1)^k\ e^{-1.056\ k} + \ldots}_{\text{exponential part}}\ \underbrace{-\ \frac{3}{k^5} + \ldots}_{\text{algebraic part}} \tag{14.7}$$

Figures 14.7.5a,b compare, using log-linear and log-log scales respectively, the true values for $|\lambda_k|$ against the 2-term approximation in (14.7).

The agreement is seen to be nearly perfect (when considering that only the first term of the exponential part and of the algebraic part were retained).

The same procedure as above can be carried through for any value of the shape parameter ε and also for all other RBF types. There will in every case be an exponential decay process, featuring oscillations in sign. It will depend on the regularity of $\widehat{\phi}(\xi)$ at $\xi = 0$ (if this is a branch point or not when continued to complex ξ) whether there will also be an algebraic non-oscillatory decay present. Given the expressions for $\widehat{\phi}(\xi)$ in Table 1, we can see that λ_k will decay exponentially for all k in the cases of MN (confirming what we obtained earlier when we considered MN splines

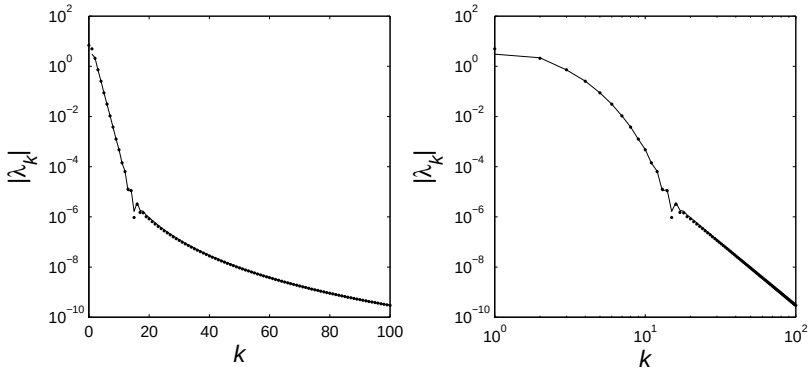


FIGURE 14.7.5. Comparison between correct values of $|\lambda_k|$ for MQ in 1-D, $\varepsilon = 1$ (dots) and the 2-term asymptotic formula (14.7) (solid line). The subplot to the left is log-linear and the one to the right of type log-log.

in Section 14.2), GA and SH, whereas there will be a transition from exponential to algebraic decay in the cases of TPS, MQ, IQ and IMQ. Formulas corresponding to (14.7) for large numbers of RBF cases (and for different ε) as well as some generalizations to 2-D lattices can be found in [?].

14.7.2.3. *RBF and the Gibbs phenomenon.* The best known version of the Gibbs phenomenon is the overshoot that arises when a discontinuous function is represented by a truncated set of Fourier expansion terms. A similar situation arises if equispaced data is interpolated for example by trigonometric functions or splines.

Figures 14.7.6(a)—(c) show more detailed pictures near a unit height jump in these three cases. Exact formulas for the overshoots are available in these and other cases, see for example [?], Section 2.4, also [?], [?] and [?]. The amplitudes of successive oscillations decay in inverse proportion with the distance from the jump in the first two cases, but exponentially fast in case of splines. Following [?], we will next see that RBF interpolants can feature a variety of decay patterns.

If we, in place of cardinal data (as in Section 14.7.2.2) consider step (Gibbs-) data ($f_k = 1$ at $x = k$ non-positive integer and $f_k = 0$ at $x = k$ positive integer), adding translates of (14.5) gives the Gibbs interpolant as

$$s_G(x) = \sum_{j=0}^{\infty} s_C(x + j). \tag{14.8}$$

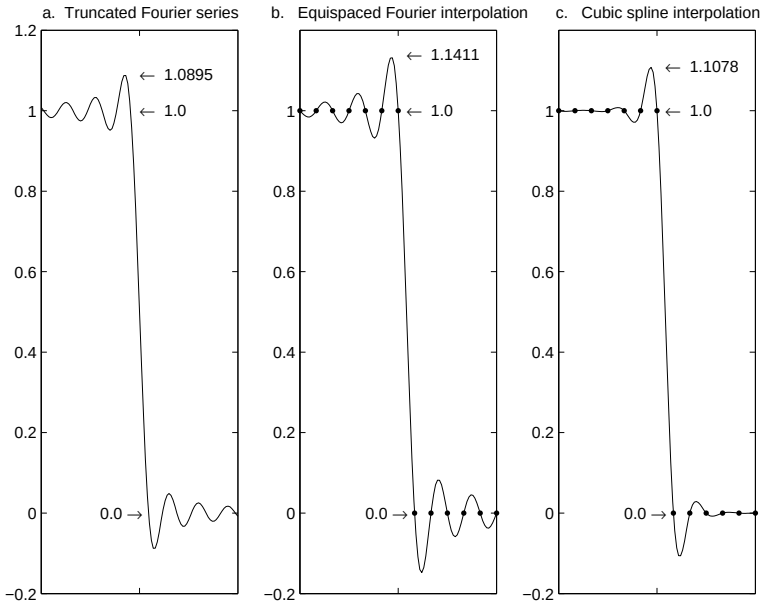


FIGURE 14.7.6. The Gibbs phenomenon for (a) truncated Fourier series, (b) equispaced Fourier interpolation, and (c) cubic spline interpolation. For (b) and (c), the nodes are located at the integers, with function value zero at positive integers, else one.

The similarity between the integrals (14.4) and (14.5) will permit the key observations for λ_k to be carried over, first to $s_C(x)$ and then by (14.8) to $s_G(x)$. The integrals differ only in a few respects:

- A trivial multiplicative factor,
- The free parameter is called x instead of k (and we will consider it also for non-integer values),
- There is an extra factor $\widehat{\phi}(\xi)$ in the numerator of (14.5),
- The integration interval is $[-\infty, \infty]$ instead of $[0, 2\pi]$.

In the same way as we deformed the contour for the integral (14.4), as was shown in Figure 14.7.4, and is again shown more schematically still in Figure 14.7.7(a), we now deform the contour for (14.5) as is shown in Figure 14.7.7(b).

Since $\Xi(\xi)$is 2π-periodic, the poles will in the two cases have the same imaginary parts, and therefore the exponential decay rates will be the same for $s_C(x)$ as was found for λ_k. The singularity at the origin will be canceled by the factor $\widehat{\phi}(\xi)$ in

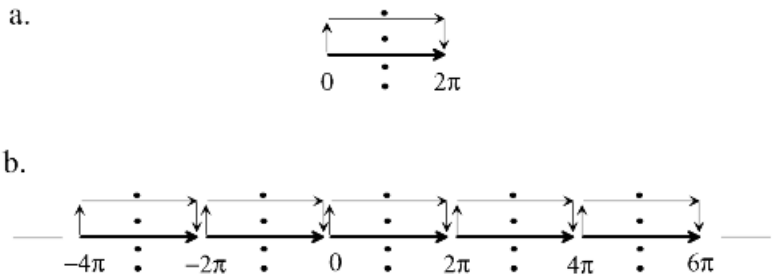


FIGURE 14.7.7. Original and modified integration contours for evaluating (a) the integral (14.4) for λ_k and (b) the integral (14.5) for $s_C(x)$.

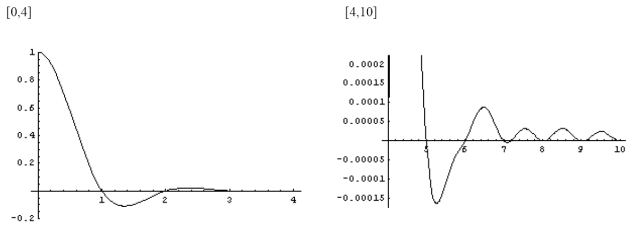


FIGURE 14.7.8. Cardinal interpolant $s_C(x)$ in the case of IQ, $\varepsilon = 1$, shown over the intervals (a) [0,4] and (b) [4,10].

the numerator of (14.5), but the contributions from other multiples of 2π will not be canceled, so algebraic decay rates (if at all present) will also be the same (to leading order) for λ_k and $s_C(x)$.

We illustrate the general observations above with the case of IQ. In this case, it transpires that we can simplify (14.5) to

$$s_C(x) = \frac{\sinh \frac{2\pi}{\varepsilon} \sin \pi x}{\pi \varepsilon x (\cosh \frac{2\pi}{\varepsilon} - \cos 2\pi x)} \int_0^\pi \frac{\cos x\xi}{\cosh^2(\frac{\xi}{\varepsilon})} \, d\xi \ .$$

Figure 14.7.8 illustrates how this cardinal data interpolant at first decays in an oscillatory manner at an exponential rate, followed by algebraic decay without changes of sign, entirely as predicted by the general argument above.

Considering the fast decay of cardinal RBF interpolants, it is clear that superposing translates of these according to (14.8) will give results for $s_G(x)$ which are qualitatively the same as those for the cardinal interpolant $s_C(x)$. With help of the

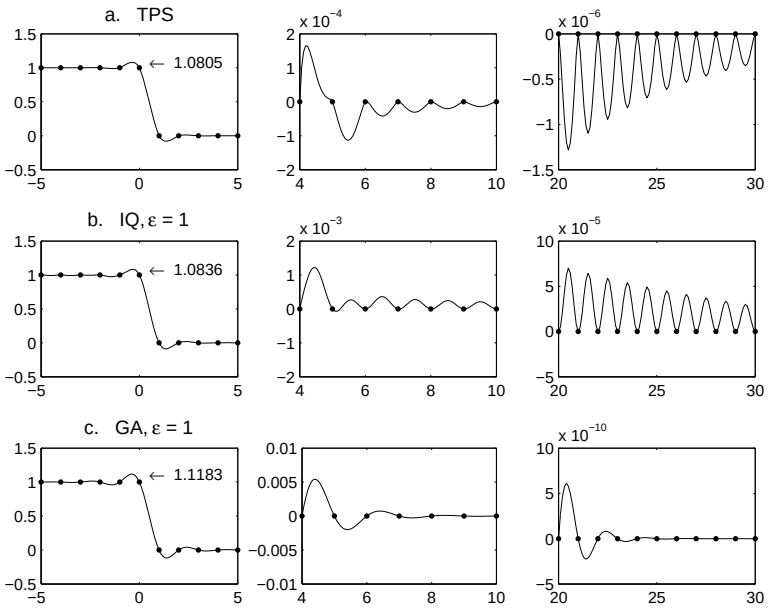


FIGURE 14.7.9. The Gibbs oscillations around a jump, and further out to the right, in the cases of (a) TPS, (b) IQ, ε=1, and (c) GA, ε=1.

formulas (14.5) and (14.8), one can readily compute the Gibbs interpolants for any radial function.

Figures 14.7.9(a)—(c) show the Gibbs oscillations in a number of cases. In accordance with the analysis, we can note that the oscillations decay exponentially for all distances in cases when $\frac{1}{\widehat{\phi}(\xi)}$ is analytic around the origin (here shown only in the case of GA), but otherwise there will at some distance be a transition to one-sided oscillations which decay at a slower algebraic rate. For the infinitely smooth RBF, it is also of interest to see how the Gibbs phenomenon varies with the shape parameter ε.

As $\varepsilon \to 0$, the oscillations seen in Figure 14.7.10 c (MQ, $\varepsilon = 0.1$) increasingly resemble the trigonometric interpolation case shown in Figure 14.7.6b. The transition point between exponential and algebraic decay, visible around $x = 4$ in the case of $\varepsilon = 10$ (Figure 14.7.10(a)) and around $x = 16$ for $\varepsilon = 1$ (Figure 14.7.10(b)) has in the $\varepsilon = 0.1$ case moved too far out to be visible in computations carried out in

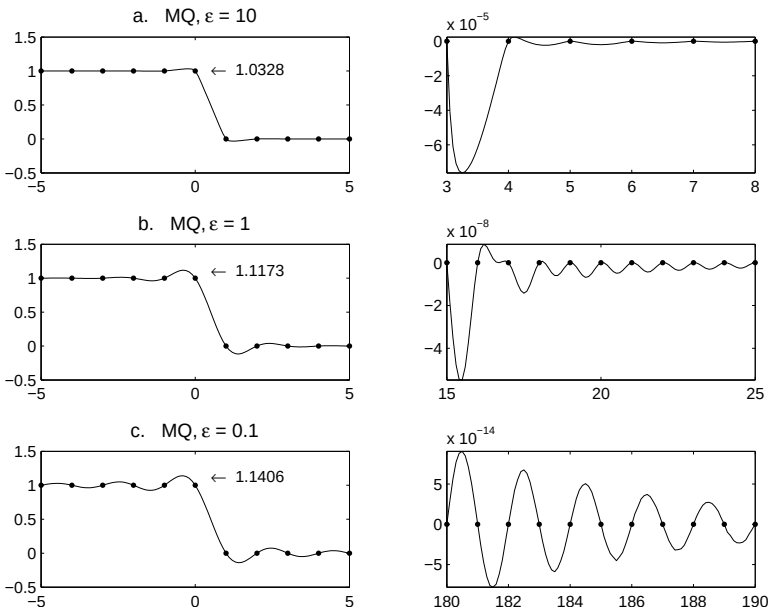


FIGURE 14.7.10. The Gibbs oscillations for MQ in the case of different values of the shape parameter (a) $\varepsilon = 10$, (b) $\varepsilon = 1$, and (c) $\varepsilon = 0.1$.

standard 16-digit numerical precision. In this limit, the exponential decay has itself slowed up, and turned into the slow algebraic one of trigonometric interpolation.

In many situations, the Gibbs oscillations are undesirable. As illustrated in Figure 14.3.7 (Section 14.7.2.3), a spatially variable shape parameter can be very effective in eliminating these.

CHAPTER 15

THE FFT ALGORITHM

15.1. Introduction

The discovery by Cooley and Tukey (1965) of the FFT algorithm caused one of the greatest computational revolutions of all times. Applications of the FFT soon proved abundant in nearly all fields. Not only could many existing tasks be solved orders of magnitude faster, computing could be brought to bear on new areas. The FFT principle has since been found in several earlier works, e.g. by Gauss (1866) and Runge (1903, 1905). It is described in a numerical survey book by Runge and König (1924) and again in a book on trigonometric computations by Stumpff (1939). X-ray crystallographers in Cambridge used the method in the 1930's. It is described in this context by Danielson and Lanczos (1942). Still, it remained on the fringes of numerical knowledge until its revolutionary potential became apparent to the rediscoverers James W. Cooley and John W. Tukey (Tukey was a professor of statistics at Princeton University, and Cooley then a programmer at IBM's Thomas J. Watson Research Center, Yorktown Heights, NY).

Part of the reason the FFT idea had not had much impac earlier was that electronic computers were then not available. Clever symmetries had been found that were just as effective in speeding up the calculations for the small problem sizes that were all that then could be handled. By the end of the 1960's, big computers performed Fourier transforms on data sets with thousands of points.

A dramatic example of the impact of the FFT is described by Cooley et.al. (1969) regarding the spectral analysis of a seismogram recording from the big Alaska earthquake of 1965. A 2048 point discretization of the seismic trace seen in Figure 15.1.1 required 26 minutes to transform on a then state-of-the-art computer (producing the spectrum seen in Figure 15.1.2. With the FFT algorithm, this same task took 2.4 seconds (and the result also became more accurate). The difference between the

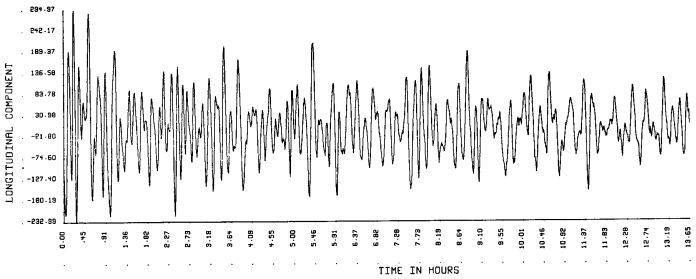


FIGURE 15.1.1. Strain seismograph recording of Rat Island earthquake; $N = 2048$, $T = 13\frac{1}{2}$ hours.

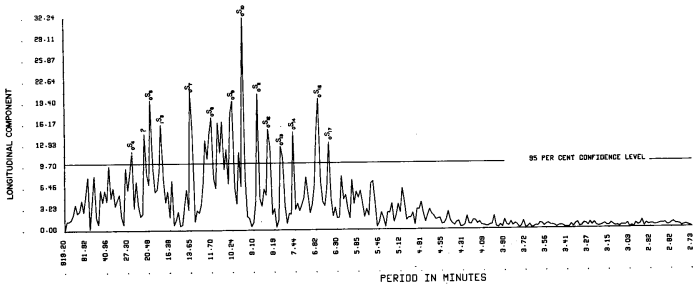


FIGURE 15.1.2. Power spectrum of the recording in the previous Figure.

operation counts of $O(n^2)$ and $O(n \log n)$ become larger still when n is increased further - as is very often the case in present applications.

15.2. FFT implementations

We noted in Section 7.4 that the discrete Fourier Transform (DFT), given by the equations

$$u_j = \sum_{k=0}^{N-1} \hat{u}_k e^{2\pi i k j/N} \quad , \quad j = 0, 1, \ldots, N-1 \tag{15.1}$$

and

$$\hat{u}_k = \tfrac{1}{N} \sum_{j=0}^{N-1} u_j e^{-2\pi i k j/N} \quad , \quad k = 0, 1, \ldots, N-1. \tag{15.2}$$

are often best expressed in matrix×vector form. For example, in the case of (15.1) we get

$$
(15.3)\qquad
\begin{bmatrix}
1 & 1 & 1 & 1 & \dots & 1 \\
1 & \omega & \omega^2 & \omega^3 & \dots & \omega^{N-1} \\
1 & \omega^2 & \omega^4 & \omega^6 & \dots & \omega^{2N-2} \\
\vdots & \vdots & & & & \vdots \\
\vdots & \vdots & & & & \vdots \\
1 & \omega^{N-1} & \dots & \dots & \dots & \omega^{(N-1)^2}
\end{bmatrix}
\begin{bmatrix} \hat{u}_0 \\ \hat{u}_1 \\ \hat{u}_2 \\ \vdots \\ \vdots \\ \hat{u}_{N-1} \end{bmatrix}
=
\begin{bmatrix} u_0 \\ u_1 \\ u_2 \\ \vdots \\ \vdots \\ u_{N-1} \end{bmatrix}
$$

where $\omega = e^{2\pi i/N}$ is primary the N^{th} root of unity.

We note that the matrix is independent on the data (which appears only in the $\hat{\mathbf{u}}$ and $\mathbf{u}$ - vectors). It turns out that we can factorize the DFT matrix into a product of a few very sparse matrices. That allows us to carry out what otherwise would be a full matrix×vector multiplication instead as a sequence of a few very sparse matrix×vector multiplications. The FFT algorithm is an implementation of that idea.

15.2.1. Cooley-Tukey factorization. Figure 15.2.1 shows in the case of $N = 8$ a factorization of the DFT matrix into a product of sparse matrices. From left to right, these factors are

- a very sparse matrix (two entries per each row only),
- a 2×2 block-diagonal matrix in which each diagonal block is again a DFT matrix, but of half size, and
- a permutation matrix.

It is straightforward to verify that this factorization works whenever N is an even number (remembering that $\omega^{k+N} = \omega^k$ and $\omega^{k+N/2} = -\omega^k$). In the $N = 8$ case, we repeat this splitting idea on the two 4×4 blocks, and then on the resulting four 2×2 blocks. That will generate (emerging at the left in the three steps) the first three matrices seen in the top display in Figure **??**. The three permutation matrices that emerged to the right in each of the steps can be combined into a single permutation matrix. The full 8×8 DFT matrix we started with is therefore equal to the Coley-Tukey product seen at the top of Figure 15.2.2. The generalization to a matrix of

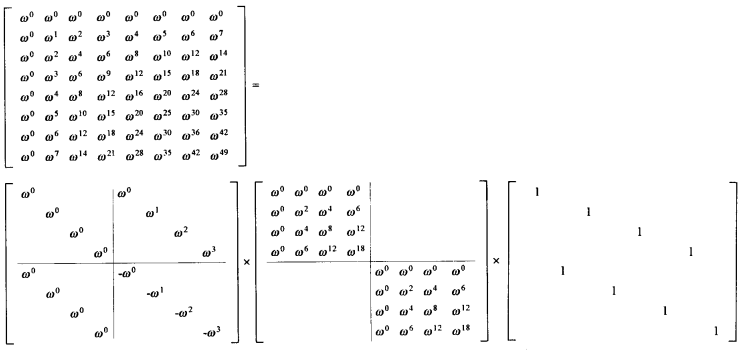


FIGURE 15.2.1. One step in FFT factorization of the DFT matrix, shown for size $N = 8$.

size $N = 2^m$ is immediate. We get $m = \log_2 N$ sparse factors with only $2N$ non-zero entries each (half of which are ones, thus costing no multiply operations) The total operation count for the matrix×vector multiplication has been reduced from $O(N^2)$ to $O(N \log N)$.

A couple of observations:

(1) **Vector overwriting:** Usually when one performs a matrix×vector multiplication, one needs to store two separate vectors. As the multiplication is in progress, it is not allowed to let the output vector overwrite the input vector. This time, the matrix factors happen to have a structure that allows this (assuming new wntries are computed pairwise). Of course, no storage is needed for the matrix factors—they are cheaply generated as the algorithm proceeds.

(2) **Generation of permutation matrix:** Table 1 illustrates the idea of *bit reversal* in our 8×8 case; this generalizes directly to any size $N = 2^m$. This principle can be shown for example by induction over m, together with noting the structure of the individual permutation matrices that emerged to the right. There are very quick ways available to implement this computationally.

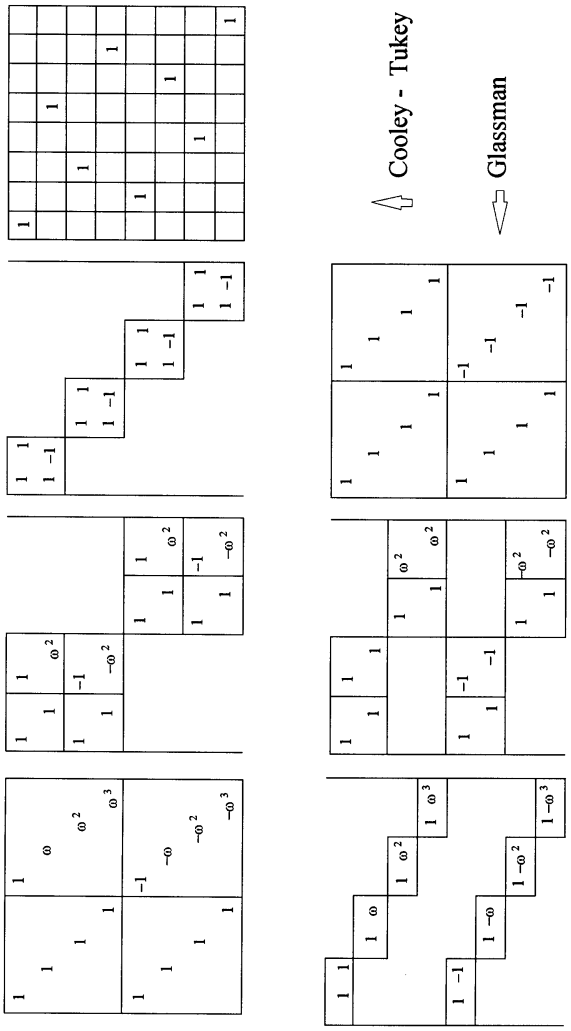


FIGURE 15.2.2. Cooley-Tukey and Glassman factorizations of the DFT matrix in case of $N = 8$.

15.2.2. Glassman factorization. An alternative to the Cooley-Tukey factorization was presented by Glassman (1970) - seen in the bottom part of Figure 15.2.2. This can be derived by step-by-step incorporating the successive permutation matrices into the factors that emerged to the left. This factorization may be the easiest of all to implement in a computer code, and the absence of permutations can be

Illustration of bit reversal in case of N=8

successive integers		expressed in binary	binary digits reversed		becomes	equal to position of entry in column
0	=	000	000	=	0	0
1	=	001	100	=	4	1
2	=	010	010	=	2	2
3	=	011	110	=	6	3
4	=	100	001	=	1	4
5	=	101	101	=	5	5
6	=	110	011	=	3	6
7	=	111	111	=	7	7

TABLE 1. Illustration of bit reversal. Note that we number the matrix rows and columns starting from zero.

particularly adventageous on vector- or parallel computers that are fast for arithmetic operations, but do 'data shuffling' relatively slowly. However, no successive overwriting of output vector on top of the input vector is allowed this time.

There exists several other sparse factorizations of the DFT matrix than the two we have described. For other algorithms (although usually not described in the language of matrix factorizations), see for ex. [..], [..].

15.3. A selection of FFT applications

Applications of the FFT algorithm are ubiquitous in almost all areas that include computing. The few brief examples here barely start to scratch the surface. In most problems that feature some form of periodicity (e.g. PDEs on a spatially periodic domain) very powerful numerical methods can be based on numerical transform to Fourier space (in which the usually delicate task of taking derivatives reduces to regular multiplications). In this book, we have looked at the Fourier method for tomographic inversion Chapter 1 and later, in Chapter 30, for X-ray analysis of crystals. In this section, we will first mention how it can be used for image processing. This is then followed by a few uses in basic numerical algorithms.

15.3.1. Image enhancement. A common task is to sharpen an image that is blurred, for example by camera motion or bad focusing. As an example, let us

consider the sharp image in Figure (15.3.1), and one of its horizontal scan lines. If the camera was turned sideways during the exposure, the brightness trace would be smeared just as if it had been convolved with a matrix whose rows are shifted versions of a rectangular pulse, as shown in Figure (15.3.2). Comparing the sketch of this matrix×vector multiply with the discrete convolution theorem as written in equation ...., we can identify the vectors x, y, z as follows:

z rectangular pulse
x sharp trace
y blurred trace

Given any two of these vectors, we quickly find the third by applying (...). Our interest is of course to recover x when y and z are given (or if the correct z is not known, we can try with different z and see what works the best). The idea we have here sketched in 1-D generalizes immediately to 2-D. For example, if the imaging error was bad focusing, the equivalent z would be a 2-D matrix featuring a circularly symmetric smooth hump.

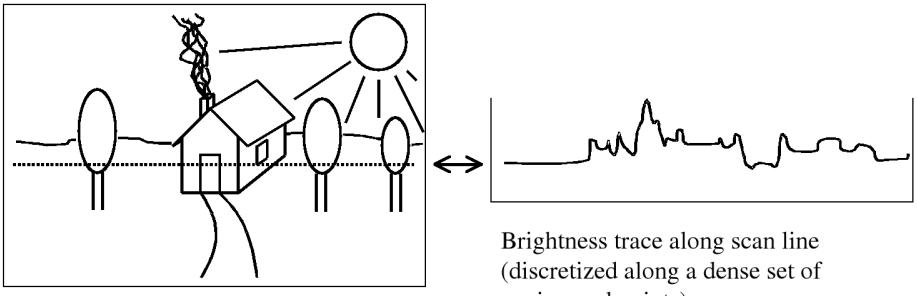


FIGURE 15.3.1. Example of an image and the grey level of a scan line through it.

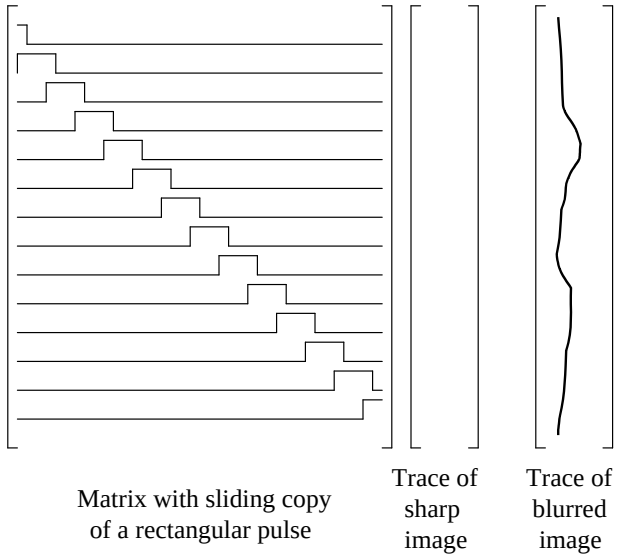


FIGURE 15.3.2. Schematic illustration of how the convolution theorem relates sharp and blurred images.

15.3.2. Numerical Chebyshev expansion of a function. Any continuous function on, say, [-1,1] can be approximated arbitrarily closely by a polynomial. A bad way to do this is to interpolate the function at increasingly many equispaced points - the polynomial will usually oscillate violently near the ends of the interval. We saw the cause of this *Runge phenomenon* in Section 12.4 on Lagrange's interpolation formula. We saw also (cf. Figure 12.4.1 vs. Figure 12.4.6) that these spurious oscillations could be suppressed by simply clustering the interpolation nodes denser towards the edges. The choice

$$x_k = -\cos\frac{\pi k}{N}, \qquad k = 0, 1, 2, \ldots, N$$

is particularly effective. Convergence of the interpolating polynomial $p_N(x)$ to the function $f(x)$ is then usually very rapid as N increases. For this particular set of nodes, it turns out that $p_N(x)$ can be found much faster using the Fast Cosine Transform (FCT; see Section 7.4) than by using Lagrange's or Newton's interpolation formulas. For this, we need first to introduce the *Chebyshev polynomials*

$$\begin{array}{lll} T_0(x) & = \cos(0 \arccos x) & = 1 \quad , \\ T_1(x) & = \cos(1 \arccos x) & = x \quad , \\ T_2(x) & = \cos(2 \arccos x) & = 2x^2 - 1 \quad , \\ T_3(x) & = \cos(3 \arccos x) & = 4x^3 - 3x \quad , \\ \vdots & \vdots & \vdots \\ T_n(x) & = \cos(n \arccos x) & = 2^{n-1}x^n - \ldots \quad . \end{array}$$

$T_n(x)$ is clearly a polynomial of degree n for $n = 0$ and for $n = 1$. For higher n the result follows for ex. from the three-term recursion formula $T_{n+1}(x) = 2x\ T_n(x) - T_{n-1}(x)$ (a consequence of the trigonometric identity $\cos((n+1)\alpha) + \cos((n-1)\alpha) = 2\cos\alpha\ \cos n\alpha$ if we let $\alpha = \arccos x$, i.e. $\cos\alpha = x$).

Making the polynomial

$$p_N(x) = \sum_{\nu=0}^{N} \alpha_\nu T_\nu(x)$$

agree with $f(x)$ at the nodes $x_k = -\cos\frac{\pi k}{N}$ amounts to finding values of α_ν such that

$$\sum_{\nu=0}^{N} \alpha_\nu T_\nu(x_k) = f(x_k) \quad .$$

With $T_\nu(x_k) = \cos(\nu \arccos(-\cos\frac{\pi k}{N})) = \cos\nu\frac{\pi k}{N}$, this becomes

$$\sum_{\nu=0}^{N} \alpha_\nu \cos\nu\frac{\pi k}{N} = f(x_k), \quad k = 0, 1, 2, \ldots, N$$

Therefore, the desired expansion coefficients are obtained as the output of an FCT applied to the function values $f(x_k),\ k = 0, 1, 2, \ldots, N$.Also, given the coefficients, this allows a quick evaluation of the function values at the Chebyshev points.

If one wants, one can re-sort the expansion $\Sigma_{\nu=0}^{N}\alpha_\nu T_\nu(x)$ into the more usual form for a polynomial $\Sigma_{\nu=0}^{N}\beta_\nu x^\nu$, but there is often little reason to do this. Thanks to many formulas relating Chebyshev polynomials of different orders, these are fast to evaluate and easy to manipulate (to differentiate etc.). A notable advantage with them is that for any $f(x)$ continuous on [-1,1], there is a unique expansion $f(x) = \Sigma_{\nu=0}^{\infty}\alpha_\nu T_\nu(x)$. Truncations of this series offer excellent approximations to $f(x)$. The situation for power series is much less attractive. For $f(x) = \Sigma_{\nu=0}^{\infty}\beta_\nu x^\nu$ to hold, it is not even sufficient that $f(x)$ is infinitely many times differentiable over [-1,1]. And even when there is a convergent power series expansion, truncations are usually accurate only near $x = 0$.

15.3.3. Numerical computation of Taylor coefficients. Although this is a task that is not particularly often needed, it is interesting to note that the DFT makes it possible and the FFT makes it very fast. If a function $f(x)$ can be numerically computed only for real arguments x (the most common situation), we can not do much better than applying some finite difference approximation from Section.... Especially approximations for high derivatives become very sensitive to small errors in function values.

For example, if we approximate a fourth derivative by

(15.1)
$$f^{(4)}(x) = \frac{f(x-2h) - 4f(x-h) + 6f(x) - 4f(x+h) + f(x+2h)}{h^4} + O(h^2) \quad ,$$

we need h to be small so that the truncation error $O(h^2)$ is small. But then, the h^4 in the denominator is far smaller still, causing errors in the terms in the numerator to be greatly magnified. Another way to put it: The numerator must be small like the denominator to evaluate to an $O(1)$ result. It must therefore involve serious cancellation of digits - the ONLY way to loose large numbers of significant digits in floating point arithmetic.

If $f(x)$ is an analytic function, i.e. once differentiable in the sense that $f'(x) = \lim_{h\to 0}\frac{f(x+h)-f(x)}{h}$ exists for all *complex* $h \to 0$, it can then be shown to automatically

be infinitely many times differentiable (only one of very many remarkable theorems regarding analytic functions).

then $f(x)$ can be Taylor expanded around x:

$$f(x+h) = f(x) + h\frac{f'(x)}{1!} + h^2\frac{f''(x)}{2!} + h^3\frac{f'''(x)}{3!} + \dots \quad . \tag{15.2}$$

If we sample $f(x)$ not at equispaced points on the real axis in the neighborhood of x (such as in (15.1)) but instead around a circle centered at x in the complex plane $z = x + h$ with $h = r\ e^{i\theta},\ 0 \le \theta \le 2\pi$, then (15.2) becomes

$$f(x + r\ e^{i\theta}) = f(x) + r\frac{f'(x)}{1!}\ e^{i\theta} + r^2\frac{f''(x)}{2!}\ e^{2i\theta} + r^3\frac{f'''(x)}{3!}\ e^{3i\theta} + \dots \quad .$$

The RHS is a Fourier series. Once we have computed the LHS for equispaced values of θ, a standard complex FFT will give approximations to the Fourier coefficients (which are $r^k f^{(k)}(x)/k!-$ hence also to the derivatives $f^{(k)}(x)$). In this procedure, the variable r is a free parameter. Fornberg (1981 a,b) describe in some detail how results for a few different values of r can be combined to get very accurate approximations to the Taylor coefficients (or derivatives).

15.3.4. Multiplication of large polynomials. The product of

$$\begin{aligned} p_1(x) &= a_0 + a_1x + a_2x^2 + \dots + a_Nx^N \qquad \text{and} \\ p_2(x) &= b_0 + b_1x + b_2x^2 + \dots + b_Nx^N \end{aligned}$$

is

$$\begin{aligned} p_1(x)\cdot p_2(x) \ = \ & 1 && \cdot\{a_0b_0\}+ \\ & x && \cdot\{a_1b_0 + a_0b_1\}+ \\ & x^2 && \cdot\{a_2b_0 + a_1b_1 + a_0b_2\}+ \\ & \vdots \\ & x^N && \cdot\{a_Nb_0 + a_{N-1}b_1 + a_{N-2}b_2 + \dots + a_1b_{N-1} + a_0b_N\}+ \\ & \vdots \\ & x^{2N-1} && \cdot\{a_Nb_{N-1} + a_{N-1}b_N\}+ \\ & x^{2N} && \cdot\{a_Nb_N\} \\ \\ & = && c_0 + c_1x + c_2x^2 + \dots + c_{2N}x^{2N}. \end{aligned}$$

This can be written in matrix×vector form as

$$\begin{bmatrix} a_0 & 0 & \dots & \dots & 0 & a_N & \dots & a_2 & a_1 \\ a_1 & a_0 & 0 & \dots & \dots & 0 & a_N & \dots & a_2 \\ a_2 & a_1 & a_0 & 0 & \dots & \dots & 0 & a_N & \dots \\ \dots & a_2 & a_1 & a_0 & 0 & \dots & \dots & 0 & a_N \\ a_N & \dots & a_2 & a_1 & a_0 & 0 & \dots & \dots & 0 \\ 0 & a_N & \dots & a_2 & a_1 & a_0 & 0 & \dots & \dots \\ \dots & 0 & a_N & \dots & a_2 & a_1 & a_0 & 0 & \dots \\ \dots & \dots & 0 & a_N & \dots & a_2 & a_1 & a_0 & 0 \\ 0 & \dots & \dots & 0 & a_N & \dots & a_2 & a_1 & a_0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ : \\ b_N \\ 0 \\ : \\ : \\ 0 \end{bmatrix} = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ : \\ c_N \\ : \\ : \\ : \\ c_{2N} \end{bmatrix}$$

where we have introduced N zeros following the $N+1$ elements of the a- and b-vectors (we can pad with additional zeros if we want to make the vector sizes a power of two or something else that is particularly fast for the FFT). This matrix×vector product is of the form that was shown in equation .... to allow a fast evaluation by means of the discrete convolution theorem. The cost becomes $O(N \log N)$ operations rather than $O(N^2)$ if computed directly.

15.3.5. Filtering. Basic application of the convolution theorem. To be written.

15.3.6. Sharpening blurred images. Suppose you are taking a picture of the moon using a long exposure. Even if your camera is mounted on a very sturdy tripod, the motion of the moon across the sky during the exposure blurs your image. Since you know exactly what caused the blur—it is just a convolution with an appropriate blurring function—there is the possibility that the blurring can be undone.

Figure 15.3.3 shows an image of the moon. Find an appropriate function to simulate the motion of the moon, convolve that with the image to caused a blurred moon. Now undo the effect. To be written

FIGURE 15.3.3. Original, unblurred image of the moon.

CHAPTER 16

NUMERICAL METHODS FOR ODE INITIAL VALUE PROBLEMS

16.1. Introduction.

A scalar first order *ordinary differential equation* (ODE) takes the form

$$y' = f(t, y) \,, \tag{16.1}$$

where $y = y(t)$ is a function to be determined. It can be illustrated by a vector field, as shown in Figure 16.1.1 for the case of

$$y' = t^2 + y^2 \,. \tag{16.2}$$

At each location in a (t, y)-plane, equation (16.2) tells in which direction a solution curve should go. The analytic solution is a continuous function that everywhere follows the required directions. Even for very simple-looking ODEs, it may be impossible to find closed-form expressions for solutions. In the case of (16.2), together with the *initial condition* (IC) $y(0) = 0$, it happens to be possible:

$$y(t) = - \frac{t \, \left(J_{-3/4}(\frac{1}{2}t^2) - Y_{-3/4}(\frac{1}{2}t^2) \right)}{J_{1/4}(\frac{1}{2}t^2) - Y_{1/4}(\frac{1}{2}t^2)} \,,$$

but its complexity makes it of limited value (J and Y denote here Bessel functions of fractional orders; this solution is shown as the smooth curve starting at the origin in Figure 16.1.1).

In complete contrast to analytical methods, numerical techniques for ODEs do not get any more complicated if the ODE is nonlinear rather than linear, or if it is

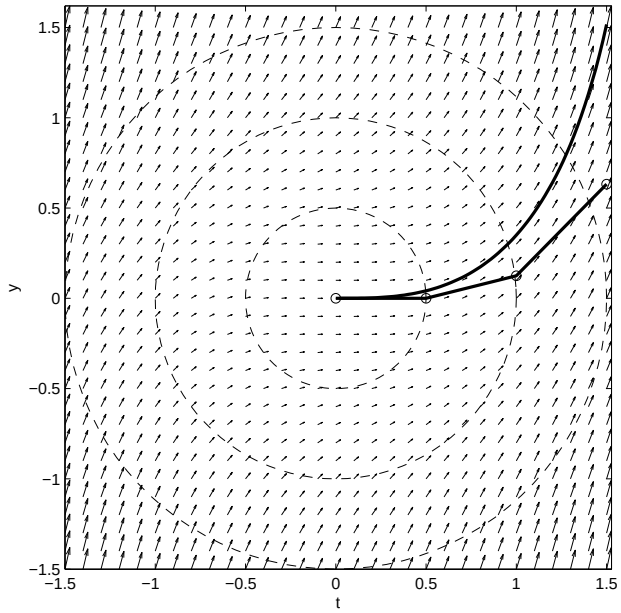


FIGURE 16.1.1. llustration of solutions to the ODE (16.2). The arrows show the direction field. The smooth and the piecewise linear curves show the analytic solution and the Euler approximation (with step $k = 0.5$) in the case of the IC $y(0) = 0$. The dashed circles are examples of *isoclines* - curves along which the slopes are constant.

generalized from one scalar ODE to a system of many coupled ODEs, for ex.

$$(16.3) \qquad \begin{cases} y_1' &= f_1(t, y_1, y_2, \ldots, y_n) \\ y_2' &= f_2(t, y_1, y_2, \ldots, y_n) \\ \vdots & \\ y_n' &= f_n(t, y_1, y_2, \ldots, y_n) \end{cases} \quad \text{with IC} \quad \begin{cases} y_1(0) &= a_1 \\ y_2(0) &= a_2 \\ \vdots & \\ y_n(0) &= a_n \end{cases}.$$

The ease with which we will soon see that first order systems can be handled numerically makes it attractive to turn higher order equations into coupled systems of first order ones. For example, instead of

$$y''' = f\ (t, y, y', y'') \text{ with IC } y(0) = a_0,\ y'(0) = a_1,\ y''(0) = a_2,$$

we introduce extra dependent variables through $y_0(t) = y(t)$, $y_1(t) = y'(t)$, $y_2(t) = y''(t)$ and then solve

$$\begin{cases} y_0' &= y_1 \\ y_1' &= y_2 \\ y_2' &= f\ (t, y_0, y_1, y_2) \end{cases} \qquad \text{with IC} \quad \begin{cases} y_0(0) &= a_0 \\ y_1(0) &= a_1 \\ y_2(0) &= a_2 \end{cases} .$$

In Section 16.2, we introduce the simplest possible numerical ODE scheme, *Forward Euler* (FE). In its basic form, it is very inaccurate (as seen already in Figure 16.1.1), and it is rarely used. We give in Section 16.3 numerous examples of linear multistep (LM) methods, which represent one direction for generalizing FE towards higher accuracy and efficiency. It will transpire that some of these schemes that are described are very efficient while others, appearing equally plausible, in fact are entirely useless. The analysis that is needed to address this involves four key concepts: accuracy, consistency, stability and stability domain, and is given in Section 16.4. With the help of this additional background, we discuss in Section 16.5 predictor-corrector methods, in Section 16.6 Runge-Kutta methods and in Section 16.7 Taylor series methods. In the concluding Section 16.8, we describe stiff ODEs and how stability domains give insights into how to choose between different ODE methods.

16.2. Forward Euler (FE) scheme.

The numerical solution produced by this scheme is made up of piecewise straight line segments, each one with the slope of the direction field at its start location. A step from time t to time $t + k$ amounts to using the first two terms of a Taylor expansion of the solution at time t:

$$\underbrace{y(t+k) = y(t) + k\ y'(t)}_{\text{Use for Forward Euler}} \quad \underbrace{+ \frac{k^2}{2} y''(t) + \ldots}_{\text{Local error } O(k^2)} \tag{16.1}$$

Substituting the ODE (16.1) for y', we get the forward Euler method

$$y(t+k) = y(t) + k\ f(t, y)\ . \tag{16.2}$$

A very convenient way to graphically illustrate the stencil of this scheme is as follows:

$$
\begin{array}{lccccl}
 & & y & f & & \\
\text{time level} & & \Downarrow & \Downarrow & & \\
\text{new: } t+k & & \boxdot & & \boxdot/\boxplus & \text{unknown/known values for } y \\
 & - & + & - & \odot/\oplus & \text{unknown/known values for } f \\
\text{old: } t & & \boxplus & \oplus & &
\end{array}
\tag{16.3}
$$

or, more compactly, $\begin{bmatrix} \boxdot & \\ \boxplus & \oplus \end{bmatrix}$.

Each time step, the local error is of size $O(k^2)$. In order to advance over a time interval of fixed duration, we need to take $O(1/k)$ time steps. It is therefore not surprising that the total error can be shown to become of the size $O(k^2) \cdot O(1/k) = O(k^1)$ as $k \to 0$. The FE scheme is therefore known as a *first order* scheme.

16.3. Examples of linear multistep (LM) methods.

The general idea behind linear multistep (LM) methods is to extend the computational stencil shown for forward Euler in (16.3) backwards to still earlier time levels. In the examples below, we do this in various ways. In all the cases, once the shape of the stencil is decided on, we find the actual coefficients to use (giving the highest local accuracy) most easily by means of the two-line Padé-based Mathematica algorithm described in Section 12.5:

```
t = Pade[xs*Log[x]m,{x,1,n,d}]; {CoefficientList[Numerator[t],x],CoefficientList[Denc
```

The parameter m is here always one since we are approximating a first derivative; the other three numbers s, n and d describe the 'shape' of the stencil, as is described in Section 12.5. and is also illustrated in Figure 16.3.1.

EXAMPLE 57.

$\begin{bmatrix} \boxdot & \\ \boxplus & \oplus \\ & \oplus \\ & \oplus \end{bmatrix}$

Scheme: $y(t+k) = y(t) + k\,\left[\frac{23}{12}y'(t) - \frac{4}{3}y'(t-k) + \frac{5}{12}y'(t-2k)\right]$
Find coeff: $m = 1,\ s = -2,\ n = 1,\ d = 2$
Accuracy: 3^{rd} order

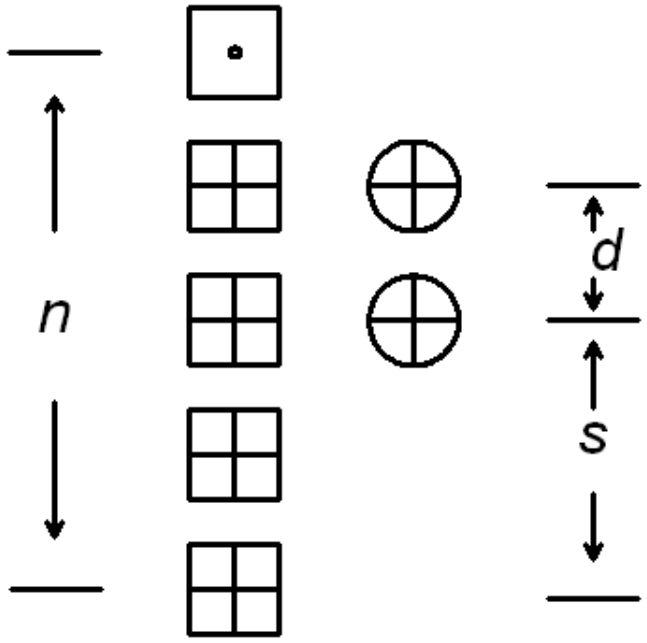


FIGURE 16.3.1. Illustration of how the shape of a linear multistep stencil is described by the parameters s, n and d. If the right column of entries descends further down than the left column, s will be negative. While n and d need to be integers, we will in a later context consider time-staggared stencils for which s is a half-integer.

The output of the Mathematica algorithm is $\{\{-1,1\},\{\frac{5}{12},-\frac{4}{3},\frac{23}{12}\}\}$ which should be interpreted as

$$-1\ y(t)+1\ y(t+k)=\tfrac{5}{12}y'(t-2k)-\tfrac{4}{3}y'(t-k)+\tfrac{23}{12}y'(t).$$

In this and the following examples, $y'(\cdot)$ should be replaced by $f\ (\cdot,y(\cdot))$, i.e. the right hand side of the ODE at time level " $\cdot$ ". In this scheme, the stencil is extended backwards two additional steps in the right column compared to the FE scheme. Schemes of this general type (extending backwards in the right column only) are known as Adams-Bashforth schemes. Every step backwards relative to the FE scheme increases the order by one. Due to its third order of accuracy, we denote the present scheme as AB3. □

EXAMPLE 58.

$$\begin{bmatrix} \boxdot & \odot \\ \boxplus & \oplus \\ & \oplus \\ & \oplus \end{bmatrix} \quad \begin{array}{ll} \text{Scheme:} & y(t+k) = y(t) + \\ & +k\ \left[\frac{3}{8}y'(t+k) + \frac{19}{24}y'(t) - \frac{5}{24}y'(t-k) + \frac{1}{24}y'(t-2k)\right] \\ \text{Find coeff:} & m=1,\ s=-2,\ n=1,\ d=3 \\ \text{Accuracy:} & 4^{th} \text{ order} \end{array}$$

This is an example of an *implicit* scheme (as opposed to the AB-type schemes, which are *explicit*). At each time step, both $y(t+k)$ and $y'(t+k) = f\ (t+k, y(t+k))$ are unknown. However, the scheme provides a relation between these two quantities, from which we can solve for $y(t+k)$ (maybe requiring Newton's method in case $f(t,y)$ is a nonlinear function of y). In the next Section 16.4, we will see that, depending on the ODE, this inconvenience may be well worth it. Schemes of this type (two levels for y, while implicit and extending different distances backwards for y') are known as Adams-Moulton schemes. If the right column contains only the entry $\odot$, the resulting first order scheme AM1 is also known as *Backward Euler* (BE). Again, the accuracy increases by one for every further entry in the right column. □

EXAMPLE 59.

$$\begin{bmatrix} \boxdot & \\ \boxplus & \oplus \\ \boxplus & \oplus \\ \boxplus & \oplus \end{bmatrix} \quad \begin{array}{ll} \text{Scheme:} & y(t+k) = [-18y(t) + 9y(t-k) + 10y(t-2k)] - \\ & +k\ [9y'(t) + 18y'(t-k) + 3y'(t-2k)] \\ \text{Find coeff:} & m=1,\ s=0,\ n=3,\ d=2 \\ \text{Accuracy:} & 5^{th} \text{ order} \end{array}$$

Here, we are attempting (and we indeed succeed) to reach a very high formal order of accuracy by extending the stencil backwards in both of its columns. However, as we will see soon, the result is nevertheless a disaster. When letting $k \to 0$, the numerical solution will explode to infinity. Although the scheme is useless for all practical work, it will serve as a good motivation for why we need the convergence theory that is summarized in Section 16.4. □

EXAMPLE 60.

$$\begin{bmatrix} \boxdot & \odot \\ \boxplus & \\ \boxplus & \\ \boxplus & \end{bmatrix} \quad \begin{array}{ll} \text{Scheme:} & y(t+k) = \left[\frac{18}{11}y(t) - \frac{18}{22}y(t-k) + \frac{1}{3}y(t-2k)\right] + \\ & \quad - k\ \frac{6}{11}y'(t+k) \\ \text{Find coeff:} & m=1,\ s=3,\ n=3,\ d=0 \\ \text{Accuracy:} & 3^{rd} \text{ order} \end{array}$$

This scheme generalizes BE by extending two more levels backwards in the left column (and its accuracy is consequently two orders higher than that of BE). Schemes of this type are called Backward Differentiation (BD) methods, with this case being abbreviated as BD3. Again, as the analysis in Section 16.4 will show, this class of implicit schemes can be very attractive in certain situations. Although AB and AM schemes can be successfully brought to any order by just pushing increasingly far back in the second column, there will turn out to be a barrier against increasing BD schemes past order 6. If that is attempted, the same disaster will arise as in Example 3. □

The four examples above have illustrated three important families of LM methods: AB, AM, and BD. In the Section 16.5, we will see how the AB and AM classes can be combined in a particularly effective way.

We conclude this section by showing how one can determine the accuracy of a LM scheme if its coefficients are given (and not necessarily are the optimal ones with regard to accuracy). One example suffices to illustrate the procedure.

Example 61.

$$\begin{bmatrix} \boxdot & \\ \boxplus & \oplus \\ \boxplus & \oplus \end{bmatrix} \quad \begin{array}{ll} \text{Scheme:} & y(t+k) = [3y(t) - 2y(t-k)] + \\ & \quad + k\ \left[\frac{1}{2}y'(t) - \frac{3}{2}y'(t-k)\right] \\ \text{Accuracy:} & \text{To be determined} \end{array}$$

Since each LM formula is an example of a finite difference formula, relating y and y'-values, we can obtain the scheme's order simply by checking for how high powers of t the formula is exact. We test the error $E(t) = \{LHS\} - \{RHS\}$ of the present

scheme as follows:

$$\begin{cases} y(t) = 1,\ y'(t) = 0 \\ E(t) = \{1\} - \{[3-2] + k[\frac{1}{2}0 - \frac{3}{2}0]\} = 0 \end{cases}$$

$$\begin{cases} y(t) = t,\ y'(t) = 1 \\ E(t) = \{t+k\} - \{[3t - 2(t-k)] + k[\frac{1}{2}1 - \frac{3}{2}1]\} = 0 \end{cases}$$

$$\begin{cases} y(t) = t^2,\ y'(t) = 2t \\ E(t) = \{(t+k)^2\} - \{[3t^2 - 2(t-k)^2] + k[\frac{1}{2}2t - \frac{3}{2}2(t-k)]\} = 0 \end{cases}$$

$$\begin{cases} y(t) = t^3,\ y'(t) = 3t^2 \\ E(t) = \{(t+k)^3\} - \{[3t^3 - 2(t-k)^3] + k[\frac{1}{2}3t^2 - \frac{3}{2}3(t-k)^2]\} = \frac{7}{2}k^3 \neq 0 \end{cases}$$

Given that the scheme fails to be exact for $y(t) = t^3$, it is a second order scheme. Since the result in this type of test will not depend on the value of k, one can simplify the algebra by first setting $k = 1$. □

16.4. Key numerical ODE concepts.

16.4.1. Convergence. The key issues when using a numerical ODE solver are

(1) Will the numerical solution converge to the true solution when $k \to 0$ and, if so,
(2) How fast is the rate of convergence?
(3) How small does the time step k need to be for the scheme to give a qualitatively correct answer?

The first of these questions is answered by the following theorem

THEOREM 62. *A numerical ODE scheme converges to the true solution of an ODE (linear or nonlinear) if and only if both consistency and stability hold.*

Consistency is a very easy concept to check. A scheme is consistent if the *local* error, when the true ODE solution is substituted into the scheme, goes to zero when $k \to 0$. In particular, any ODE scheme that is of at least of first order accuracy is automatically consistent. No scheme that we would ever consider would fail this

condition. Assuming from now on that consistency is satisfied, a scheme will therefore converge if and only if it is stable.

16.4.2. Stability. Another key result states that, in order to check stability, it suffices to test the scheme in case of the trivially simple ODE $y'(t) = 0$. To first illustrate what can go wrong, let us consider the scheme in Example 5. When applied to $y'(t) = 0$, it becomes

$$y(t+k) - 3y(t) + 2y(t-k) = 0. \tag{16.1}$$

This is a 3-term recursion relation, which is most easily solved by forming its characteristic equation

$$r^2 - 3r + 2 = 0,$$

with roots $r_1 = 1$ and $r_2 = 2$. The general solution to (16.1) is therefore

$$y(t + n\ k) = c_1 \cdot 1^n + c_2 \cdot 2^n$$

where c_1 and c_2 are some constants ($c_1 = 2y(t) - y(t+k)$ and $c_2 = y(t+k) - y(t)$). For every time step forward, the term $c_2 \cdot 2^n$ doubles in size. The smaller k is made, the more steps need to be taken, and the numerical solution will thus diverge to infinity (given that machine rounding errors are always present and that we simplified the ODE by setting its RHS to zero, it will not happen in practice that c_2 is *exactly* equal to zero). The cause of the divergence is that the characteristic equation has a root (here $r_2 = 2$) outside the unit circle $|r| = 1$. Omitting a few further details, this leads us to the *root condition*:

THEOREM 63. *A LM scheme is stable if and only if all the roots to its characteristic equation lie inside or on the periphery of the unit circle. If any root is on the periphery, it needs to be a simple root.*

We can note that whenever a scheme is accurate to first order or better, it will admit the exact solution $y(t) \equiv 1$ to the ODE $y'(t) = 0$. From this follows that $r_1 = 1$ is always a root to the characteristic equation. All other roots are 'spurious', and what matters is that powers of these spurious roots must not grow. In the graphical

stencils we use to illustrate LM schemes, stability (as opposed to accuracy) depends only on the entries in the left column.

16.4.3. Stability domains. The concept of *stability domain* (or *domain of absolute stability*) has very little to do with stability, as defined in the previous Section 16.4.2, so the traditional naming convention is unfortunate. To every ODE method corresponds a stability domain (in a complex plane, which we will soon define). For ODEs, this stability domain provides a guide to how small time step one need to take in order to get a 'reasonable' numerical solution. Stability domains provide often a good guide to what type of numerical method to choose for different ODEs, as discussed in Section 16.8. A related application area of stability domains will arise when we, in Chapter ??, apply ODE methods for numerically solving PDEs. In that context, the stability domain will tell whether schemes will converge or diverge when both time and space steps are refined—much like how stability (as tested by the root condition) is the key issue for convergence in the case of ODEs.

We will start the discussion of stability domains by an example which illustrates the need for a practical guide in choosing an appropriate time step:

EXAMPLE 64. Determine how small we need to choose k in order to get a qualitatively reasonable approximation when using forward Euler to solve the ODE

$$y'(t) = -10y(t). \tag{16.2}$$

The analytic solution $y(t) = e^{-10\,t}y(0)$ decays rapidly for increasing t. Suppose we try to solve (16.2) with forward Euler using the time step $k = 1$. The scheme becomes $y(t+1) = y(t) + 1 \cdot (-10y(t)) = -9y(t)$. Obviously, stepping forward in this way will lead to an oscillatory and rapidly divergent numerical solution, bearing no relation to the analytical one. Since forward Euler is consistent and satisfies the root condition (with $r_1 = 1$ as the only root), we know that all will be well in the limit of $k \to 0$. However, in practice one needs to calculate with a finite value of k, so a natural question to ask is how small k need to be so that the numerical solution will not grow in time. For a general value of k, the forward Euler scheme for (16.2) becomes $y(t+k) = y(t) + k \cdot (-10y(t)) = (1 - 10k)\ y(t)$, so the answer becomes $k \leq 1/5$. □

16.4.3.1. *Linearization of a general ODE..* The general ODE (16.1) can, locally around a point $(t_0, y_0) = (t_0, y(t_0))$, be linearized in both t and y:

$$y' \approx \underbrace{f(t_0,y_0) + f_t(t_0,y_0)(t-t_0)}_{g(t)} + \underbrace{f_y(t_0,y_0)}_{\lambda}\underbrace{(y-y_0)}$$

$$; \quad v(t)$$

i.e. $v'(t) \approx \lambda\, v(t) + g(t)$. In the present context, it turns out that we can further ignore $g(t)$, and only the value of $\lambda = f_y(t_0, y_0)$ will remain significant.

16.4.3.2. *Stability domains for AB- and AM-type LM methods.* In the case of FE (AB1), we can compute the *stability domain* as follows:

EXAMPLE 65. Calculate the stability domain for FE.
For the ODE $y' = \lambda y$, FE becomes $y(t+k) = y(t) + \lambda k\, y(t)$. For all ODE methods (not just as here for FE), the variables λ and k will at this point only enter in the combination $\xi = \lambda k$. Thus $y(t+k) = (1+\xi)y(t)$, and $y(t+nk) = (1+\xi)^n y(t)$. The condition for no growing solutions therefore becomes $|1+\xi| \leq 1$, i.e. a circle in a complex ξ-plane centered at $\xi = -1$ and with radius 1, shown by the label "1" in Figure 16.4.1(a). □

In the case of Example 65 , we have $\lambda = -10$. The stability domain for FE tells that we have no-growth if $-2 \leq \xi \leq 0$. Given that $\xi = \lambda k$, we have thus again arrived at the previous condition $k \leq 1/5$.

Even for real-valued ODEs, we are also interested in complex values of ξ. If we for example want to solve a system of ODEs such as

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix}' = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

(which can arise directly in applications or as the result of rewriting a higher order equation as a first order system; in this case $y'' + y = 0$), a simple change of variable decouples the two equations, and the eigenvalues to consider are $\lambda_{1,2} = \pm i$.

EXAMPLE 66. Calculate the stability domain for AB2.

Applying $y(t+k) = y(t) + k\,[\frac{3}{2}y'(t) - \frac{1}{2}y'(t-k)]$ to $y' = \lambda y$ and substituting $\xi = \lambda k$ gives

$$y(t+k) - (1 + \tfrac{3}{2}\xi)\, y(t) + \tfrac{1}{2}\,\xi\, y(t-k) = 0. \tag{16.3}$$

This is a 3-term linear recursion relation, which is most easily solved via its characteristic equation

$$r^2 - (1 + \tfrac{3}{2}\xi)\, r + \tfrac{1}{2}\,\xi = 0. \tag{16.4}$$

We now want to plot, in the complex ξ-plane, the domain with the property that both roots r_1 and r_2 to (16.4) lie inside or on the unit circle - the condition for (16.3) not to have growing solutions. Trying to solve the quadratic equation (16.4) for r turns out to be a bad idea. A much better one is to realize that the boundary of the stability domain (in the ξ-plane) must be characterized by one root r to (16.4) being on the edge of the unit circle. Thus, we solve (16.4) for ξ

$$\xi = \frac{2r\,(r-1)}{3r-1} \tag{16.5}$$

and obtain the stability domain by letting $r = e^{i\theta}$ and plotting ξ as θ runs from 0 to 2π. In Matlab, the complete code for this becomes

```
r = exp(i*linspace(0,2*pi)); plot(2*r.*(r-1)./(3*r-1));
```

The resulting domain is labeled "2" Figure 16.4.1(a). □

The procedure in Example 66 for AB2 works for all LM methods. In case of AB3, we get in place of (16.5)

$$\xi = \frac{12r^2\,(r-1)}{23r^2 - 16r + 5}$$

and for AB4

$$\xi = \frac{24r^3\,(r-1)}{55r^3 - 59r^2 + 37r - 9}.$$

At this point (of AB4) enters a minor complication in that the curve that is traced out in the ξ-plane intersects itself. The curve marks where one root to the characteristic equation changes from $|r| < 1$ to $|r| > 1$. It can happen that another root is larger than one, so that both sides of the traced curve are outside the stability domain. In

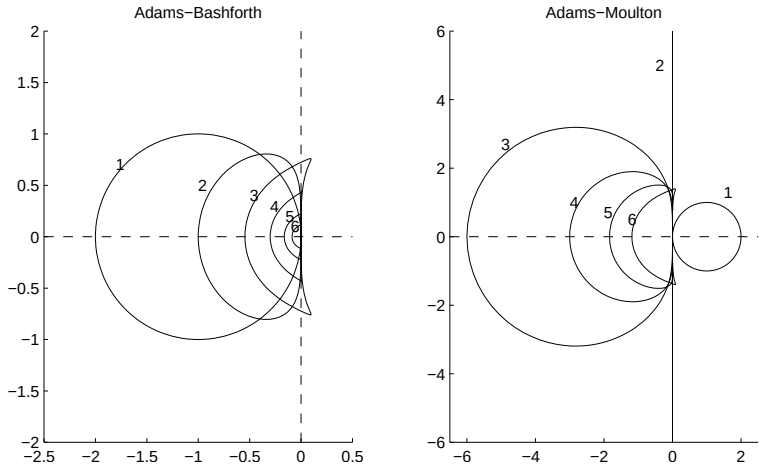


FIGURE 16.4.1. Stability domains for (a) AB and (b) AM methods of orders 1-6. The stability domains in all cases include the regions immediately to the left of the origin, i.e. for AB1 it is the domain outside the circle $|1-\xi| = 1$, and for AB2 the left halfplane. In all other cases, the regions are bounded. Note that the scale differs by a factor of three between the two pictures.

such cases of intersecting loops, one simply chooses any ξ-value inside the loop, and solves for all the roots - that will tell if the loop is part of the stability region or not.

Figures 16.4.1(a) and (b) summarize the stability domains for AB and AM methods of increasing orders.

In both cases, the stability domains get smaller when the order increases. For each fixed order, the domain for the AM method is much larger than the one for the AB method. Dependent on the ODE that is solved, this advantage may outweigh the disadvantage of the AM methods being implicit.

In some important applications discussed in Chapter ??, all eigenvalues of the ODE will be purely imaginary, and it is then essential to know if a method's stability domain will include an interval of the imaginary axis around the origin. For AB methods, this turns out to happen if the order is {3,4}, {7,8}, {11,12}, etc. and the opposite holds for AM methods, i.e. there is some imaginary axis coverage for orders {1,2}, {5,6}, {9,10}, etc. [..].

16.4.3.3. *Stability domains for BD methods.* Example 4 in Section 16.3 provided an example of a BD scheme. The schemes BD1-BD6 differ only in the number of

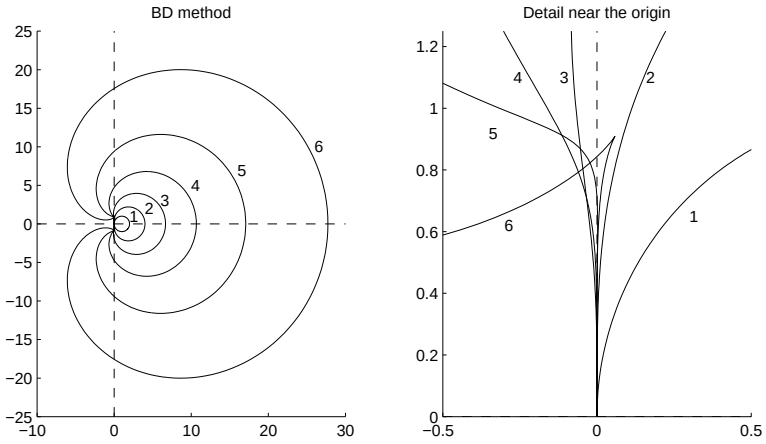


FIGURE 16.4.2. Stability domains of BD methods of orders 1-6. The domains are in all cases on the outside of the shown boundaries: (a) the complete boundaries (b) detailed boundary structures near the origin.

back levels that are employed for y. Figure 16.4.2 a show their stability domains and part b gives a more detailed picture near the origin.

The stability domains are this time what falls outside (not inside) the closed curves. For all these schemes, the whole negative real axis is included in the domains. For BD schemes of orders $p = 7$ and above, not only does this property get lost; the schemes will also fail the root condition.

16.4.4. The Dahlquist barriers. The concepts of stability (root condition) and of stability domain are not entirely unrelated for LM methods, as shown by the Dahlquist stability barriers.

First barrier: The order of accuracy p of a stable s-step LM formula satisfies

$$p \;\leq\; \begin{cases} s+2 & \text{if } s \text{ is even} \\ s+1 & \text{if } s \text{ is odd} \\ s & \text{if the formula is explicit} \end{cases} .$$

For example, the AB3 scheme illustrated in Example 1 is explicit, and features $p = s = 3$ (note that s here is not the same s as in the Mathematica code to generate LM schemes). From this first barrier follows that any attempt to increase the order of accuracy by introducing further entries also down the left stencil column (as in

Example 3) must cause the root condition to become violated. For explicit LM schemes, the AB class features as high orders of accuracy as is possible, given any value of s.

It is natural to want a method to include the whole negative half plane ($\text{Re}\ \xi < 0$) in the domain, since the analytical solutions to $y' = \lambda y$ decay when $\text{Re}\ \lambda < 0$. Such a scheme is called A-stable.

Second barrier: The following two results hold:

(1) An explicit LM method can never be A-stable, and
(2) An implicit A-stable method can be at most second order accurate.

The AM1 (also described as BE and BD1), AM2 and BD2 methods are all examples of A-stable schemes. In most applications, A-stability is a more restrictive requirement than what is needed, cf. the discussion about stiff systems in Section 16.8.

16.5. Predictor-corrector methods.

An interesting idea for combining the key strength of AB methods (being explicit) with that of AM methods (having much larger stability domains) is to use them in succession as a *predictor-corrector* pair. One of several possible strategies to obtain such a combined method of order p is the following:

(1) use AB of order $p-1$ to get a predicted value at the new time level: $\widehat{y}(t+k)$,
(2) use $\widehat{y}(t+k)$ to calculate a predicted value $\widehat{f}(t+k, \widehat{y}(t+k))$,
(3) use $\widehat{f}(t+k, \widehat{y}(t+k))$ in the RHS of an AM scheme of order p, to obtain $y(t+k)$.
(4) calculate a corrected value of $f(t+k, y(t+k))$ (to use for subsequent time steps).

This combined scheme can be shown to be accurate of order p, is entirely explicit, and will have considerably larger stability domain than corresponding AB methods, as seen by comparing Figure 16.4.1(a) with Figure 16.5.1(a).
It can be shown that all schemes of this type with $p \geq 3$ will have some imaginary axis coverage near the origin. A disadvantage with the schemes is that they require two (rather than one) function evaluations per time step.

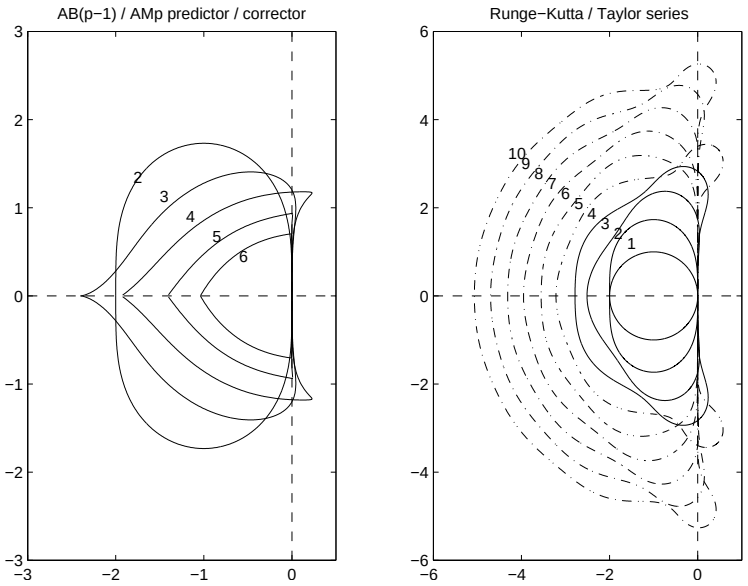


FIGURE 16.5.1. Stability domains for (a) AB(p-1)/AMp predictor/corrector methods for $p = 2, 3, \ldots, 6$. (b) Solid curves: RK methods with $s = p = 1, 2, 3, 4$; Solid and dash-dot curves: TS methods of orders $p = 1, 2, \ldots, 10$. Note that the scale differs by a factor of two between the two plots.

The method to calculate a predictor/corrector scheme's stability domain is very similar to that for a regular LM scheme:

EXAMPLE 67. Calculate the stability domain for the AB2/AM3 scheme.
As always when calculating stability domains, we consider the ODE $y' = \lambda y$. The two steps can be written

$$\begin{cases} \widehat{y}(t+k) = y(t) + \lambda k \, [\frac{3}{2}y(t) - \frac{1}{2}y(t-k)] \\ y(t+k) = y(t) + \lambda k \, [\frac{5}{12}\widehat{y}(t+k) + \frac{2}{3}y(t) - \frac{1}{12}y(t-k)] \, . \end{cases}$$

After the substitutions of $\lambda k = \xi$ and of $\widehat{y}(t+k)$ from the first equation into the second one, we get a linear 3-term recursion relation with characteristic equation

$$r^2 - \left(1 + \tfrac{13}{12}\xi + \tfrac{5}{8}\xi^2\right) \, r + \left(\tfrac{1}{12}\xi + \tfrac{5}{24}\xi^2\right) = 0 \, . \tag{16.1}$$

To trace out the edge of the stability domain, we again let r run around the periphery of the unit circle, and follow how ξ then moves. Although we, for each r-value can

do this by solving a quadratic equation for ξ, an easier strategy is to start by noting that $r = 1$ always corresponds to $\xi = 0$. As we step along in r, we can use Newton's method to find the corresponding ξ-value (using the ξ-value from the last computed case as the start approximation for the next one). $\square$

16.6. Runge-Kutta (RK) methods.

Runge-Kutta methods are more convenient than LM methods in that one does not need any previous time levels to get started. The lowest order RK scheme is FE, which we now write as

$$\begin{array}{ll} s=1,\ p=1 & d^{(1)} = k\ f(t, y(t)) \\ & ---------- \\ & y(t+k) = y(t) + [d^{(1)}] \end{array}$$

For higher order RK methods, each time step is split into s internal *stages*, requiring one function evaluation each. The following are examples of methods of increasing numbers of stages s and of correspondingly increasing accuracies p:

$$\begin{array}{ll} s=2,\ p=2 & d^{(1)} = k\ f(t, y(t)) \\ & d^{(2)} = k\ f(t + \frac{k}{2}, y(t) + \frac{1}{2} d^{(1)}) \\ & ---------- \\ & y(t+k) = y(t) + [d^{(2)}] \end{array}$$

$$\begin{array}{ll} s=3,\ p=3 & d^{(1)} = k\ f(t, y(t)) \\ & d^{(2)} = k\ f(t + \frac{k}{3}, y(t) + \frac{1}{3} d^{(1)}) \\ & d^{(3)} = k\ f(t + \frac{2k}{3}, y(t) + \frac{2}{3} d^{(2)}) \\ & ---------- \\ & y(t+k) = y(t) + [\frac{1}{4} d^{(1)} + \frac{3}{4} d^{(3)}] \end{array}$$

The best known of all RK schemes is probably

$s = 4,\ p = 4$

$$
\begin{aligned}
d^{(1)} &= k\ f(t, y(t)) \\
d^{(2)} &= k\ f(t + \tfrac{k}{2}, y(t) + \tfrac{1}{2} d^{(1)}) \\
d^{(3)} &= k\ f(t + \tfrac{k}{2}, y(t) + \tfrac{1}{2} d^{(2)}) \\
d^{(4)} &= k\ f(t + k, y(t) + d^{(3)}) \\
&- - - - - - - - \\
y(t + k) &= y(t) + \tfrac{1}{6}[d^{(1)} + 2d^{(2)} + 2d^{(3)} + d^{(4)}]\ .
\end{aligned}
$$

Although there are many different RK schemes with the same values for s and p, it turns out that whenever $s = p$, the stability domains are described by the relation

$$r = \sum_{n=0}^{p} \frac{\xi^n}{n!}\ , \tag{16.1}$$

giving the domains shown by solid lines (numbered 1-4) in Figure 16.5.1(b). For $s = p = 1$, we recognize the domain for FE. In contrast to the AB and AM methods, the stability domains increase in size when the order increases. In particular, we can note that the domains for RK3 and RK4 cover sizeable sections of the imaginary axis. For RK methods of higher orders than $p = 4$, a few complications arise:

(1) computation of coefficients within the stages becomes extremely complicated (however, schemes with orders up to around $p = 10$ have been tabulated in the literature, and need not be re-derived by users),
(2) there are no longer any schemes with $s = p$, only with $s > p$,
(3) the stability domains become dependent on the RK coefficients (i.e. no longer dependent only on $p = s$), and
(4) the accuracy may be lower for systems of PDEs than for scalar ODEs.

Numerical ODE packages that are based on RK methods usually employ RK variations which provide not only a value at the new time level, but also an error estimate for each step. The step lengths can then be adjusted automatically in order to meet a specified error tolerance.

Apart from explicit RK methods, there are also implicit RK methods. These can reach order $2s$ with only s stages, and can also have perfect stability domains - precisely the left half plane (matching where the solutions to the ODE $y' = \lambda y$

feature no growth). However, the cost required for solving the systems of equations that arise often make them less practical.

16.7. Taylor series (TS) methods.

The TS approach is in some cases the most powerful technique available, but it is somewhat limited in that

- The right hand side of the ODE (16.1) must be composed only of analytic functions (e.g. of powers, fractions, exponentials, and trigonometric functions), and
- Quite advanced software is needed if one wants to automatically translate (compile) an ODE into executable code. In contrast, for linear multistep and Runge-Kutta methods, the ODE solver itself does not need to be altered between different ODEs.

Recalling from (16.1) that forward Euler corresponds to truncating after the second term in a Taylor series

$$y(t+k) = y(t) + k\, y'(t) + \frac{k^2}{2!}y''(t) + \frac{k^3}{3!}y'''(t) + \frac{k^4}{4!}y''''(t) + \dots \,, \tag{16.1}$$

the idea is simply to truncate at a much later level. That requires that we somehow can determine higher derivatives of y. Surprisingly many numerical analysis text books mention and then dismiss this approach after considering only an inefficient way to go about this - repeated differentiation. Given $y'(t) = f(t, y(t))$, differentiation (based on the chain rule) gives

$$\begin{aligned}
y'' &= f\,\tfrac{\partial f}{\partial y} + \tfrac{\partial f}{\partial t} \\
y''' &= f^2\tfrac{\partial^2 f}{\partial y^2} + f\,\left\{2\tfrac{\partial^2 f}{\partial t \partial y} + \left(\tfrac{\partial f}{\partial y}\right)^2\right\} + \tfrac{\partial f}{\partial t}\cdot\tfrac{\partial f}{\partial y} + \tfrac{\partial^2 f}{\partial t^2} \\
y'''' &= f^3\tfrac{\partial^3 f}{\partial y^3} + f^2\left\{3\tfrac{\partial^3 f}{\partial t \partial y^2} + 4\tfrac{\partial f}{\partial y}\cdot\tfrac{\partial^2 f}{\partial y^2}\right\} + \tfrac{\partial f}{\partial t}\cdot\left(\tfrac{\partial f}{\partial y}\right)^2 + \tfrac{\partial^3 f}{\partial t^3} + 3\tfrac{\partial f}{\partial t}\cdot\tfrac{\partial^2 f}{\partial t \partial y} + \tfrac{\partial^2 f}{\partial t^2}\cdot\tfrac{\partial f}{\partial y} + \\
&\quad + f\,\left\{\left(\tfrac{\partial f}{\partial y}\right)^3 + 5\tfrac{\partial^2 f}{\partial t \partial y}\cdot\tfrac{\partial f}{\partial y} + 3\tfrac{\partial^3 f}{\partial t^2 \partial y} + 3\tfrac{\partial f}{\partial t}\cdot\tfrac{\partial^2 f}{\partial y^2}\right\} \\
\cdots &\quad \text{etc.}
\end{aligned}$$

Not only does the number of terms increase very rapidly, the individual derivatives also get very complex quickly for all but the simplest functions $f(t, y)$. It is much

more efficient to determine the expansion coefficients recursively. This is best explained by means of an example. Repeating example (16.2) here for convenience,

$$y' = t^2 + y^2, \quad y(0) = y_0, \tag{16.2}$$

we substitute the Taylor series centered at $t = 0$,

$$y(t) = \sum_{n=0}^{N} a_n t^n \tag{16.3}$$

into (16.2),

$$\sum_{n=0}^{N-1} (n+1) a_{n+1} t^n = t^2 + \sum_{n=0}^{2N} \sum_{s=0}^{n} a_s a_{n-s} t^n.$$

Matching the different powers of t we get the recursion relation

$$\begin{aligned}
t^0: \quad a_1 &= a_0^2 \\
t^1: \quad a_2 &= a_0 a_1 \\
t^2: \quad a_3 &= \frac{1}{3}\left(1 + \sum_{j=0}^{2} a_j a_{2-j}\right) \\
&\vdots \\
t^n: \quad a_{n+1} &= \frac{1}{n+1} \sum_{j=0}^{n} a_j a_{n-j},
\end{aligned}$$

with $a_0 = y_0$, the initial value. Once the Taylor expansion coefficients are known, up to any desired order, an approximation of $y(t)$ can now be calculated from (16.3) for any value of t, say t_1, within the radius of convergence of the Taylor series. Once an approximation of $y(t_1)$ is known, the process is repeated by Taylor expanding at $t = t_1$.

Of course it is not necessary to do the above by hand. The following brief Mathematica script illustrates how, if given a value y at time t, one can generate the coefficients a_1, a_2, ... in the expansion

$$y(t+k) = y(t) + a_1 k + a_2 k^2 + a_3 k^3 + \ldots \tag{16.4}$$

In the case of $n = 8$ and the ODE (16.2), the script would be

```
n=8;(* Specify number of coefficients *)y[k]:=y+Sum[a[i] ki,{i,1,n}]+O[k](n+1)
f[t,y]:=t^2+y^2(* Specify the RHS of the ODE *) LogicalExpand[y'[k]==f[t+k,y[k]]]
```

giving the output

```
t^2+y^2-a[1]==0 && 2t+2ya[1]-2a[2]==0 && 1+a[1]^2+2y a[2]-3a[3]==0 &&
2a[1]a[2]+2ya[3]-4a[4]==0 && a[2]^2+2a[1]a[3]+2ya[4]-5a[5]==0 &&
2a[2]a[3]+2a[1]a[4]+2ya[5]-6a[6]==0 &&
a[3]^2+2a[2]a[4]+2a[1]a[5]+2y a[6]-7a[7]==0 &&
2a[3]a[4]+2a[2]a[5]+2a[1]a[6]+2ya[7]-8a[8]==0
```

again become simple algebraic recursions that explicitly provide the successive coefficients. A similar approach to obtain the same recursions starts with considering

$$\frac{dy(t+k)}{dk} = f(t+k, y(t+k)).$$

Every time a truncated version of (16.4) is substituted into the RHS $f(t+k, y(t+k))$ and is integrated with respect to k, we gain (at least) one more term in (16.4). By always truncating to no more terms than what are known to be correct, the algebra remains moderate and is well suited for automation without the need of general symbolic packages such as Mathematica. In the TS method, the recursion relations are generated as a preprocessing step, and are then compiled into the actual solution algorithm. Like for the other numerical ODE approaches, it is usually most convenient to rely on library software when using the TS approach.

The stability domains for the TS methods are also described by (16.1). For RK methods, this formula was valid only when $s = p = 1, 2, 3, 4$. The TS method continues this same formula up to any value of p. The dashed curves in Figure 16.5.1(b) show how the domains continue to grow as p increases.

One intuitive way to understand why TS methods have so much larger stability domains (and also are much more accurate) than AB or AM methods of corresponding orders is to note that the latter pick up their 'extra' information from very old and 'obsolete' back values of y', essentially relying on the notoriously ill-conditioned concept of using one-sided high order polynomial approximations (cf. the discussion of the Runge phenomenon in Section 12.4). In contrast, a TS method increases the

order by means of extracting much more analytic information from the ODE itself, right at the most recent time position.

The following example illustrates a remarkable property of Taylor methods.

EXAMPLE 68. Solve

$$y' = y^2, \;\; y(0) = 1.$$

The analytical solution is $y(t) = \frac{1}{1-t}$, it therefore has a simple pole at $t = 1$ on the real axis. Let us see how the TS methods deals with this singularity. Substituting the Taylor expansion (16.3), we find the recursion relation,

$$\begin{aligned} a_0 &= 1 \\ a_1 &= a_0^2 \\ &\vdots \\ a_{n+1} &= \frac{1}{n+1} \sum_{j=0}^{n} a_j a_{n-j}. \end{aligned}$$

It should not be hard to show that the solution is given by $a_n = 1$, for all n. This is a remarkable, as should become clear if we convert it to a Padé approximation, as investigated in Exercise 25. Thus remarkably, the TS method returns the exact analytical solution, despite the presence of a pole. This is a point worth stressing: TS methods together with Padé approximations are eminently suitable for capturing poles in the complex plane, a property not shared by any of the polynomial-based methods.

The next example is one of a family of six equations, the Painlevé equations, that is all about the structure of their poles in the complex plane. It is not possible to go into any details here but one of their distinguishing properties is that their solutions all have only poles or essential singularities in the complex plane, no branch cuts are present. One of the main outstanding numerical issues was how to calculate numerical solutions in the presence of these singularities. The previous example suggests that TS methods, together with Padé may be suitable. Let us therefore calculate the Taylor expansion coefficients for the first Painlevé equation.

n	0	1	2	3	4	5	6	7	8	9
a_n	0.0000	0.5000	0.0000	0.1667	0.0000	0.1250	0.0000	0.0333	0.0000	0.0164

TABLE 1. Taylor expansion coefficients for Painlevé I.

EXAMPLE 69. Solve the first Painlevé equation, $\mathrm{P_I}$,

$$y'' = 6y^2 + t, \quad y(0) = 0, \quad y'(0) = 0.5, \tag{16.5}$$

numerically using Taylor's method. Substituting (16.3), and matching different powers of t, we find the recursion relations

$$\begin{aligned} t^0: \quad a_2 &= 3a_0^2 \\ t^1: \quad a_3 &= 2a_0a_1 + \frac{1}{6} \\ &\vdots \\ t^n: \quad a_{n+2} &= \frac{6}{(n+2)(n+1)} \sum_{j=0}^{n} a_j a_{n-j}. \end{aligned}$$

From the initial values $a_0 = 0, \quad a_1 = 0.5$, the numerical values for the Taylor expansion coefficients are given in Table 1.

We leave it as an exercise to convert this expansion into a Padé approximation, as explained in Section 10.3.

16.8. Stiff ODEs.

A *stiff* system of ODEs is one in which some processes happen extremely fast while others are much slower. A typical example could be a chemical process with different coupled reactions. Generalizing the linearization procedure described in Section 16.4.3.1 to a system of ODEs leads us to consider

$$\underline{v}'(t) \approx A\, \underline{v}(t) + \underline{b}.$$

If the eigenvalues λ of A differ in size by several orders of magnitude, especially if some are negative and very large in magnitude, we have a stiff system. In order not to be forced to use an extremely small time step (given that $\xi = \lambda k$ for all the λ have to fall within the stability domain of the ODE solver), it is essential that the

stability domain extends far out to the left in the complex ξ-plane. Of the methods that we have discussed, the BD methods excel in this regard.

In the context of solving wave-type PDEs, we will come across cases where the λ's will be lying far out along the imaginary axis. Once the eigenvalue character of an ODE or (more commonly) of a system of ODEs has been assessed, comparisons with different methods' stability domains provide a very good guide for choosing a suitable method.

CHAPTER 17

FINITE DIFFERENCE METHODS FOR PDE's

17.1. Introduction.

Chapter to be written.

CHAPTER 18

OPTIMIZATION: LINE SEARCH TECHNIQUES

18.1. Introduction.

If nature is left to run its course it tends to find the best or optimal solutions for its problems. In the Western Cape, South Africa, several Gladiolus species for example, produce heavy scent in the evening but none during the day, so as not to waste precious energy during day time when its pollinator moth is not active. In physics equilibrium solutions correspond to minimum energy solutions, and scientists and engineers in general are constantly faced with find solutions that minimizes some objective function. The generalized inverse of a general linear system of algebraic solutions discussed in Section 11.5 will soon be interpreted as an optimization problem with linear constraints, in which case it appears in the guise of Lagrange multipliers.

Optimization can be constrained or unconstrained. In the former it means that additional constraints are imposed that the solution has to satisfy. One popular method for incorporating the constraints is Lagrange multipliers, discussed in the next section. Our discussion differs from most other in that it is based on the Singular Value Decomposition (SVD), as alluded to above. From there we proceed to discuss the so-called line-search methods. These are all iterative methods; given an estimate of the solution (minimizing the objective function), one systematically improves on the estimate until convergence. In the case of line-search methods, the improvement consists of two general steps. First an approximate direction from the current estimate to the optimum is determined. One then moves in this direction towards the optimum. The second step is to determine how far one has to move in this direction, see for example **[5, 14]**.

The different line-search methods differ in the ways in which the directions and distances are determined. It is important to note that the directions are, for these methods, always determined by *local* information. This leads to a major complication

with line-search methods— they tend to get stuck in local optima. Starting from an initial guess the iterative improvements tend to proceed in directions of decreasing objective function values. If a point is reached where all directions lead to an increase in the objective function, i.e. a local minimum, the solution is stuck at this local minimum. Since the line-search directions are determined by local properties of the objective function, it is not possible to determine the position, or even the presence, of better (global) minima. These complications are addressed in the next chapter.

18.2. Lagrange Multipliers.

Suppose we are given a function, the objective function $f(\mathbf{x})$ where $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$ and we want to find the minimum value of f. More precisely, we are interested in finding $\mathbf{x}^\star$ that minimizes $f(\mathbf{x})$, i.e.

$$\mathbf{x}^\star = \arg\min_{\mathbf{x}} f(\mathbf{x}). \tag{18.1}$$

The well-known *necessary* condition for the extreme values is given by (assuming of course that the necessary partial derivatives are available),

$$\left.\frac{\partial f}{\partial x_j}\right|_{\mathbf{x}=\mathbf{x}^\star} = 0,\ j = 1, \ldots, n.$$

This means that in general one has to solve a nonlinear system of algebraic equations. Assuming that problem is tractable, one finds all the extreme values, and then pick the global optima.

The problem becomes more complicated if we put constraints on the solution space. Let us illustrate it with an example.

Example 70. Let $f(x, y) := x^2 + y^2$ and we are asked to find the minimum of f subject to $g(x, y) := x + y = 1$. In this case the two independent variables are connected through a constraint and it is no longer possible to find the extreme values by setting the partial derivatives of f equal to zero. We need to find the minimum of f on the line $x + y = 1$. Since $f(x, y) = x^2 + y^2$ is just the square of the distance from the origin, the minimum we are looking for is the point on the line closest to the origin, i.e. $x = \frac{1}{2} = y$, see Figure 18.2.1.

□

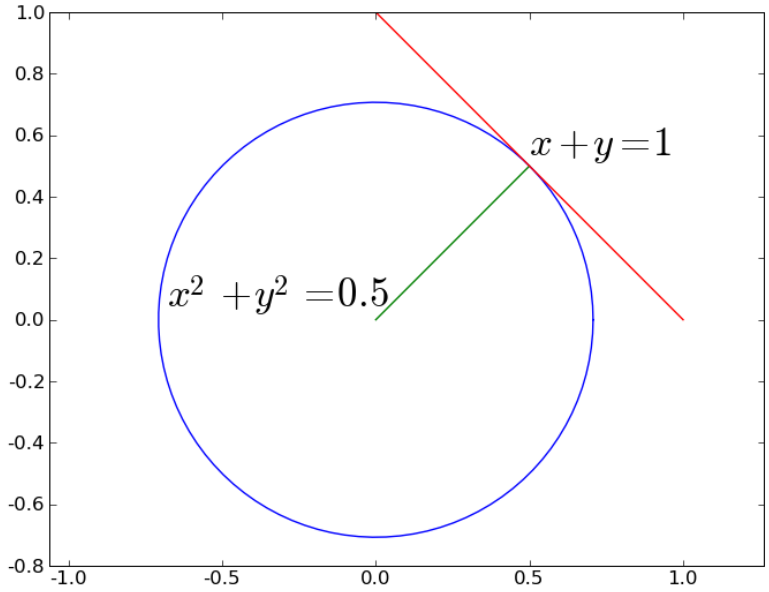


FIGURE 18.2.1. Minimizing $x^2 + y^2$ subject to $x + y = 1$.

This does not easily generalize to higher dimensions and a different point of view is required. Both $f(x, y) = c$ and $g(x, y) = 1$ define curves in the plane. In fact as we vary c a family of curves is defined by f where each member of the family is uniquely associated with c. We need to find that member of the family with the smallest value of c, subject to the constraint $g(x, y) = 1$. Imagine that we start with a very large value of c, corresponding to the circle with a very large radius in our example, intersecting the curve $g(x, y) = 1$ in two points. As we decrease the value of c, the circles start to shrink, and at some some stage we end up with a circle that just touches the curve $g(x, y) = 1$. The corresponding value of c is the minimum value for f, subject to the constraint. Out goal therefore is to find where the circle and curve touch. One way of doing this is to recognize that the two curves have a common tangent, where they touch, i.e. their gradients (perpendicular to the tangents) point in the same direction,

$$\boldsymbol{\nabla} f = -\lambda \boldsymbol{\nabla} g, \tag{18.2}$$

where λ is a scalar known as the Lagrange multiplier (the minus sign is for later convenience). We now have three equations for three unknowns x, y and λ, two equations from the Lagrange multiplier equation (18.2), and $g(x, y) = 1$. For our example these are

$$\begin{aligned} 2x &= -\lambda \\ 2y &= -\lambda \\ x + y &= 1. \end{aligned}$$

The first two equations tell us that $x = y$, and the third one that $x = \frac{1}{2} = y$. We are not really interested in λ but we find that $\lambda = -1$.

There is only one more observation to be made in order to generalize these ideas, and that is that (18.2) is equivalent to finding the minimum of

$$f(x, y) + \lambda \left(g(x, y) - 1 \right)$$

with respect to x, y and λ, where the constraint is recovered from minimizing with respect to λ.

EXAMPLE 71. For the problem in the previous Example 70, calculate the generalized solution of $x + y = 1$ using the SVD of Section 11.5.

Writing this as a system of linear equations (a system consisting of a single equation!),

$$\begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = 1,$$

we note that it admits an infinite number of solutions. As explain in Section 11.5 the SVD finds the solution that minimizes $|\mathbf{x}^2| = x^2 + y^2$. Thus the SVD gives the same answer as the Lagrange multiplier problem. Is this a coincidence or is there something more fundamental behind it?

□

A similar geometric argument that led to (18.2) can be used to generalize to higher dimensions. We prefer an alternative approach. We first give the general equations and then justify them using the Singular Value Decomposition (SVD). In

general the problem is, find We are looking for the minimum of

$$\mathbf{x}^\star = \arg\max_{\mathbf{x}} f(\mathbf{x}), \quad \mathbf{x} = [x_1, x_2, \ldots, x_n]^T,$$

subject to m constraints

$$g_j(\mathbf{x}) = 0, \ j = 1, \ldots, m, \tag{18.3}$$

assuming that $m < n$. Introducing the m Lagrange multipliers $\boldsymbol{\lambda} = [\lambda_1, \ldots, \lambda_m]^T$ we minimize

$$\mathcal{L}(\mathbf{x}) = f(\mathbf{x}) + \boldsymbol{\lambda}^T \mathbf{g}(\mathbf{x}), \tag{18.4}$$

where $\mathbf{g}(\mathbf{x}) = [g_1(\mathbf{x}), \ldots, g_m(\mathbf{x})]^T$. The necessary conditions for an extreme value,

$$\boldsymbol{\nabla}_{\boldsymbol{x}} \mathcal{L}(\mathbf{x}) = 0$$

gives us n equations in $m+n$ unknowns (the $\mathbf{x}$'s and $\boldsymbol{\lambda}$'s). The remaining m equations come from the constraints (18.3). Before we justify these equations using the SVD, let us do another example.

EXAMPLE 72. Find the minimum of

$$f(x, y, z) = x^2 + y^2 + z^2$$

subject to

$$\begin{aligned} x + y + z &= 1 \\ x - y - z &= 1. \end{aligned} \tag{18.5}$$

According to (18.4) we minimize

$$\mathcal{L}(x, y, z) = x^2 + y^2 + z^2 + \lambda_1 (x + y + z - 1) + \lambda_2 (x - y - z - 1)$$

subject to the constraints. From $\boldsymbol{\nabla}\mathcal{L} = 0$ we find

$$\begin{aligned} 2x + \lambda_1 + \lambda_2 &= 0 \\ 2y + \lambda_1 - \lambda_2 &= 0 \\ 2z + \lambda_2 - \lambda_2 &= 0. \end{aligned} \tag{18.6}$$

Solving for $x,\ y$ and z and substituting into the equations for the constraints (18.5) we get

$$\begin{aligned} -3\lambda_1 + \lambda_2 &= 2 \\ \lambda_1 - 3\lambda_2 &= 2, \end{aligned}$$

or $\lambda_1 = -1 = \lambda_2$. Substituting back into (18.6) we find $x = 1,\ y = 0$ and $z = 0$. The minimum value of f subject to the constraints is therefore $f = 1$.

□

Now for the justification, using the SVD. The constraints (18.3) form a $n - m$ dimensional manifold in $\mathbb{R}^n$. Suppose we have found the minimum of $f(\mathbf{x}^\star)$ on this manifold. Let us move a small distance on the manifold, away from the minimum, i.e. let $\mathbf{x} = \mathbf{x}^\star + \Delta\mathbf{x}$. Since we stay on the manifold it follows that $\mathbf{g}(\mathbf{x}) = 0 = \mathbf{g}(\mathbf{x}^\star)$ and from this follows that

$$J_g(\mathbf{x}^\star)\Delta\mathbf{x} = \mathbf{0},$$

where J_g is the Jacobian matrix of $\mathbf{g}$, i.e.

$$J_g(\mathbf{x}_m) = \left[\begin{array}{cccc} \frac{\partial g_1}{\partial x_1} & \frac{\partial g_1}{\partial x_2} & \cdots & \frac{\partial g_1}{\partial x_n} \\ \frac{\partial g_2}{\partial x_1} & \frac{\partial g_2}{\partial x_2} & \cdots & \frac{\partial g_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial g_m}{\partial x_1} & \frac{\partial g_m}{\partial x_2} & \cdots & \frac{\partial g_m}{\partial x_n} \end{array}\right]_{\mathbf{x}=\mathbf{x}^\star} = \left[\begin{array}{c} \nabla g_1(\mathbf{x}_m)^T \\ \nabla g_2(\mathbf{x}_m)^T \\ \vdots \\ \nabla g_m(\mathbf{x}_m)^T \end{array}\right]. \tag{18.7}$$

This means that $\Delta\mathbf{x}$ lies in the null space of the Jacobian matrix of $\mathbf{g}$. Since $\mathbf{x}^\star$ is a local minimum of $f(\mathbf{x})$ on the manifold, it follows that

$$\nabla f(\mathbf{x}^\star)\Delta\mathbf{x}^T = 0.$$

This tells us that $\nabla f(\mathbf{x}^\star)$ is orthogonal to the null space of $J_g(\mathbf{x}^\star)$, i.e. we can write $\nabla f(\mathbf{x}^\star)$ as a linear combination of the rows of (18.7) which is equivalent to minimizing (18.4).

EXAMPLE 73. *Entropy.*

Suppose we have N symbols $x_i,\ i = 1, \ldots, N$ that we want to send over a communication channel. Each symbol occurs with a probability $p(x_i)$. The question is how much information $h(x_i)$ do we receive if we receive a specific symbol x_i. If

$p(x_i) = 1$, then the symbol x_i appears with absolute certainty and there is no surprise; the information it carries should be zero, thus we require that $h(x_i) = 0$, whenever $p(x_i) = 1$. Now suppose we receive two unrelated symbols, x_i, x_j. The information gained by observing both of them should be $h(x_i, x_j) = h(x_i) + h(x_j)$. From these considerations it follows that a reasonable definition of the information content is given by

$$h(x) = -\log_2 p(x) \tag{18.8}$$

where the negative sign ensures that the information is non-negative.

The average amount of information received is given by

$$H(x) = -\sum_j p(x_j) \log_2 p(x_j) \tag{18.9}$$

and is known as the *entropy*. Calculating the maximum entropy is easy, using Lagrange Multipliers. Thus we want to maximize (18.9) subject to

$$\sum_j p(x_j) = 1. \tag{18.10}$$

Using Lagrange multipliers we maximize

$$-\sum_j p(x_j) \log_2 p(x_j) - \lambda \left(\sum_j p(x_j) - 1 \right)$$

subject to (18.10). Taking partial derivatives with respect to $p(x_j)$ we find

$$-\log_2 p(x_j) - 1 - \lambda = 0.$$

Since this expression is independent of j it means that $p(x_j)$ are the same for all j. From the constraint then follows that for maximum entropy,

$$p(x_j) = \frac{1}{N}.$$

□

EXAMPLE 74. Let A be a positive definite matrix. We are asking for the minimum of $\mathbf{x}^T A\mathbf{x}$ subject to $\mathbf{x}^T\mathbf{x} = 1$. Note that without the constraint the minimum value

would trivially be zero. Using Lagrange multipliers we need to minimize

$$\mathbf{x}^T A\mathbf{x} - \lambda\left(\mathbf{x}^T\mathbf{x} - 1\right) = 0.$$

Setting partial derivatives with respect to the x_j equal to zero gives,

$$A\mathbf{x} = \lambda\mathbf{x}.$$

The minimum values is just $\mathbf{x}^T A\mathbf{x} = \lambda$. Thus λ is the smallest eigenvalue of A.

□

EXAMPLE 75. *Sampson Correction.*

18.3. Line Search Methods.

In practice one repeatedly encounters the situation where a real valued objective function $f(\mathbf{x})$ that depends on the real variables $\mathbf{x} \in \mathbb{R}^n$ has to minimized with respect to $\mathbf{x}$. In mathematical terms, the problem becomes, find

$$\mathbf{x}^\star := \arg\min_{\mathbf{x}} f(\mathbf{x}), \tag{18.1}$$

which is of course the same as (18.1), but this time we are interested in the *unconstrained* problem. This formulation implies that we are interested in a *global* optimizer, i.e. we optimize over all values of $\mathbf{x}$. In practice this is not easy as the objective function may have several local minima in high dimensions in which case it can be hard to find the global optimum. In this section we concentrate on finding *local* optima, leaving the harder question of finding the global optimum for the next chapter (Chapter 19).

EXAMPLE 76. In order to fix the ideas, let us turn to an example discussed from a different point of view in Section 11.6. The linear least squares problem assumes that we are given data pairs, $(x_i, t_i),\ i = 1, \ldots, N$. The idea is to find a straight line $t = ax + b$ that best fits the given data. Thus we need to find values for a and b that are optimal in the sense that the straight line best fits the data. Accordingly we need to specify an objective function, and in this case a natural choice is

$$f(a, b) = \sum_{i=1}^{N} (t_i - ax_i - b)^2.$$

The minimum is easily obtained by setting $\frac{\partial f}{\partial a} = 0 = \frac{\partial f}{\partial b}$, or

$$\begin{aligned} \sum_{i=1}^{N} -2(t_i - ax_i - b)x_i &= 0 \\ \sum_{i=1}^{N} -2(t_i - ax_i - b) &= 0. \end{aligned}$$

Although this system can be easily solved, our real interest lies elsewhere.

□

18.3.1. Setting the scene. Let us state from the outset that in this chapter we always assume that the second partial derivatives of f are continuous. The first question we have to answer is how to recognize a local minimum. First we find the necessary conditions from Taylor's theorem in the form

$$f(\mathbf{x} + \mathbf{p}) = f(\mathbf{x}) + \nabla f(\mathbf{x} + t\mathbf{p})^T \mathbf{p}$$

where ∇f is the gradient of f, defined by

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_N} \end{bmatrix}$$

and $t \in (0, 1)$. Now suppose that $\mathbf{x}^\star$ is a local minimizer of $f(\mathbf{x})$, i.e. $f(\mathbf{x}^\star) \leq f(\mathbf{x})$ for all $\mathbf{x}$ in a neighborhood of $\mathbf{x}^\star$ and that $\nabla f(\mathbf{x}^\star) \neq 0$. Choosing $\mathbf{p} = -\epsilon \nabla f(\mathbf{x}^\star)$ we conclude from the continuity of the gradient operator that if we choose ϵ small enough, $\nabla f(\mathbf{x}^\star + t\mathbf{p})^T \mathbf{p} > 0$ in which case $f(\mathbf{x}^\star + t\mathbf{p}) < f(\mathbf{x}^\star)$, contradicting the assumption that $\mathbf{x}^\star$ is a local minimum.

A necessary condition for $\mathbf{x}^\star$ to be a local minimum is therefore

$$\nabla f(\mathbf{x}^\star) = \mathbf{0}.$$

In order to find sufficient condition we need one more term in the Taylor expansion

$$f(\mathbf{x} + \mathbf{p}) = f(\mathbf{x}) + \nabla f(\mathbf{x})^T \mathbf{p} + \frac{1}{2} \mathbf{p}^T \nabla^2 f(\mathbf{x} + t\mathbf{p}) \mathbf{p}$$

where $\nabla^2 f$ is the Hessian matrix of f, given by

$$\nabla^2 f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1 \partial x_1} & \cdots & \frac{\partial^2 f}{\partial x_N \partial x_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_1 \partial x_N} & \cdots & \frac{\partial^2 f}{\partial x_N \partial x_N} \end{bmatrix},$$

and $t \in (0,1)$. Since $\nabla f(\mathbf{x}^\star) = \mathbf{0}$, it follows, using the continuity of the Hessian matrix, that $f(\mathbf{x}^\star + \mathbf{p}) > f(\mathbf{x}^\star)$ for all sufficiently small $\mathbf{p}$ if and only if the Hessian matrix, $\nabla^2 f(\mathbf{x}^\star)$ is positive definite, written as

$$\nabla^2 f(\mathbf{x}^\star) > 0.$$

18.3.2. Line Search, the basic ideas. An iterative strategy will be followed in order to calculate the minimum $\mathbf{x}^\star$. Accordingly we select an initial estimate $\mathbf{x}_0$ and the algorithm then calculates successive approximations, $\mathbf{x}_i,\ i = 1, \ldots$ There are several ways how to do this and in this section we concentrate on a specific class, known as *Line Search* methods.

Imagine that your are standing on the side of a valley in dense fog. You know your current position by consulting your GPS. Let us say that your position, in some coordinate system, is given by two coordinates, $\mathbf{x}_0 \in \mathbb{R}^2$ that indicate your 'horizontal' position, and and another coordinate $f(\mathbf{x}_0)$, indicating your altitude. Using your GPS, you want to get to the bottom (the lowest point) of the valley (assuming it has one; it might be the bottom of a lake in which case you will get wet, but mathematicians don't care about that). What is a good strategy to get to the bottom? Since you can't see your surroundings it may be a good idea to set off in a downhill direction. You might decide to go straight down. In order to find this direction of steepest descent you may have to experiment a little by stepping a short way in different directions, but it should not be too hard to find it. In fact, it may be easier on the legs if you don't go straight down but at an angle. It should not matter too much, as long as you keep on going downhill. The next question is how far should you go in your chosen direction? Obviously you don't want to go so far as to start going up again. The moment that happens, it is a good time to change direction, and repeat the process all over again. If you have ever experienced getting off a mountain, you know you should be concerned about the possibility of

wondering off into a side valley, which can be disastrous. Remember you are in dense fog, and don't have a global view of your surroundings.

Line search methods are very much the mathematical equivalent of what we have just described.

Given an objective function $f(\mathbf{x})$ that we can evaluate for any value of $\mathbf{x}$ and a starting value $\mathbf{x}_0$ we find a direction $\mathbf{p}_0$, as well as a step length α_0. The new estimate estimate is then given by $\mathbf{x}_1 = \mathbf{x}_0 + \alpha_0 \mathbf{p}_0$, and in general the iterates are given by

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k, \quad k = 0, 1, \ldots, \tag{18.2}$$

with $\|\mathbf{p}_k\| = 1$. In order to move closer to the minimum, an obvious prerequisite for the choices of $\mathbf{p}_k$ and α_k is that $f(\mathbf{x}_{k+1}) < f(\mathbf{x}_k)$. This means that $\mathbf{p}_k$ should be chosen in a descending direction. Once we have chosen the direction, the step length α_k can be chosen by solving the one-dimensional minimization problem,

$$\alpha_k = \arg\min_{\alpha} \phi(\alpha), \tag{18.3}$$

where

$$\phi(\alpha) := f(\mathbf{x}_k + \alpha \mathbf{p}_k).$$

In practice this may be too expensive to solve exactly, and it may actually be better to use approximations that are cheap to calculate. We discuss a few strategies shortly.

Three popular choices for the directions are, steepest descent, Newton's method, and conjugate gradients, to be discussed in more detail shortly. For now only note that for $\mathbf{p}_k$ to be in a descent direction we require

$$[\nabla f_k]^T \mathbf{p}_k < 0,$$

where we have introduced the notation

$$f_k := f(\mathbf{x}_k)$$

Assuming we have identified a downhill search direction $\mathbf{p}_k$, we turn to addressing the question of estimating the step size, α_k.

18.3.3. Step size estimation. As alluded to above, solving equation (18.3) exactly may be too expensive, and an approximate strategy may be more appropriate.

In general the estimation proceeds in two steps. First an interval containing appropriate step sizes is determined. Secondly a good step size is selected from within the interval. So what is an appropriate step size? Surely we need a step size that reduces the value of the objective function, i.e. $\phi(\alpha) < f(\mathbf{x}_k)$. This in itself however, is not sufficient, the value of the objective function should be sufficiently reduced to ensure that the minimum is reached. Suppose, for example that the step sizes are chosen so small that $f(\mathbf{x}_k), \quad k = 0, 1, \ldots$ is reduced according to $f(\mathbf{x}_k) \propto 1/k$. In this case $f(\mathbf{x}_k) \longrightarrow 0$, no matter that the actual value might for instance be $f(\mathbf{x}^\star) = -1$. The first Wolfe condition ensures that there is a sufficient decrease in the objective function,

$$\phi(\alpha) \leq f_k + c_1 \alpha \left[\nabla f_k\right]^T \mathbf{p}_k, \tag{18.4}$$

which can be rewritten as

$$\phi(\alpha) \leq \phi(0) + c_1 \alpha \phi'(0),$$

for some constant $c_1 \in (0, 1)$. In practice c_1 is chosen to be quite small, say $c_1 = 10^{-4}$. Since the right hand side is a linear function in α with a small, negative gradient, it does allow undesirably small step sizes. Another criterion is needed to rule out too small step sizes,

$$\left[\nabla f(\mathbf{x}_k + \alpha_k \mathbf{p}_k)\right]^T \mathbf{p}_k \geq c_2 \left[\nabla f_k\right]^T \mathbf{p}_k, \tag{18.5}$$

for some $c_2 \in (c_2, 1)$. This can be rewritten as

$$\phi'(\alpha_k) \quad \geq \quad c_2 \phi'(0).$$

Thus the second Wolfe condition requires that the slope of ϕ at the chosen step size is c_2 times greater than the initial slope. If the slope is strongly negative one would like to move further in the chosen direction, so as to reduce the objective function more. On the other hand, if the slope is only slightly negative, it may be better not to continue to far in the chosen direction, but instead, terminate and start over in a more promising direction.

There is one more point that deserves attention. The second Wolfe condition (18.5) potentially allows the slope $\phi'(\alpha_k)$ to be strongly positive. This has the undesirable consequence that the step size may be chosen far from the optimal where $\phi'(\alpha) = 0$. The strong Wolfe condition prevents this,

$$|\phi'(\alpha_k)| \leq c_2 \, |\phi'(0)| \,, \tag{18.6}$$

with $c_2 \in (c_1, 1)$. Provided that the objective function is continuously differentiable and bounded from below, it is possible to show that there are step sizes satisfying the Wolfe conditions, including the strong Wolfe condition. Let us proceed to calculating appropriate step sizes.

Since the second Wolfe condition primarily prevents too small step sizes, one simple strategy to satisfy it is to start with a large step size, here and below denoted α_0, and then keep on reducing it until the first Wolfe condition is satisfied. If we reduce the step size with a constant factor ρ, the algorithm may look as follows,

Algorithm 1:
Choose α_0, $\rho \in (0, 1)$, $c \in (0, 1)$
Let $\alpha = \alpha_0$
repeat until $\phi(\alpha) < \phi(0) + c\alpha\phi'(0)$
set $\alpha = \rho\alpha$
end (repeat)
set $\alpha_k = \alpha$

Note that this algorithm requires the gradient of the objective function. Fortunately it needs to be calculated only once per iteration in order to estimate the step size.

Let us consider a second algorithm based on interpolation. The idea is again to pick a rather large initial step size and then reduce it by calculating the minimum of an interpolant of the objective function. The initial estimate is again denoted by α_0, and the improved value, α_1. One then proceeds as follows.

Given the initial guess α_0, test to see whether it satisfies the first Wolfe condition,

$$\phi(\alpha_0) \leq \phi(0) + c_1\alpha_0\phi'(0).$$

If it satisfies the condition the search is terminated and we set $\alpha_k = \alpha_0$. If not, we form the quadratic interpolant $\phi_2(\alpha)$ of $\phi(\alpha)$ by interpolating the values, $\phi(\alpha_0)$, $\phi(0)$ and $\phi'(0)$. Thus

$$\phi_2(\alpha) = \left(\frac{\phi(\alpha_0) - \phi(0) - \alpha_0 \phi'(0)}{\alpha_0^2} \right) \alpha^2 + \phi'(0)\alpha + \phi(0).$$

The minimizer of $\phi_2(\alpha)$ is given by

$$\alpha_1 = \frac{\phi'(0)\alpha_0^2}{2\left[\phi(\alpha_0) - \phi(0) - \alpha_0\phi'(0)\right]}.$$

If α_1 satisfies the second Wolfe condition, it is assigned to α_k and the search is terminated. If not, we form a cubic interpolant $\phi_3(\alpha)$ from the four pieces of information, $\phi(\alpha_1)$, $\phi(\alpha_0)$, $\phi(0)$, and $\phi'(0)$. Writing

$$\phi_3(\alpha) = a\alpha^3 + b\alpha^2 + \phi'(0)\alpha + \phi(0)$$

we need to determine a and b from the conditions, $\phi_3(\alpha_0) = \phi(\alpha_0)$, and $\phi_3(\alpha_1) = \phi(\alpha_1)$. This gives

$$\begin{bmatrix} a \\ b \end{bmatrix} = \frac{1}{\alpha_0^2\alpha_1^2(\alpha_1 - \alpha_0)} \begin{bmatrix} \alpha_0^2 & -\alpha_1^2 \\ -\alpha_0^3 & \alpha_1^3 \end{bmatrix} \begin{bmatrix} \phi(\alpha_1) - \phi(0) - \phi'(0)\alpha_1 \\ \phi(\alpha_0) - \phi(0) - \phi'(0)\alpha_0 \end{bmatrix}.$$

By setting the derivative equal to zero the minimizer of $\phi_3(\alpha)$ is given by the larger root

$$\alpha_2 = \frac{-b + \sqrt{b^2 - 3a\phi'(0)}}{3a}.$$

If α_2 does not satisfy the Wolfe condition, the process can be repeated until a suitable value is obtained. Of course for higher order polynomials the polynomials and the minimizers should be calculated numerically. A useful safe-guard is to check whether the newly computed value α_i is either too close, or too far from its predecessor. In that case one can set $\alpha_i = \alpha_{i-1}/2$.

Algorithms also exist that use the strong Wolfe condition as termination condition. For details see for example Nocedal and Wright [**14**].

18.3.4. The search directions. We shall discuss three different choices for the search directions. The first one is the simplest, the direction of steepest descent. The

second one should be familiar already, Newton's method. The third one, conjugate gradient, deserves special attention and is postponed to the next section.

18.3.4.1. *Steepest descent.* For this choice we descend straight down the side of the valley, in the direction of steepest descent, i.e. we choose

$$\mathbf{p}_k = -\nabla f(\mathbf{x}_k).$$

With this choice the optimization scheme becomes,

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \nabla f_k \tag{18.7}$$

where the step size is chosen according to one of the strategies described above. There is just one additional issue that is worth pointing out. Since we do not normalize the search direction, one has to be careful how the step size is initialized at each iteration (18.7). A useful strategy is to assume that the first order change at the k-th iteration will be the same as that of the previous iteration. One can therefore initialize α_0 according to $\alpha_0 \|\nabla f_k\|^2 = \alpha_{k-1} \|\nabla f_{k-1}\|^2$, or

$$\alpha_0 = \frac{\alpha_{k-1} \|\nabla f_{k-1}\|^2}{\|\nabla f_k\|^2}.$$

Although following the direction of steepest descent is intuitively attractive, in practice convergence tends to be slow. This is best illustrated by an example that is interesting in its own right.

EXAMPLE 77. For A a symmetric, positive definite matrix, an objective function is given by,

$$f(\mathbf{x}) = \frac{1}{2}\mathbf{x}^T A\mathbf{x} - \mathbf{b}^T\mathbf{x}$$

for a given vector $\mathbf{b}$. The gradient of the objective function is given by

$$\nabla f(\mathbf{x}) = A\mathbf{x} - \mathbf{b}$$

and the minimizer is given by the unique solution of the linear system

$$A\mathbf{x} = \mathbf{b}.$$

The optimal step size minimizes

$$\begin{aligned}\phi(\alpha) &= f(\mathbf{x}_k - \alpha\nabla f_k),\\ &= \frac{1}{2}(\mathbf{x}_k - \alpha\nabla f_k)^T A(\mathbf{x}_k - \alpha\nabla f_k) - \mathbf{b}^T(\mathbf{x}_k - \alpha\nabla f_k).\end{aligned}$$

From $\phi'(\alpha) = 0$ (and some manipulation), follows that

$$\alpha_k = \frac{(\nabla f_k)^T \nabla f_k}{(\nabla f_k)^T A\nabla f_k}. \tag{18.8}$$

The steepest descent iteration is therefore given by

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \left(\frac{(\nabla f_k)^T \nabla f_k}{(\nabla f_k)^T A\nabla f_k}\right)\nabla f_k.$$

Since $\nabla f_k = A\mathbf{x}_k - \mathbf{b}$ we have easily calculate $\mathbf{x}_{k+1}$ in terms of $\mathbf{x}_k$.

In order to quantify the rate of convergence it is useful to introduce the weighted norm

$$\|\mathbf{x}\|_A^2 := \mathbf{x}^T A\mathbf{x}.$$

Note that

$$\begin{aligned}\frac{1}{2}\|\mathbf{x} - \mathbf{x}^\star\|_A^2 &= \frac{1}{2}(\mathbf{x} - \mathbf{x}^\star)^T A(\mathbf{x} - \mathbf{x}^\star)\\ &=_{\frac{1}{2}} \left(\mathbf{x}^T A\mathbf{x} - \mathbf{x}^{\star T}A\mathbf{x} - \mathbf{x}^T A\mathbf{x}^\star + \mathbf{x}^{\star T}A\mathbf{x}^\star\right)\\ &= \frac{1}{2}\left(\mathbf{x}^T(A\mathbf{x} - 2\mathbf{b}) + \mathbf{x}^{\star T}\mathbf{b}\right)\\ &= \frac{1}{2}\mathbf{x}^T A\mathbf{x} - \mathbf{x}^T\mathbf{b} - \frac{1}{2}\mathbf{x}^{\star T}\mathbf{b} + \mathbf{x}^{\star T}\mathbf{b}\\ &= f(\mathbf{x}) - f(\mathbf{x}^\star)\end{aligned}$$

since $A\mathbf{x}^\star = \mathbf{b}$.

In order to obtain the convergence result, we need to calculate $\|\mathbf{x}_{k+1} - \mathbf{x}^\star\|_A^2$. Making use of (18.7) and (18.8) a tedious calculation gives (see Nocedal and Wright)

$$\|\mathbf{x}_{k+1} - \mathbf{x}^\star\|_A^2 = K_k\|\mathbf{x}_k - \mathbf{x}^\star\|_A^2$$

where

$$K_k = 1 - \frac{\left(\nabla f_k^T\nabla f_k\right)^2}{\left(\nabla f_k^T A\nabla f_k\right)\left(\nabla f_k^T A^{-1}\nabla f_k\right)}.$$

This expression suggests linear convergence, but it is not easy to interpret K_k. A bound in terms of the eigenvalues of A is given by Luenberger, in which case the error satisfies,

$$\|\mathbf{x}_{k+1} - \mathbf{x}^\star\|_A^2 \leq \left(\frac{\lambda_N - \lambda_1}{\lambda_N + \lambda_1}\right)^2 \|\mathbf{x}_k - \mathbf{x}^\star\|_A^2$$

where $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_N$, are the eigenvalues of A. Note in particular that when all the eigenvalues are the same, i.e. $\lambda_1 = \lambda_N$ then convergence is in one step. If on the other hand, the condition number of A is large, i.e. $\lambda_N >> \lambda_1$, then convergence can be excruciatingly slow.

□

For general nonlinear objective functions the result is essentially the same.

18.3.4.2. *Newton's method.* Since we are essentially finding the zeros of the gradient of the objective function, Newton's method is likely to be particularly useful. The system of equations to be solved are,

$$\nabla f(\mathbf{x}) = \mathbf{0}$$

and a straightforward application of Newton's method yields the iterative scheme

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \left(\nabla^2 f_k\right)^{-1} \nabla f_k.$$

This is a gradient method with search direction given by

$$\mathbf{p}_k = -\left(\nabla^2 f_k\right)^{-1} \nabla f_k.$$

That it is indeed a descent direction follows easily. Since

$$\mathbf{p}_k^T \nabla f_k = -\nabla f_k^T \left(\nabla^2 f_k\right)^{-1} \nabla f_k,$$

and $(\nabla^2 f_k)$ is positive definite, it follows that $\mathbf{p}_k^T \nabla f_k < 0$.

Note that there is a natural step size associated with Newton's method, namely $\alpha_k = 1$. If it happens that a Newton step does not give a sufficient reduction in the value of the objective function, it is of course possible to use different step sizes. To get the full advantage of the second order convergence of Newton's method, that we'll describe below, it is a good idea to use a step size $\alpha_k = 1$ during the latter part of the iteration.

As in the case of one-dimensional problems, Newton's method is locally second order convergent. Consider,

$$\begin{aligned}\mathbf{x}_{k+1} - \mathbf{x}^\star &= \mathbf{x}_k + \mathbf{p}_k - \mathbf{x}^\star \\ &= \mathbf{x}_k - \mathbf{x}^\star - \left(\nabla^2 f_k\right)^{-1} \nabla f_k \\ &= \left\|\left(\nabla^2 f_k\right)\right\| \left(\nabla^2 f_k \left(\mathbf{x}_k - \mathbf{x}^\star\right) - \left(\nabla f_k - \nabla f_\star\right)\right).\end{aligned}$$

Since

$$\nabla f_k - \nabla f_\star = \int_0^1 \nabla^2 f\left(\mathbf{x}_k + t(\mathbf{x}^\star - \mathbf{x}_k)\right)(\mathbf{x}^\star - \mathbf{x}_k) dt$$

it follows that

$$\begin{aligned}\left\|\nabla^2 f_k \left(\mathbf{x}_k - \mathbf{x}^\star\right) - \left(\nabla f_k - \nabla f_\star\right)\right\| &= \left\|\int_0^1 \left[\nabla^2 f_k - \nabla^2 f\left(\mathbf{x}_k + t(\mathbf{x}^\star - \mathbf{x}_k)\right)\right](\mathbf{x}^\star - \mathbf{x}_k) dt\right\| \\ &\leq \int_0^1 \left\|\nabla^2 f(\mathbf{x}_k) - \nabla^2 f\left(\mathbf{x}_k + t(\mathbf{x}^\star - \mathbf{x}_k)\right)\right\| \left\|\mathbf{x}^\star - \mathbf{x}_k\right\| dt \\ &\leq \left\|\mathbf{x}^\star - \mathbf{x}_k\right\|^2 \int_0^1 Lt\, dt \\ &= \frac{1}{2} L \left\|\mathbf{x}^\star - \mathbf{x}_k\right\|^2\end{aligned}$$

where is a Lipschitz constant for the Hessian matrix,

$$\left\|\nabla^2 f(\mathbf{x}_k) - \nabla^2 f\left(\mathbf{x}_k + t(\mathbf{x}^\star - \mathbf{x}_k)\right)\right\| \leq L \left\|t(\mathbf{x}^\star - \mathbf{x}_k)\right\|.$$

Since we assume that $\nabla^2 f(\mathbf{x}^\star)$ is positive definite, hence non-singular, there is a radius around $\mathbf{x}^\star$ where $\nabla^2 f(\mathbf{x})$ is non-singular and can be bounded from above. We therefore have that

$$\begin{aligned}\left\|\mathbf{x}_{k+1} - \mathbf{x}^\star\right\| &\leq \frac{1}{2} L \left\|\left(\nabla^2 f_k\right)^{-1}\right\| \left\|\mathbf{x}^\star - \mathbf{x}_k\right\|^2 \\ &\leq \widetilde{L} \left\|\mathbf{x}^\star - \mathbf{x}_k\right\|^2.\end{aligned}$$

Thus, if we choose the initial position $\mathbf{x}_0$ close enough to the solution $\mathbf{x}^\star$, we have quadratic convergence for Newton's method.

18.3.4.3. *Quasi Newton methods.* The main drawback of using Newton's method is the fact that it requires the Hessian matrix of the objective function. This can be expensive to calculate, or not available at all. The idea is to replace the Hessian

matrix with some approximation that is also cheap to calculate. The search direction is therefore replaced by

$$\mathbf{p}_k = -B_k^{-1}\nabla f_k$$

where B is a symmetric positive definite replacement for the Hessian $\nabla^2 f_k$, and the optimization iteration becomes,

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k,$$

where the step size α_k is required to satisfy the Wolfe condition (18.5) or the strong Wolfe condition (18.6). The question is what is a suitable replacement for the Hessian matrix? A Taylor expansion gives,

$$\nabla f_{k+1} = \nabla f_k + \nabla^2 f_k \alpha_k \left(\mathbf{x}_{k+1} - \mathbf{x}_k\right) + O\left(\left\|\mathbf{x}_{k+1} - \mathbf{x}_k\right\|^2\right).$$

With $\mathbf{x}_{k+1}$ and $\mathbf{x}_k$ sufficiently close to $\mathbf{x}^\star$, the $O\left(\left\|\mathbf{x}_{k+1} - \mathbf{x}_k\right\|^2\right)$ term may be ignored, and an approximation of the Hessian is given by

$$\nabla f_{k+1} \approx \nabla f_k + \nabla^2 f_k \left(\mathbf{x}_{k+1} - \mathbf{x}_k\right).$$

The idea is to choose B_{k+1} such that it reflects this situation. We therefore impose the secant condition and choose B_{k+1} such that

$$B_{k+1}\left(\mathbf{x}_{k+1} - \mathbf{x}_k\right) = \nabla f_{k+1} - \nabla f_k.$$

Introducing the notation

$$\mathbf{s}_k = \left(\mathbf{x}_{k+1} - \mathbf{x}_k\right) = \alpha_k \mathbf{p}_k \text{ and } \mathbf{y}_k = \nabla f_{k+1} - \nabla f_k,$$

the secant condition is written as

$$B_{k+1}\mathbf{s}_k = \mathbf{y}_k. \tag{18.9}$$

Note that this still requires the inversion of B_{k+1} at each iteration. It would be a useful to work directly with its inverse, H_{k+1}, with corresponding secant condition given by,

$$H_{k+1}\mathbf{y}_k = \mathbf{s}_k. \tag{18.10}$$

Given $\mathbf{s}_k$ and $\mathbf{y}_k$ the problem is to find a suitable B_{k+1}or H_{k+1}. Let us do a parameter count. An $N \times N$ symmetric matrix has $\frac{1}{2}N(N+1)$ parameters. Positive definiteness

imposes another N conditions—all N eigenvalues need to be positive so that we are left with $\frac{1}{2}N(N-1)$ free parameters. The secant condition impose N more conditions so that we are left with $\frac{1}{2}N(N-3)$ free parameters. It is indeed the case that an infinite number of suitable replacements for the Hessian matrix can be obtained. Many have been proposed but one reigns supreme—the BFGS formula, named after its inventors, Broyden, Fletcher, Goldfarb and Shannon,

$$B_{k+1} = B_k - \frac{B_k \mathbf{s}_k \mathbf{s}_k^T B_k}{\mathbf{s}_k^T B_k \mathbf{s}_k} + \frac{\mathbf{y}_k \mathbf{y}_k^T}{\mathbf{y}_k^T \mathbf{s}_k}. \tag{18.11}$$

The BFGS formula may come as a surprise—B_{k+1} is in the form of a rank-two *update* of B_k. It does make sense however—it is easy to compute. The original idea is due to W.C. Davidon, an idea that eventually transformed optimization. Although already proposed in 1959, his paper was not accepted for publication until 1991.

A derivation of the BFGS formula will take us outside the scope of this book. Note however the following:

(1) A necessary condition for satisfying the secant condition is given by

$$0 < \mathbf{s}_k^T B_{k+1} \mathbf{s}_k = \mathbf{s}_k^T \mathbf{y}_k,$$

or

$$\mathbf{s}^T \mathbf{y} > 0.$$

This automatically satisfied if we impose the Wolfe conditions (18.5) or (18.6) on the step size.

(2) The inverse $H_{k+1} = B_{k+1}^{-1}$ is given by

$$H_{k+1} = \left(I - \rho_k \mathbf{s}_k \mathbf{y}^T\right) H_k \left(I - \rho_k \mathbf{y}_k \mathbf{s}_k^T\right).$$

In this case the optimization iteration becomes,

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k H_k \nabla f_k.$$

(3) Since both B_k and H_k are calculated using a recursion scheme, the initial values B_0 and H_0 need to be provided. One obvious restriction is that both should be symmetric positive definite. Unfortunately there are no further guidelines and a popular choice is to choose them as a user defined multiple

of the identity matrix. Fortunately the BFGS formula has very effective self-correcting properties. Even if H_k is an inaccurate estimate of the inverse Hessian, the step sizes tend to decrease while the Hessian approximation corrects itself.

(4) As alluded to above the step size is chosen so that it satisfies either the Wolfe condition (18.5) or the strong Wolfe condition (18.6). For the BFGS the step size iteration always uses the initial estimate $\alpha_0 = 1$. In this way the step sizes are forced to the ideal value $\alpha_k = 1$, as the optimization iteration approaches convergence.

(5) Because of the the approximation of the Hessian, the quadratic convergence of Newton's method is lost. The rate of convergence however, remains super-linear and therefore still significantly faster than say the steepest descent method.

EXERCISE 78. What is the geometric significance of the one-step convergence in Example 77, in case all the eigenvalues are equal?

EXERCISE 79. Assuming that B_k satisfies the secant condition, show that B_{k+1} also satisfies the secant condition in (18.11).

18.4. The conjugate gradient method

When we discussed the solution of systems of linear equations of the form

$$A\mathbf{x} = \mathbf{b} \tag{18.1}$$

in Chapter 11 we concentrated on Gaussian elimination, one of the so-called direct methods. This means that the algorithm terminates after a finite, fixed number of steps. In the case of Gaussian elimination the intermediate steps don't tell us much about the information and we need to wait for the solution until the algorithm terminates. Although iterative methods can terminate after a finite number of steps, each iteration brings us closer to the solution and it is possible to terminate whenever the desired accuracy is attained. The conjugate gradient method is the method of choice for large, positive definite systems. Since large, positive definite systems occur frequently in practice it is a method of considerable interest in its own right. In this

section, we also use it to introduce the main ideas for solving nonlinear optimization problems.

18.4.1. The linear conjugate gradient method. If A is an $n \times n$ symmetric positive definite in (18.1) its solution can be written as the equivalent optimization problem

$$\mathbf{x}^\star = \arg\min_{\mathbf{x}} \phi(\mathbf{x}) \text{ with } \phi(\mathbf{x}) := \frac{1}{2}\mathbf{x}^T A\mathbf{x} - \mathbf{b}^T\mathbf{x}. \tag{18.2}$$

The solution of the optimization problem is obtained by setting the gradient of $\phi(\mathbf{x})$ equal to zero, and since A is positive definite, it has a unique minimum. Accordingly, if define the residual,

$$\mathbf{r}(\mathbf{x}) := \boldsymbol{\nabla}\phi(\mathbf{x}) = A\mathbf{x} - \mathbf{b}, \tag{18.3}$$

then solving,

$$\mathbf{r}(\mathbf{x}) = \mathbf{0} = \boldsymbol{\nabla}\phi(\mathbf{x}),$$

is the same as solving (18.1).

The basic strategy is a line-search technique, i.e. we solve (18.2) by iterating

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k, \tag{18.4}$$

which is of course identical to the general line-search formula (18.2). Moreover, given $\mathbf{x}_k$ and $\mathbf{p}_k$, the optimal step-size α_k is given by

$$\alpha_k = \arg\min_{\alpha} \phi(\mathbf{x}_k + \alpha\mathbf{p}_k). \tag{18.5}$$

Unlike the general problem, in this case it is straightforward to solve for α_k,

$$\alpha_k = -\frac{\mathbf{r}_k^T\mathbf{p}_k}{\mathbf{p}_k^T A\mathbf{p}_k}. \tag{18.6}$$

The main difference with the general optimization problem, and the real magic, is in the choice of the search direction $\mathbf{p}_k$. In order to set the scene, let us work through a simplest possible example—we solve a diagonal system.

EXAMPLE 80. Solve

$$\begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{x} = \begin{bmatrix} 3 \\ 1 \end{bmatrix},$$

using line-search, with the step-size given by (18.6), and for search directions we choose the coordinate axes.

For this problem we minimize,

$$\phi(\mathbf{x}) = \frac{1}{2}\left(3x_1^2 + x_2^2\right) - (3x_1 + x_2)$$

with solution

$$\mathbf{x}^\star = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

Choosing

$$\mathbf{x}_0 = \begin{bmatrix} -0.5 \\ -1 \end{bmatrix}$$

and the first search direction along the x_1 axis,

$$\mathbf{p}_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix},$$

it follows that $\alpha_0 = 1.5$ and

$$\mathbf{x}_1 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}.$$

With the second search direction along the x_2 axis

$$\mathbf{p}_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix},$$

it now follows that

$$\mathbf{r}_1 = \begin{bmatrix} 0 \\ -2 \end{bmatrix}, \text{ and } \alpha_1 = 2.$$

Thus we arrive at the solution after exactly two iterations,

$$\mathbf{x}_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} = \mathbf{x}^\star.$$

The situation is illustrated in Figure 18.4.1.

We should note a few interesting features of this procedure, facts that will become even more significant shortly. First we should note that no matter what the initial value, the exact solution is obtained in no more than two steps. For an $n \times n$ diagonal

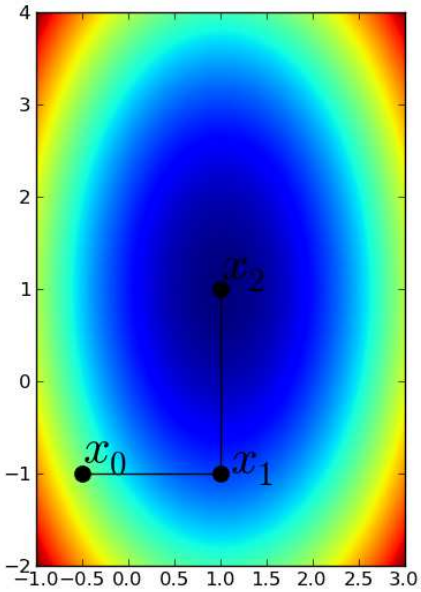


FIGURE 18.4.1. Search directions along the coordinate axes. For a diagonal matrix the solution is reached after two iterations.

matrix, exact convergence is obtained in no more than n steps. Of course this does not take roundoff errors into account. Secondly, after each step iteration one of the coordinates of the solution is obtained and it does not change with subsequent iterations. Finally, we note that the matrix A is not transformed during this process, unlike say. Gaussian elimination with pivoting. Therefore if A is sparse to begin with, there is no fill-in with during the iteration process.

Unfortunately it might appear that all these highly desirable features are lost for more general matrices, as illustrated in the following example.

EXAMPLE 81. Solve

$$\begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix} \mathbf{x} = \begin{bmatrix} 3 \\ 3 \end{bmatrix},$$

using line-search, with the step-size given by (18.6), and for search directions we choose the coordinate axes.

For this problem we minimize,

$$\phi(\mathbf{x}) = x_1^2 + x_1 x_2 + x_2^2 - 3\,(x_1 + x_2)$$

with solution

$$\mathbf{x}^\star = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

Choosing

$$\mathbf{x}_0 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

and the first search direction along the x_1 axis,

$$\mathbf{r}_0 = -\begin{bmatrix} 3 \\ 3 \end{bmatrix}, \quad \mathbf{p}_0 = \begin{bmatrix} 1 \\ 0 \end{bmatrix},$$

and it follows that $\alpha_0 = 1.5$ with

$$\mathbf{x}_1 = 1.5 \begin{bmatrix} 1 \\ 0 \end{bmatrix}.$$

With the second search direction along the x_2 axis

$$\mathbf{p}_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix},$$

it now follows that

$$\mathbf{r}_1 = -1.5 \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \text{ and } \alpha_1 = 0.75.$$

Thus we arrive at,

$$\mathbf{x}_2 = \begin{bmatrix} 1.5 \\ 0.75 \end{bmatrix}.$$

The situation is illustrated in Figure 18.4.2.

The most striking difference between Examples 80 and 81 is that in the former case, convergence is reached after no more than two iterations. The first question is how to recover this desirable behavior for the general symmetric positive definite matrix in (18.2). Example 80 suggests a transformation to diagonalize A,

$$\mathbf{x} = P\widehat{\mathbf{x}}$$

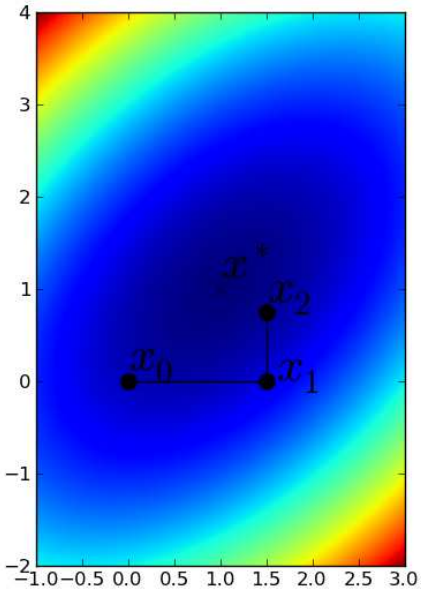


FIGURE 18.4.2. Using search directions along the coordinate axes. For general positive definite matrices, he solution is reached after an infinite number of iterations.

where P is invertible. Accordingly (18.2) becomes,

$$\widehat{\mathbf{x}}^\star = \arg\min_{\widehat{\mathbf{x}}} \phi(\widehat{\mathbf{x}}), \text{ with } \phi(\widehat{\mathbf{x}}) := \frac{1}{2}\widehat{\mathbf{x}}^T P^T A P \widehat{\mathbf{x}} - \mathbf{b}^T P \widehat{\mathbf{x}}.$$

Thus we are looking for n directions, $\mathbf{p}_0, \ldots, \mathbf{p}_{n-1}$, *conjugate* to A such that

$$\mathbf{p}_j^T A \mathbf{p}_k = 0, \quad j \neq k. \tag{18.7}$$

With $P = \begin{bmatrix} \mathbf{p}_0 & \cdots & \mathbf{p}_{n-1} \end{bmatrix}$ can therefore write,

$$S^T A S = \Lambda, \tag{18.8}$$

where Λ is a diagonal matrix.

We shall address the question of finding a set of conjugate directions below; for the moment note that one possibility is to choose the eigenvectors of A. This however, is

not computationally efficient and part of the magic of the conjugate gradient method lies in its construction of a set of conjugate directions.

Now suppose that somehow we are given a set of n conjugate directions $\mathbf{p}_0, \ldots, \mathbf{p}_{n-1}$ and a starting value $\mathbf{x}_0$, then the line search optimization (18.4), using the conjugate directions, becomes

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k \tag{18.9}$$

$$= \mathbf{x}_0 + \alpha_0 \mathbf{p}_0 + \cdots + \alpha_k \mathbf{p}_k, \tag{18.10}$$

with the step sizes α_k given by (18.6). We now show that using the conjugate directions, the iteration terminates within at most n steps.

Since the conjugate directions are linearly independent it is possible to write the error in the initial estimate as

$$\mathbf{x}^\star - \mathbf{x}_0 = \sigma_0 \mathbf{p}_0 + \cdots + \sigma_{n-1} \mathbf{p}_{n-1}.$$

Premultiplying this expression by $\mathbf{p}_k^T A$, and using the conjugacy condition, it follows immediately that,

$$\begin{aligned} \sigma_k &= \frac{\mathbf{p}_k^T A \left(\mathbf{x}^\star - \mathbf{x}_0\right)}{\mathbf{p}_k^T A \mathbf{p}_k} \\ &= -\frac{\mathbf{r}_0^T \mathbf{p}_k}{\mathbf{p}_k^T A \mathbf{p}_k}. \end{aligned}$$

Noting that we have arrived at a conjugate gradient recursion of the form (18.9), all that remains to be done is to show that $\sigma_k = \alpha_k$, and for that we have to show that $\mathbf{r}_0^T \mathbf{p}_k = \mathbf{r}_k^T \mathbf{p}_k$, or

$$\mathbf{p}_k^T A \left(\mathbf{x}^\star - \mathbf{x}_0\right) = \mathbf{p}_k^T A \left(\mathbf{x}^\star - \mathbf{x}_k\right).$$

Since (see (18.9)),

$$\mathbf{x}_k = \mathbf{x}_0 + \alpha_0 \mathbf{p}_0 + \cdots + \alpha_{k-1} \mathbf{p}_{k-1},$$

the same argument as above shows that

$$\mathbf{p}_k^T A \left(\mathbf{x}_k - \mathbf{x}_0\right) = 0.$$

Thus

$$\begin{aligned}\mathbf{p}_k^T A\left(\mathbf{x}^\star - \mathbf{x}_0\right) &= \mathbf{p}_k^T A\left(\mathbf{x}^\star - \mathbf{x}_k + \mathbf{x}_k - \mathbf{x}_0\right)\\ &= \mathbf{p}_k^T A\left(\mathbf{x}^\star - \mathbf{x}_k\right) + \mathbf{p}_k^T A\left(\mathbf{x}_k - \mathbf{x}_0\right)\\ &= \mathbf{p}_k^T A\left(\mathbf{x}^\star - \mathbf{x}_k\right),\end{aligned}$$

as required.

A key result in the conjugate gradient method is that the conjugate directions can be calculated recursively from the previous direction—it does not need all the previous directions. If we write $\mathbf{p}_k$ as a linear combination,

$$\mathbf{p}_k = -\mathbf{r}_k + \beta_k \mathbf{p}_{k-1}, \tag{18.11}$$

of the steepest descent direction $\mathbf{r}_k$ of ϕ, and the previous direction, $\mathbf{p}_k$ is conjugate to $\mathbf{p}_{k-1}$ provided,

$$\beta_k = \frac{\mathbf{p}_{k-1}^T A \mathbf{r}_k}{\mathbf{p}_{k-1}^T A \mathbf{p}_{k-1}}.$$

An induction argument shows that this indeed guarantees that $\mathbf{p}_k$ is conjugate to all previous directions, provided that the first direction is chosen as the steepest descent direction at the initial estimate $\mathbf{x}_0$, i.e. we choose $\mathbf{p}_0 = -\mathbf{r}_0$. Making use of the orthogonality conditions,

$$\mathbf{r}_k^T \mathbf{p}_i = 0, \quad i = 0, \ldots, k-1,$$

which follows using an induction argument, it is possible to simplify the equations for α_k and β_k,

$$\alpha_k = \frac{\mathbf{r}_k^T \mathbf{r}_k}{\mathbf{p}_k^T A \mathbf{p}_k}, \text{ and } \beta_{k+1} = \frac{\mathbf{r}_{k+1}^T \mathbf{r}_{k+1}}{\mathbf{r}_k^T \mathbf{r}_k}.$$

From the definition of the residual (18.3) written as

$$\mathbf{r}_k = A\mathbf{x}_k - \mathbf{b},$$

and the conjugate iteration (18.9) it follows that the residual is also given by a simple recursion,

$$\mathbf{r}_{k+1} = \mathbf{r}_k + \alpha_k A \mathbf{p}_k.$$

Thus we arrive at the following standard form of the conjugate gradient algorithm:

Algorithm:

$$
\begin{aligned}
&\text{Set } \mathbf{x}_0 = \mathbf{0},\ \mathbf{r}_0 = -\mathbf{b},\ \mathbf{p}_0 = -\mathbf{r}_0,\ k = 0\\
&\text{while } \mathbf{r}_k \neq \mathbf{0}\\
&\qquad \alpha_k = \frac{\mathbf{r}_k^T \mathbf{r}_k}{\mathbf{p}_k^T A \mathbf{p}_k}\\
&\qquad \mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{p}_k\\
&\qquad \mathbf{r}_{k+1} = \mathbf{r}_k + \alpha_k A \mathbf{p}_k\\
&\qquad \beta_{k+1} = \frac{\mathbf{r}_{k+1}^T \mathbf{r}_{k+1}}{\mathbf{r}_k^T \mathbf{r}_k}\\
&\qquad \mathbf{p}_{k+1} = -\mathbf{r}_{k+1} + \beta_{k+1} \mathbf{p}_k\\
&\qquad k = k + 1
\end{aligned}
$$

Several remarks are in order.

(1) The main cost at each iteration is a single matrix-vector multiplication $A\mathbf{p}_k$. In addition a single inner product $\mathbf{r}_{k+1}^T \mathbf{r}_{k+1}$ has to be calculated giving a total of $O(n^2 + n)$ operations. Since the algorithm terminates after at most n iterations (without taking roundoff error into account), we have $O(n^3 + n^2)$ operations. This is a gross overestimate, as will soon become clear. In any case, if n is large we would want to terminate early.

(2) The matrix A is not changes in any way during any stage of the iteration. This allows one to exploit any special structure of A to speed up the calculation of the matrix-vector product. This is easily achieved if A is sparse, for example.

(3) Each iteration provides a more accurate approximation of $\mathbf{x}^\star$. More explicitly, if we define the A-norm by $\|\mathbf{z}\|_A^2 := \mathbf{z}^T A \mathbf{z}$ then $\mathbf{x}_k$ is the unique point the Krylov subspace

$$\mathcal{K}_k := \operatorname{span}\left\{\mathbf{b}, A\mathbf{b}, \cdots, A^{k-1}\mathbf{b}\right\}$$

that minimizes $\|\mathbf{x}^\star - \mathbf{x}_k\|_A$. Moreover convergence is monotonic,

$$\|\mathbf{x}^\star - \mathbf{x}_k\|_A \leq \|\mathbf{x}^\star - \mathbf{x}_{k-1}\|_A .$$

(4) The general result describing the convergence of the conjugate gradient method is one of the truly beautiful theorems of numerical linear algebra,

$$\|\mathbf{x}^\star - \mathbf{x}_k\|_A \leq \inf_{p\in\mathbb{P}_k} \max_{\lambda\in\Lambda(A)} |p(\lambda)| \times \|\mathbf{x}^\star - \mathbf{x}_0\|_A \,, \tag{18.12}$$

where $\mathbb{P}_k$ is the set of all polynomials p with degree $\leq k$, and $\Lambda(A)$ denotes the spectrum of A. This means that we are looking for a polynomial $p(\lambda)$ whose maximum value over the eigenvalues of A, is as small as possible.

Instead of proving the convergence theorem (18.12), let us investigate some of its consequences. The first results follows directly from (18.12).

Suppose that A has only $k < n$ distinct eigenvalues, $\lambda_1, \ldots, \lambda_k$ then one can form the polynomial of degree k,

$$p(\lambda) = \prod_{j=1}^{k} \left(1 - \frac{\lambda}{\lambda_j}\right),$$

since $p(\lambda_j) = 0$ at all the eigenvalues λ_j the convergence result (18.12) tells us that the CG iteration terminates after k iterations.

This result is an extreme example of the clustering of eigenvalues of A—all the eigenvalues are clustered at the k distinct eigenvalues. In general one can show that the CG iteration can converge very fast if the eigenvalues are clustered around the smallest. More precisely, if the eigenvalues of A are given by $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$ then

$$\|\mathbf{x}^\star - \mathbf{x}_{k+1}\|_A^2 \leq \left(\frac{\lambda_{n-k} - \lambda_1}{\lambda_{n-k} + \lambda_1}\right)^2 \|\mathbf{x}^\star - \mathbf{x}_0\|_A^2 \,. \tag{18.13}$$

Suppose that $n-m$ eigenvalues are clustered around $\lambda_1 \approx 1$, such that $\lambda_{n-m} - \lambda_1 = \epsilon$, then (18.13) tells us that after m steps,

$$\|\mathbf{x}^\star - \mathbf{x}_{m+1}\|_A \approx \epsilon \, \|\mathbf{x}^\star - \mathbf{x}_0\|_A \,.$$

If ϵ is small, one can get a really good estimate after only m steps.

At the other extreme, if we know nothing about the clustering of the eigenvalues, except the 2-norm condition number, $\kappa = \lambda_{\max}/\lambda_{\min}$, then it can be shown that

$$\|\mathbf{x}^\star - \mathbf{x}_k\|_A \leq \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^k \|\mathbf{x}^\star - \mathbf{x}_0\|_A \,. \tag{18.14}$$

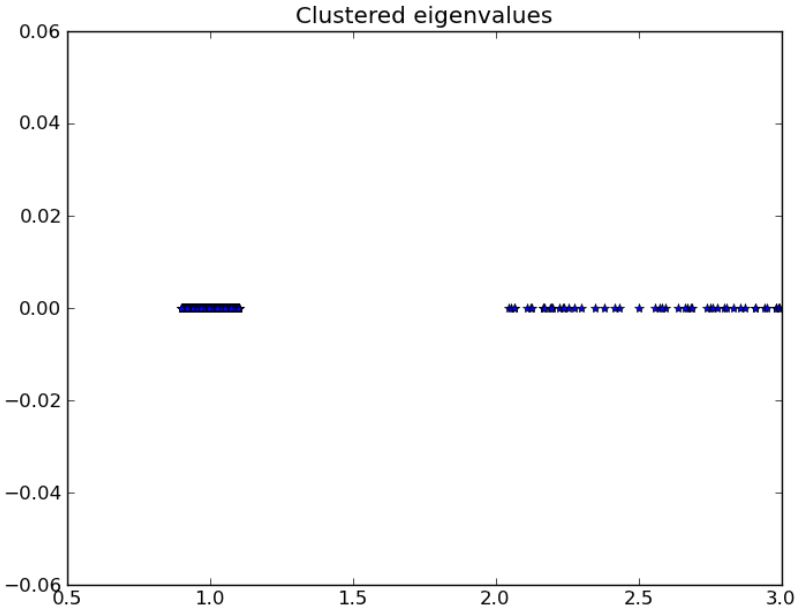


FIGURE 18.4.3. Clustered eigenvalues.

For large κ but not too large (we need $\kappa < n^2$), it follows that

$$\frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1} \approx 1 - \frac{2}{\sqrt{\kappa}}$$

so that any prescribed tolerance can be expected after $O(\sqrt{\kappa})$ steps. This might be a gross overestimate of the number of steps required.

EXAMPLE 82. The CG algorithm is applied to two different problems of size $n = 500$, illustrating the effect of the distribution of the eigenvalues on the convergence. In the first problem 50 eigenvalues are uniformly distributed (drawn randomly from a uniform distribution) between 2 and 3, with the rest uniformly distributed in the vicinity of 1, as illustrated in Figure 18.4.3. For the second problem all 500 eigenvalues are uniformly distributed over the interval $(0, 3)$, as illustrated in Figure 18.4.4. For this example the condition number is $\kappa = 400$.

The difference in convergence rates is shown in Figure 18.4.5. Clustered eigenvalues clearly have a huge effect on the convergence rate.

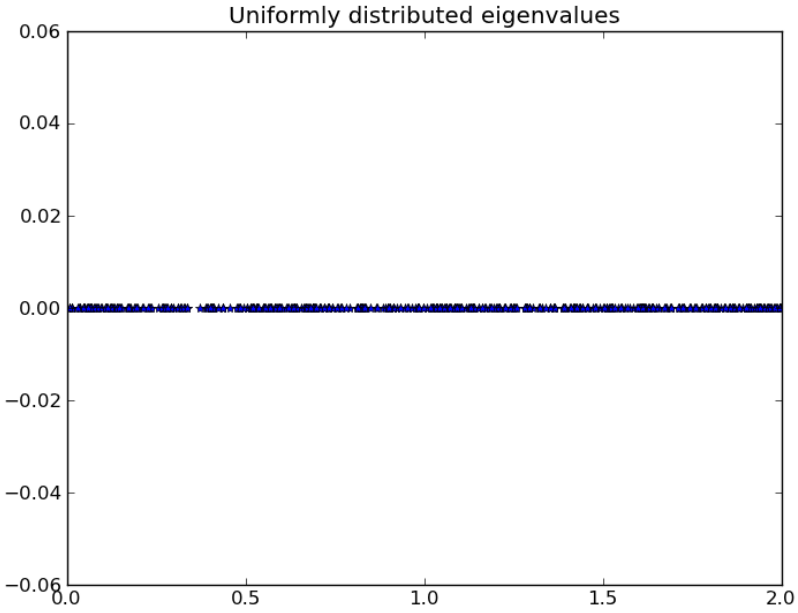


FIGURE 18.4.4. Uniformly distributed eigenvalues.

In practice a pre-conditioned conjugate gradient method is most often used. The basic idea is to multiply A and $\mathbf{b}$ with a matrix, say M, where the pre-conditioner M is chosen such that the eigenvalues of the product are more favorably clustered for faster convergence. The choice of M is as much an art as science and the best results are often obtained by constructing an M based on the specific application, see for example [**17**], [**14**].

18.4.2. A nonlinear conjugate gradient method. The conjugate gradient method as described above can be extended to nonlinear optimization problems in different ways. The Fletcher-Reeves method is a particularly simple extension. It amounts to linearizing the nonlinear problem, and then apply the standard conjugate gradient method. Two adjustments are required in order to solve the nonlinear optimization problem, repeated here for convenience,

$$\mathbf{x}^\star = \arg\min_{\mathbf{x}} \phi(\mathbf{x}). \tag{18.15}$$

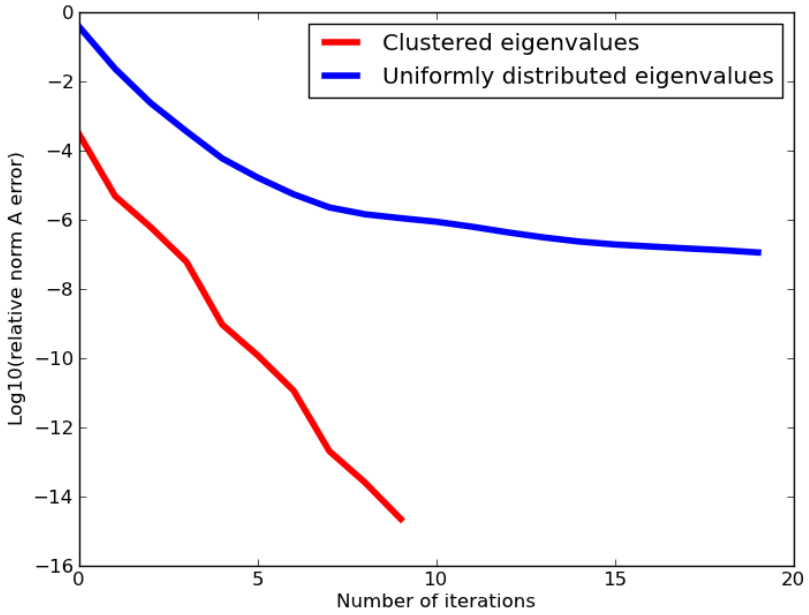


FIGURE 18.4.5. Convergence rates.

First, since ϕ is no longer a quadratic function, it is not possible to solve (18.5) analytically for the step length—an approximate minimum along the line in the direction of $\mathbf{p}_k$ is required. Secondly, the residual is replaced by the gradient $\nabla\phi_k := \nabla\phi(\mathbf{x}_k)$. With these two changes, the nonlinear conjugate gradient algorithm becomes:

Algorithm:

Given $\mathbf{x}_0 = \mathbf{0}$
Set $\mathbf{p}_0 = -\nabla\phi_0$, $k = 0$
while $\nabla\phi_k \neq \mathbf{0}$
    Calculate α_k by approximately
    minimizing $\phi(\mathbf{x}_k + \alpha_k\mathbf{p}_k)$
    $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k\mathbf{p}_k$
    $\beta_{k+1} = \frac{\nabla\phi_{k+1}^T\nabla\phi_{k+1}}{\nabla\phi_k^T\nabla\phi_k}$
    $\mathbf{p}_{k+1} = -\nabla\phi_{k+1} + \beta_{k+1}\mathbf{p}_k$
    $k = k + 1$

Since the line-search method used to estimate α_k may not be exact, there is a danger that the search direction $\mathbf{p}_{k+1}$ is not a descent direction. Taking the inner product of $\mathbf{p}_{k+1} = -\nabla\phi_{k+1} + \beta_{k+1}\mathbf{p}_k$ with $\nabla\phi_{k+1}$ we find that

$$\nabla\phi_{k+1}\mathbf{p}_{k+1} = -\nabla\phi_{k+1}^T\nabla\phi_{k+1} + \beta_{k+1}\nabla\phi_{k+1}^T\mathbf{p}_k. \tag{18.16}$$

If α_k is an exact minimizer of $\phi(\mathbf{x}_{k+1}) = \phi(\mathbf{x}_k + \alpha_k\mathbf{p}_k)$, i.e. if $\frac{d}{d\alpha_k}\phi(\mathbf{x}_{k+1}) = 0$, then $\nabla\phi_{k+1}^T\mathbf{p}_k = 0$, and $\nabla\phi_{k+1}\mathbf{p}_{k+1} < 0$, showing that $\mathbf{p}_{k+1}$ is indeed a descent direction. On the other hand if $\nabla\phi_{k+1}^T\mathbf{p}_k \neq 0$ then the second term on the right hand side of (18.16) may dominate, in which case $\mathbf{p}_{k+1}$ may not be a descent direction. It can be shown that $\mathbf{p}_{k+1}$ is a descent direction if the step length satisfies the strong Wolfe conditions,

$$\begin{aligned} \phi(\mathbf{x}_k + \alpha_k\mathbf{p}_k) &\leq \phi(\mathbf{x}_k) + c_1\alpha_k\nabla\phi_k^T\mathbf{p}_k, \\ \left|\nabla\phi^T(\mathbf{x}_k + \alpha_k\mathbf{p}_k)\mathbf{p}_k\right| &\leq -c_2\nabla\phi_k^T\mathbf{p}_k, \end{aligned}$$

where $0 < c_1 < c_2 < \frac{1}{2}$, and where the stronger condition on c_2 should be noted.

Recall that the desirable properties of the linear conjugate gradient depends on the fact that the first search direction is chosen as the steepest descent direction. For the nonlinear problem, the initial estimate is not necessarily inside the quadratic region in the neighborhood of $\mathbf{x}^\star$, but eventually it should enter this region. In order to regain the desirable convergence properties of the linear problem, it is advisable to restart the iteration with a steepest descent direction. Since one does not know when the quadratic region is entered the algorithm should incorporate regular restarts setting $\beta_{k+1} = 0$.

CHAPTER 19

GLOBAL OPTIMIZATION

19.1. Introduction.

The strategies discussed in the previous chapter for finding minima of an objective function $f(\mathbf{x})$ suffer from two serious problems. In the first place, we need to assume that the optimization surface $f(\mathbf{x})$is smooth, i.e. we need to calculate or approximate the gradient of f. There are many problems that do not allow us to do this. In the second place, the algorithm tends to get stuck in local minima. Let us address the first problem first by changing strategies.

Instead of finding a new estimate $\mathbf{x}_{k+1}$, given $\mathbf{x}_k$ by moving in a prescribed direction—no longer available since we no longer assume the necessary smoothness of f—let us select an $\mathbf{x}$ in the vicinity of $\mathbf{x}_k$ through a probabilistic process. This means that any $\mathbf{x}$ in a small neighborhood of $\mathbf{x}_k$ can potentially be selected. Once $\mathbf{x}$ is selected, calculate $f(\mathbf{x})$. There is no guarantee that it is a better estimate than $\mathbf{x}_k$, i.e. it is possible that $f(\mathbf{x}) \geq f(\mathbf{x}_k)$. If this happens, we discard it and select a new random candidate. If, on the other hand, $f(\mathbf{x}) \leq f(\mathbf{x}_k)$, we assign $\mathbf{x}_{k+1} = \mathbf{x}$. It should be clear that following this probabilistic procedure, we again generate a sequence $\mathbf{x}_0,\ \mathbf{x}_{1,} \ldots$ such that $f(\mathbf{x}_{k+1}) \leq f(\mathbf{x}_k),\ k = 0, 1, \ldots$.

Although we no longer require the gradient, it should be obvious that this procedure is also prone to get stuck in local minima as it moves down the slope of decreasing $f(\mathbf{x})$ with no hope of getting out once it reaches a local minimum, see Figure 19.1.1. It should also be clear that the local minimum reached by moving 'downhill' all the time, depends on the starting value—the first local minimum that is reached, is the one where it gets stuck.

If we are interested in the global minimum, we need to find a way to avoid getting stuck in local minima. This is the fundamental problem addressed in this chapter.

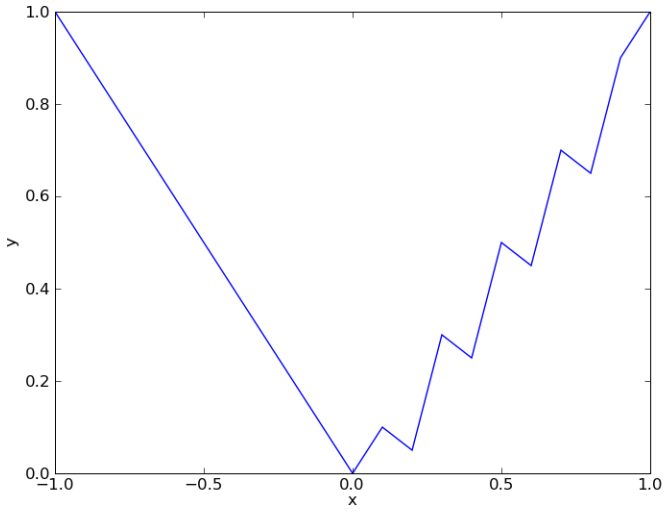


FIGURE 19.1.1. A function with many local minima.

One strategy for finding a global minimum readily presents itself. Since the minimum that gets selected depends on the starting value, one can choose N different, randomly selected, starting values. If one has reason to favor a specific region of the optimization space, it is always possible to reduce the number of required starting values by choosing from that particular region. It should not be hard to convince yourself that if $N \to \infty$, the global minima (no reason why there should be only one), will be found. Of course in practice it may simply not be possible to cover the solution space in any meaningful way, in which case there is a real danger of missing the global minima.

The two algorithms discussed in this chapter, Simulated Annealing (SA), and Genetic Algorithms (GA) provide different *strategies* to cover the solution space. However it is important to bear the following in mind:

(1) Using N different starting values together with say, a line search method, is often a viable approach, in particular if additional knowledge about the structure of the solution space is available. Once a starting value is given, line-search methods tend to be fast.

(2) Both SA and GA provide *strategies* for finding global optima. In practice neither guarantees that a global optimum will be found given finite computational resources.

(3) A consequence of the previous point is that in engineering one often has to be satisfied with a good optimum, not necessarily the global or best optimum. This is not necessarily disastrous. An interesting example is provided by evolutionary biology. The eye of vertebrates (including humans) and the octopus developed independently. From our point of view one can perhaps think, different starting values, or different choices during the execution of the algorithms. Evolution has a strong stochastic character! Although there are similarities between the structures of the eyes, two different solutions (local minima) were reached as illustrated in Figure 19.1.2. Two different, but functional designs were obtained. The main difference is that in vertebrates the nerve fibers lie in front of the retina whereas with the octopus the retina is in front of the nerve fibers. This means that in the case of vertebrates light needs to pass through the nerve fibers in order to reach the retina. It also causes a blind spot where the nerve fibers pass through the retina. Thus the octopus has perhaps the better design!

The point is that in engineering one can sometimes live with a good local optimum, often the best one can do anyway, given limited computational power.

It is interesting that the two algorithms discussed in this chapter are based on concepts from statistical physics and evolutionary biology, thus creating connections between three different disciplines.

19.2. Simulated Annealing

In order to reflect the statistical physics origins of Simulated Annealing (SA), as explained below, we now change notation. The objective function is now called the *energy* $E(\cdot)$ and we are interested in finding the *state* $\mathbf{x}^\star$ of lowest energy,

$$\mathbf{x}^\star = \arg\max_{\mathbf{x}} E(\mathbf{x}), \tag{19.1}$$

where the states can either be discrete, in which case we are dealing with a combinatorial optimization problem, or continuous. This is of course the same problem that

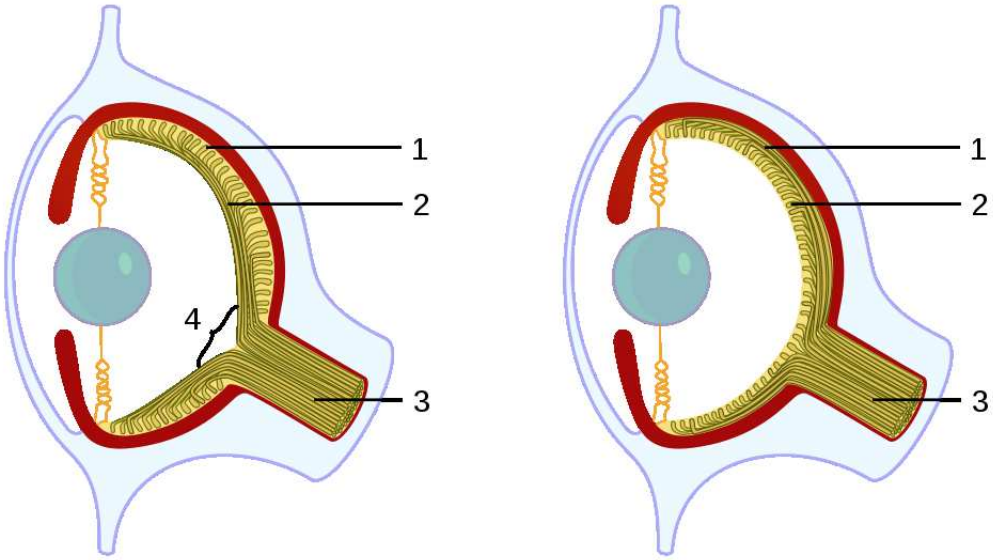


FIGURE 19.1.2. The eye of vertebrates (left) and the octopus (right): 1. Retina, 2. Nerve fibers, 3. Optical Nerve, 4. Blind spot, in vertebrates.

we solved in the previous chapter using line-search methods. Also note that we do not make any assumptions on the smoothness of $E(\mathbf{x})$.

Good general references include, **[18, 12]**

19.2.1. Basic algorithm. Starting with the probabilistic algorithm described in the Introduction (Section 19.1), we select a candidate $\mathbf{x}$ in the vicinity of the current estimate $\mathbf{x}_k$ according to some probability distribution centered at $\mathbf{x}_k$. If we proceed as described above, the iteration reaches a minimum, where it gets stuck. There are basically two ways out of a local minimum. The first is the one mentioned in Section 19.1, namely start from a different initial value. Alternatively, and this is the choice explored in this section, it is necessary to sometimes accept selections $\mathbf{x}$ that are worse than the current estimate in the sense that $E(\mathbf{x}) > E(\mathbf{x}_k)$. The only way out of the local minima shown in Figure 19.1.1, is to climb up and out of them. We first a general description of this can be achieved, filling in more details later.

(1) SA is an iterative procedure. From an initial value $\mathbf{x}_0$ SA generates a sequence of estimates $\mathbf{x}_k$ that hopefully converges to a global minimum.

(2) Finding a new estimate $\mathbf{x}_{k+1}$ given $\mathbf{x}_k$ involves a probabilistic process. This probabilistic process needs to specified by the user. There is no best way of generating new candidates that will work for all problems.

(3) Since the generation of a nearby $\mathbf{x}_{k+1}$, given $\mathbf{x}_k$, involves a probabilistic process, $\mathbf{x}_{k+1}$ is allowed to be 'worse' than $\mathbf{x}_k$ in the sense that the energy at the new state may be higher than the current state, i.e. it is possible that $E(\mathbf{x}_{k+1}) > E(\mathbf{x}_k)$. This is essentially the mechanism that allows one to escape from a local minimum.

(4) Once a candidate state $\mathbf{x}$ is selected by the probabilistic process, there is a choice whether to accept or reject it. For $\Delta E_k = E(\mathbf{x}_{k+1}) - E(\mathbf{x}_k)$ it is accepted with probability 1, if the energy is decreased, i.e. $\Delta E < 0$ (this means it is always accepted if $\Delta E < 0$). Otherwise it is accepted with probability given by the Metropolis criterion, $\exp\left(-\frac{\Delta E}{T}\right)$, where T is the so-called *temperature*, more about it later. The selected state $\mathbf{x}$ is therefore accepted with a probability given by

$$P(\mathbf{x}_{k+1} = \mathbf{x}|\Delta E_k, T) = \min\left[1,\ \exp\left(-\frac{\Delta E_k}{T}\right)\right]. \tag{19.2}$$

The larger the temperature T, the higher the probability that $\mathbf{x}$ will be accepted, regardless of whether it is an 'improvement' on $\mathbf{x}_k$. There is however, no compelling reason to use the Metropolis acceptance criterion. In order to escape local minima we only need $P(\mathbf{x}_{k+1} = \mathbf{x}|\Delta E_k, T) > 0$ if $\Delta E_k > 0$ and $T > 0$, and if the temperature T goes to zero, then $P(\mathbf{x}_{k+1} = \mathbf{x}|\Delta E_k, T)$ must tend to zero if $\Delta E_k > 0$, and to a positive value if $\Delta E_k < 0$.

(5) The temperature T is gradually decreased during the process, according to a 'cooling schedule' that needs to be specified by the user. Thus fewer candidate states that lead to an increase in the energy are accepted as the temperature cools down.

Before we provide further detail, let us outline a very basic SA algorithm. The basic SA algorithm given in Table 1 assumes that the energy $E(\mathbf{x})$, initial state estimate $\mathbf{x}_0$, a selection criterion, `select`$(\mathbf{x})$, and a cooling schedule, `temp`(t), are available. In addition it also needs a random number generator, `random`(), that returns a random

$\mathbf{x} \leftarrow \mathbf{x}_0$, $e \leftarrow E(\mathbf{x}_0)$	# Set initial state and energy
$k \leftarrow 0$	# Set iteration counter
while $k < kmax$	# While there is time left
$\mathbf{x}_{new} \leftarrow$ **select**$(\mathbf{x})$	# Find a new state in the vicinity
$e_{new} \leftarrow E(\mathbf{x}_{new})$	# Calculate its energy
if $P\left(\mathbf{x}_{new} \vert (e_{new} - e),\ \text{temp}(\frac{k}{kmax})\right) > \text{random}()$	# Should we accept?
$\mathbf{x} \leftarrow \mathbf{x}_{new}$, $e \leftarrow e_{new}$	# Yes, change to new state
$k \leftarrow k + 1$	# One more iteration
return $\mathbf{x}_{new}$, e_{new}	# Return the optimal state and its energy

TABLE 1. The SA Algorithm.

number between 0 and 1, drawn from a uniform distribution, as well as a stopping criterion. This algorithm terminates after $kmax$ iterations.

The performance of SA depends on the selections we make. As pointed out above, there is no single choice that works for all problems. In fact, a small modification to the algorithm described above, vastly increases its performancs. In order to understand the reasons for this, and to develop some intuition of how the different choices might affect the performance of the algorithm, we briefly recall its origins in statistical physics.

19.2.2. Origins in Statistical Physics. In condensed matter physics 'annealing' refers to the process where a solid is heated in a heat bath to the point where all the particles arrange themselves randomly in a liquid phase of the solid. If this is followed by a cooling, the particles arrange themselves in the low-energy ground-state of a lattice, provided the cooling takes place sufficiently slowly. This cooling down should be slow enough that at each temperature T the particles are allowed to reach thermal equilibrium, characterized by the fact that the probability of being in a state i with energy E_i is given by the Boltzmann distribution,

$$P(\text{state} = i) = \frac{1}{Z(T)} \exp\left(-\frac{E_i}{k_B T}\right), \tag{19.3}$$

where k_B is the Boltzmann constant and $Z(T)$ a normalization depending on the temperature T, also referred to as the *partition function*,

$$Z(T) = \sum_i \exp\left(-\frac{E_i}{k_B T}\right). \tag{19.4}$$

Note that we assume that the states take on discrete values, in this description we are discussing a combinatorial optimization problem.

A simulation of this annealing process was provided by Metropolis et al. They propose a Monte Carlo method very much along the lines of the SA algorithm described above. Given the current state i with energy E_i of the system characterized by the position of its particles, they propose a small randomly generated perturbation in the position of a randomly chosen particle, changing the system into state j with energy E_j. Assuming that the Boltzmann distribution is satisfied it follows that

$$\begin{aligned} \frac{P(\text{state } j)}{P(\text{state } i)} &= \frac{\exp\left(-E_j/k_B T\right)}{\exp\left(-E_i/k_B T\right)} \\ &= \exp\left(-(E_j - E_i)/k_B T\right) \\ &= \exp\left(-\Delta E/k_B T\right). \end{aligned}$$

Thus, if the energy decreases, $\Delta E < 0$, i.e. if the new state is in a more desirable lower energy state, then the new state is accepted and the process continued. If however, $\Delta E \geq 0$, then the new state is accepted with probability given by the Metropolis criterion (recall (19.2)),

$$\exp\left(-\frac{\Delta E}{k_B T}\right), \tag{19.5}$$

where-after the process is continued.

Note that if we follow this procedure we sample all possible configurations consistent with the Boltzmann distribution. This allows one to estimate averages by summing over the path we follow through all allowable states. In general it is not necessary to confine oneself to the Boltzmann distribution, any distribution will do, giving rise to the powerful Markov Chain Monte Carlo (MCMC) algorithm, widely used to estimate quantities such as expected values (integrals) or histograms, see for

example Bishop [**4**]. Note in particular that this procedure does not require knowledge about the partition function, a quantity that can be very difficult to estimate in practice since we need to average over all the possible configurations.

Some of the macroscopic quantities physicists are interested in include the *expected energy*

$$\langle E(T)\rangle = \sum_i E_i Q_i(T) \tag{19.6}$$

where $Q_i(T)$ is the probability of state i at temperature T, satisfying the Boltzmann distribution,

$$Q_i(T) = \frac{1}{Z(T)} \exp\left(-E_i/k_B T\right),$$

and the *entropy*,

$$S(T) = -\sum_i Q_i(T) \ln Q_i(T). \tag{19.7}$$

It now follows directly from the definitions and the expression for the Boltzmann distribution that

$$\frac{d}{dT}\langle E(T)\rangle = \frac{\sigma^2(T)}{k_B T^2},$$

and

$$\frac{d}{dT} S(T) = \frac{1}{k_B T}\frac{d}{dT}\langle E(T)\rangle,$$

where the variance $\sigma^2(T)$ in the energy is given by,

$$\begin{aligned}\sigma^2(T) &= \sum_i \left\langle \left(E(T) - \langle E(T)\rangle\right)^2 \right\rangle \\ &= \left\langle E^2(T)\right\rangle - \langle E(T)\rangle^2.\end{aligned}$$

These expressions describe the rate of change of the equilibrium quantities with temperature. Thus, if the system is allowed to reach equilibrium before the temperature is lowered, the average energy and entropy decrease during the annealing process.

If we now take the intuitive leap, and think of the objective function of the minimization problem as the energy of a system, and the independent variable as describing the states of a system, then the simulated annealing algorithm models

$\mathbf{x} \leftarrow \mathbf{x}_0$, $e \leftarrow E(\mathbf{x}_0)$	# Set initial state and energy
$k \leftarrow 0$	# Set iteration counter
$T \leftarrow T_{max}$	# Set initial temperature
while $T > T_{min}$	# While the temperature is high
while not in thermal equilibrium	# While not yet in thermal equilibrium
$\mathbf{x}_{new} \leftarrow$ **select**$(\mathbf{x})$	# Find a new state in the vicinity
$e_{new} \leftarrow E(\mathbf{x}_{new})$	# Calculate its energy
if $P\left(\mathbf{x}_{new}\right\|(e_{new} - e), T) >$ random()	# Should we accept?
$\mathbf{x} \leftarrow \mathbf{x}_{new}$, $e \leftarrow e_{new}$	# Yes, change to new state
$T \leftarrow \text{temp}(T)$	# Cool down the temperature
return $\mathbf{x}_{new}$, e_{new}	# Return the optimal state and its energy

TABLE 2. The modified SA Algorithm.

the physical annealing process described above. It also suggests that both the average energy (average value of the objective function), as well as the 'entropy' of the simulated annealing process should decrease as we lower the temperature, provided of course that the system is allowed to reach thermal equilibrium before the temperature is lowered.

This realization suggests a modification of the algorithm of Table 1, where no allowance is made for reaching thermal equilibrium before the temperature is decreased. This however, is easily done and the modified algorithm is shown in Table 2.

In practice the simplest way to approximate thermal equilibrium is to iterate a fixed number of times at each temperature.

19.2.3. Cooling schedule. Strictly speaking it is not necessary to wait for thermal equilibrium before the temperature is decreased. Provided the cooling is sufficiently slow, it can be shown that the original of Table 1 converges to a global optimum with probability 1. This cooling unfortunately has to be exceedingly slow, of the order, $T_k \propto \frac{1}{\log k}$, where T_k is the temperature at the $k-th$ iteration. This is not realizable in practice. It is therefore more common to achieve *approximate* equilibrium, and then use a cooling schedule of the form,

$$T_{k+1} = \alpha T_k, \quad k = 0, 1, \ldots \tag{19.8}$$

where α is a fixed parameter, normally chosen just smaller than 1. This still requires an initial temperature. While there are elaborate schemes to choose the initial value, let it suffice to note it should be high.

19.2.4. Example. A well-known example of combinatorial optimization is the Traveling Salesperson (TS) problem. In this problem the traveler has to visit N cities once and only once, and returning home at the end of the journey. We also assume that any city can be reached from all other cities. The problem is to calculate the optimal route between the cities, where the optimal route is the one with the shortest traveling distance.

For a problem with N cities, there are $N!$ possible paths. Unless N is small, say less than 20, it is just possible for a fast computer to investigate all the possibilities. There is no chance to solve the problem by brute force for $N > 40$ cities.

Let us solve this problem with SA for the twenty largest cities in the US. Table 3 shows the cities with their latitudes and longitudes.

In order to simplify the equations let us label the twenty cities as $c_i,\ i = 1, \ldots, 20$. The problem is to find an order in which the cities are visited so that the total distance traveled is as small as possible. Each ordering therefore presents a different state (there are 20! of them), and the objective function for a given an ordering, is the total distance implied by that ordering. In order to specify the objective function, the distance between two cities, given their latitudes and longitudes, needs to be calculated. Approximating the earth with a perfect sphere of radius $R = 6371km$, we need to find the great-circle distance D_{AB} between cities A and B with coordinates (ϕ_A, λ_A) and (ϕ_B, λ_B) respectively, on the surface of the sphere. A numerically stable formula is the haversine formula,

$$D_{AB} = 2R \arcsin\left(\sqrt{\sin^2\left(\frac{\Delta\phi}{2}\right) + \cos\phi_A \cos\phi_B \sin^2\left(\frac{\Delta\lambda}{2}\right)}\right),$$

where $\Delta\phi = \phi_B - \phi_A$ and $\Delta\lambda = \lambda_B - \lambda_A$. Here the latitudes and longitudes are measured in radians.

Making use of the code developed by Richard J. Wagner (wagnerr@umich.edu), the initial configuration is chosen as a random order, with New York as home city. A new configuration is generated from the present one by selecting two cities at

City	Latitude (ϕ)	Longitude (λ)
New York City	40.72	74.00
Los Angeles	34.05	118.25
Chicago	41.88	87.63
Houston	29.77	95.38
Phoenix	33.45	112.07
Philadelphia	39.95	75.17
San Antonio	29.53	98.47
Dallas	32.78	96.80
San Diego	32.78	117.15
San Jose	37.30	121.87
Detroit	42.33	83.05
San Francisco	37.78	122.42
Jacksonville	30.32	81.70
Indianapolis	39.78	86.15
Austin	30.27	97.77
Columbus	39.98	82.98
Fort Worth	32.75	97.33
Charlotte	35.23	80.85
Memphis	35.12	89.97
Baltimore	39.28	76.62

TABLE 3. Cities with latitudes and longitudes (degrees).

random and then swap their order. The acceptance of a new configuration happens according to the Metropolis criterion (19.2). Thermal equilibrium is approximated by iterating 500 times at each temperature. The temperature itself is exponentially lowered from $T_{max} = 10^7$to $T_{min} = 0.01$, according to (19.8) with $\alpha = 0.99$. The final order, with total distance 10935km, selects the route: New York City, Philadelphia, Baltimore, Charlotte, Jacksonville, Memphis, Dallas, Fort Worth, Houston, Austin, San Antonio, Phoenix, San Diego, Los Angeles, San Jose, San Francisco, Chicago, Indianapolis, Columbus, Detroit, as illustrated in Figure 19.2.1.

Information about the way SA progresses is provided in Figure 19.2.2. First note how the energy (total distance traveled) is decreased with decreasing temperature in Figure 19.2.2(a). Figure 19.2.2(b) shows that for high temperatures almost all the states are accepted, regardless whether they improve on the energy or not. As the

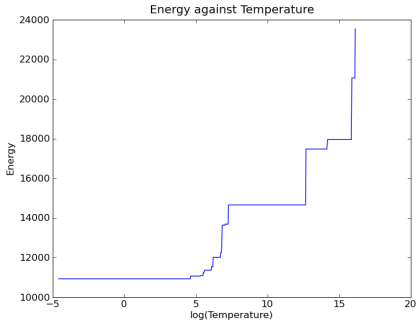
FIGURE 19.2.1. The optimal route visiting the twenty largest US cities, starting at New York city.

temperature cools down, fewer and fewer states are accepted, until eventually only those states that lowers the temperature are accepted. Figure 19.2.2(c) shows that for high temperatures about half of the states lead to an improvement in the energy (contrast this with the fact that almost all states are accepted at these temperatures). For cooler temperatures, very few states lead to an improvement—when the global minimum is reached, no selection of states improves the energy (but bear in mind that there can be multiple global minima).

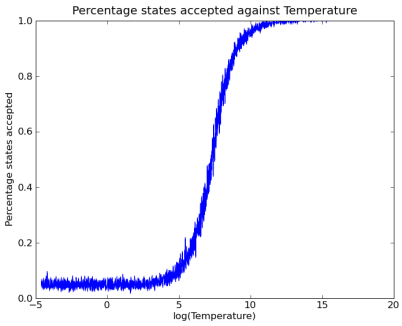
EXERCISE 83. Develop an SA program that solves the Sudoku problem.

19.3. Genetic Algorithms.

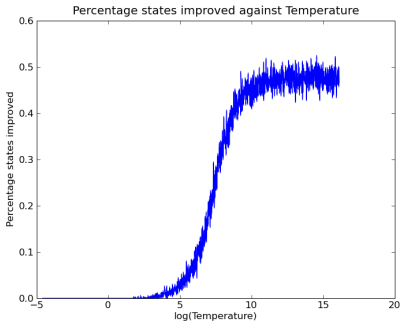
For Genetic Algorithms (GA) it is more natural to consider the maximization problem,



(a) Energy at each temperature.



(b) Fraction of states accepted at each temperature.



(c) Fraction of states leading to an improvement.

FIGURE 19.2.2. The traveling salesperson problem.

$$\mathbf{x}^\star := \arg\max_{\mathbf{x}} f(\mathbf{x}). \tag{19.1}$$

One can of course apply all the methods developed for the minimization problem to a maximization problem by replacing $f(x)$ with $-f(x)$. As in the case of simulated annealing, GAs also address the problem of finding *global* optima, without making any assumptions on the smoothness of the objective function. Unlike simulated annealing that originates from statistical physics, GAs draw their inspiration from evolutionary biology. Recalling Figure 19.1.2, it is appropriate the remind the reader that evolutionary biology does not always find the global optimum (unless the reader wants to argue that both solutions shown in Figure 19.1.2 are different *global* optima). Given finite resources there are certainly no guarantees that GAs will find

the global optima. Also, evolutionary biology work over evolutionary time scales. In our experience, GAs tend to be slow. They are however powerful and cannot be discarded. Among several good references, see for example **[8, 13]**.

Inspired by the mechanics of natural selection and natural genetics, GA's have a number of properties that distinguish it from the gradient-based methods of Chapter 18. These include

(1) Instead of working directly with the parameters, an encoding of the parameters is used. This can take many forms and need not even be numerical. We use a simple binary coding.
(2) Instead of starting with a single initial value, GA's start with a large population of initial values, often chosen at random.
(3) GA's only use information from the objective function, not its gradient.
(4) The transition from one generation to the next (one iteration) is determined by probabilistic, not deterministic rules.

The mechanics of a basic GA is quite simple, best illustrated by means of an example. Suppose we want to maximize

$$f(x) = x^2,\ x \in [0, 31],$$

i.e. we are looking for

$$x^\star = \arg\max_x f(x),\ x \in [0, 31].$$

Note that $f'(x^\star) \neq 0$ and gradient information is not particularly helpful in this case (it may help to determine the direction is which to move, but not how far). Let us systematically follow the basic steps of a GA in order to solve this problem.

Encoding:: GA's require the natural parameter (x in this case) to be coded as a finite string. Different encodings are possible. For the crossover process to be described below it is convenient to encode each member of the population with a string of the same length ℓ. In practice, it is common to use a binary encoding scheme, allowing a population consisting of at most 2^ℓ members. For our example the domain of the parameter is conveniently given as $[0, 31]$, and it is natural to encode it as a five-bit binary string. This means that

each member of the population is expressed as as a binary string of the form

$$a_5a_4a_3a_2a_1,$$

where each a_i can take on the values 0 or 1. If, in analogy to biological systems, we refer to each a_i as a single *gene*, then the full string represents a *chromosome*. It is important to note that one can assign a *fitness* value through the objective function. More specifically, by converting each string back to its decimal value

$$x = a_1 + a_2 2 + a_3 2^2 + a_4 2^3 + a_5 2^4$$

we can evaluate $f(x)$. Since we are looking to maximize $f(x)$, chromosomes with the highest values of $f(x)$ are the most desirable. Out of the 32 chromosomes allowed by our encoding, the GA finds the ones with the highest fitness.

Initial population:: As initial population we now choose n random samples from the 2^ℓ strings allowed by our encoding. Since we use a binary encoding, we simply flip an unbiased coin $n \times \ell$ times to find the n strings. If we we choose $n = 4$ for example, the initial population might look something like

$$\begin{matrix} 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \end{matrix}.$$

Of course if you again flip a coin 20 times, you should get different entries. It does not matter, the idea behind GA's is to identify the most desirable strings, somehow combine them to create even more desirable offspring. In practice n can be large, easily a number somewhere in the thousands. Instead of choosing the initial population completely randomly, it is of course possible to design schemes that will populate the most promising areas of parameter space.

Fitness:: We now determine the relative fitness of each string. For this each string is converted to its decimal equivalent, and its fitness is the value of the objective function. This is illustrated in Table 4.

No	String	x	Fitness $f(x)$	% of total
1	0 1 1 0 1	13	169	14.4
2	1 1 0 0 0	24	576	49.2
3	0 1 0 0 1	8	64	5.5
4	1 0 0 1 1	19	361	30.9
Total			1179	100
Average			293	
Max			576	

TABLE 4. Initial population and fitness values.

Reproduction:: The idea is to multiply the fittest strings and discard the unfit. One popular scheme of doing this is known in the GA literature as the roulette wheel reproduction. Imagine a roulette wheel with 100 equally spaced slots. We divide the 100 slots between the strings according to their fitness. For our example, we may award 14 slots to string 1, 49 slots to string 2, 6 slots to string 3, and 31 slots to string 4. We now spin the roulette wheel 4 times; each spin selects one of the current strings. Since string 2 has the most slots, it is to be expected that it will be selected more often than the rest. Indeed, if we do this in a real experiment, we might find the following selection after a total of 4 spins,

No.	Times selected
1	1
2	2
3	0
4	1

The new population therefore consists of

$$\begin{matrix} 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \end{matrix}.$$

Note that it is important to maintain genetic diversity. It is therefore necessary to guard against discarding too many seemingly unfit members, as this

may lead to premature convergence.
So far we have improved the quality of the population in terms of fitness by duplicating the fit and discard the unfit, but we have not introduced any new genetic material. This is achieved through a process known as *crossover*.

Crossover:: Crossover generates a new generation of strings. Having discarded (most of) the unfit members we marry two parent strings at random to produce two offspring. Note that this can already be done at the reproduction stage by simply spinning the roulette wheel twice to obtain the two parents. It is possible that the two parents are identical, in which case no new genetic material will be produced for these two individuals. This does not matter, since in general the parents will be different. At a specified crossover rate p_c the two parents will produce offspring through a crossover process. A typical value might be $p_c = 0.7$. This means that 30% of married parents produce no crossover, in which case the two offspring are identical to the parents. For strings of length ℓ, an integer number k is drawn from a uniform distribution between 1 and $\ell - 1$ (inclusive). This value specifies the crossover position. (There are $\ell - 1$ possible crossover positions for a string of length ℓ.) More specifically, say the two selected parent strings are $A_1 = \ 0 \ \ 1 \ \ 1 \ \ 0 \ \ 1 \ $, and $A_2 = \ 1 \ \ 1 \ \ 0 \ \ 0 \ \ 1 \ $, and that the crossover position turns out to be $k = 4$. Crossover means that the genes from positions $k+1$ to ℓ are swapped between the parents. Thus in our example with $k = 4$ we have

$$\begin{aligned} A_1 &= 0110 \ | \ 1 \\ A_2 &= 1100 \ | \ 0 \end{aligned}$$

where we have indicated the crossover position with a vertical line. After crossover, the children are

$$\begin{aligned} A_1' &= 0110 \ \ 0 \\ A_2' &= 1100 \ \ 1 \ . \end{aligned}$$

This random selection of parents (with replacement) and producing offspring through crossover, are repeated until we have reached the desired number of

String No.	Population after reproduction (Cross site shown)	Mate (randomly selected)	Cross site (randomly selected)	New population	x	$f(x)$
1	0 1 1 0 | 1	2	4	0 1 1 0 0	12	144
2	1 1 0 0 | 0	1	4	1 1 0 0 1	25	625
3	1 1 | 0 0 0	4	2	1 1 0 1 1	27	729
4	1 0 | 0 1 1	3	2	1 0 0 0 0	16	256
Sum						1754
Average						439
Max						729

TABLE 5. New population generated after crossover.

strings in our population. This process is summarized in Table 5. If we compare the fitness values of the new generation (Table 4) with that of the parents (Table 5), we see that both the average as well as the maximum fitness has improved.

There is just one more step before we are done with a single iteration (generation).

Mutation:: With GA's, as in nature it is important to maintain genetic diversity. This is exactly what mutations aim to achieve. After creation of the new population, each bit in the new population is inverted with a probability p_m. The main reason for this is that it creates the opportunity to explore parts of parameter space that might otherwise be unreachable. If we think of reproduction and crossover as a means to *exploit* past experience, then mutation allows us to *explore* beyond what is available to us through past observations of the fitness of our population. Exploitation and explorations are also persistent themes in Machine Learning, and it is important to strike a balance. A typical value for the mutation rate $p_m = 0.001$. From the latter one can deduce that mutations in GA's are rare, as in nature.

Termination:: We have now described the basic ingredients of a single iteration of a GA. This procedure is repeated until sufficiently high fitness levels are reached, or a maximum number of iterations (generations) reached. Note that one might end up with a whole population consisting of different, equally fit members. Thus it is possible that a number of local maxima

may be explored with a single GA. If the maximum number of iterations is reached the population may or may not contain sufficiently fit members.

It is not possible to explain the reason why GAs work, and we refer the interested reader to the literature, for example [**8, 13**].

CHAPTER 20

Quadrature

20.1. Introduction.

One of the basic tasks in numerical analysis is to compute integrals of the form

$$I = \int_a^b f(x)dx \tag{20.1}$$

where the limits a and b may be $-\infty$ or ∞ respectively, in which case it is assumed that the integral converges over the infinite interval. In this section we dicuss three approaches to the problem. Starting the with the simple trapeziodal rule, it used as a prototypical example of the polynomial-based approach where the idea is to derive rules that integrate polynomial up to certain order exactly on a uniform grid. The next idea is to relax the requirement of a uniform grid in which case it is possible to integrate higher order polynomials exactly, using the same number of grid points. Thus we arrive at the so-called Gaussian Quadrature (GQ) The final method is one of the forgotten gems of numercical analysis, namely Gregory's formula. For all these methods the integral (20.1) is replaced by the general quadrature formula,

$$I_N = \sum_{n=0}^{N} w_n f(x_n). \tag{20.2}$$

The different methods differ in their choice of the weights w_n and abscissae x_n.

20.2. Trapezoidal Rule.

Assuming a uniform grid with grid length

$$h = \frac{b-a}{N}$$

the (composite) trapezoidal rule is given by

$$I_N = \frac{1}{2}hf(a) + h\sum_{n=1}^{N-1} f(x_n) + \frac{1}{2}hf(b) \tag{20.1}$$

where $x_n = a + hn, \quad n = 0, \ldots, N$. This formula is easily derived by requiring that polynomials up to first degree (linear) are integrated exactly between x_n and $x_{n+1} = x_n + h$. More specifically, write the quadrature rule (20.2) as

$$\int_{x_n}^{x_{n+1}} f(x)dx \approx w_0 f(x_n) + w_1 f(x_{n+1})$$

where the weights are determined from choosing $f(x) = 1$,

$$h = \int_{x_n}^{x_n+h} 1dx = w_0 + w_1$$

and choosing $f(x) = x$,

$$\frac{1}{2}(x_n + h)^2 - \frac{1}{2}x_n^2 = \int_{x_n}^{x_n+h} xdx = w_0 x_n + w_1(x_n + h).$$

This system is easily solved to give the trapezoidal rule,

$$\int_{x_n}^{x_n+h} f(x)dx \approx \frac{1}{2}h\left(f(x_n) + f(x_{n+1})\right).$$

The composite trapezoidal rule (20.1) is then obtained by writing

$$\int_a^b f(x)dx = \sum_{n=0}^{N-1} \int_{x_n}^{x_{n+1}} f(x)dx \approx \sum_{n=0}^{N-1} \frac{1}{2}h\left(f(x_n) + f(x_{n+1})\right).$$

An alternative derivation is approximate $f(x)$ by a linear polynomial over $[x_n, x_{n+1}]$,

$$f(x) = -f(x_n)\frac{(x - x_{n+1})}{h} + f(x_{n+1})\frac{(x - x_n)}{h} + O(h^2), \tag{20.2}$$

and integrate the linear polynomial instead of $f(x)$.

20.2.1. Error analysis. From (20.2) follows that the error in the trapezoidal rule is given by

$$\begin{aligned}\int_{x_n}^{x_{n+1}} f(x)dx &= \frac{1}{2}h\left(f(x_n)+f(x_{n+1})\right)+\int_{x_n}^{x_{n+1}} O(h^2)dx \\ &= \frac{1}{2}h\left(f(x_n)+f(x_{n+1})\right)+O(h^3).\end{aligned}$$

Thus for the composite rule we commit an error of $O(h^3)$ over each of the $N=\frac{b-a}{h}$ intervals. The total error of the composite rule therefore becomes $NO(h^3)=O(h^2)$. Thus if we half h in the trapezoidal rule we expect the error to be reduced by a quarter. In general this is not sufficient and one would therefore consider higher order methods.

This is estimate is rather pessimistic however and in specific situations one can do a whole lot better. Let us assume for instance that the integrand is periodic, $f(x)=f(x+2\pi)$, and let us integrate over one period

$$\begin{aligned}I &= \int_0^{2\pi} f(x)dx \\ &\approx \frac{1}{2}f(0)+\sum_{n=1}^{N-1} f(x_n)+\frac{1}{2}f(2\pi) =: I_N,\end{aligned}$$

where $x_n=nh,\ \ n=0,\ldots,N$ with $h=2\pi/N$. Since $f(x)$ is periodic one can write it in terms of its Fourier series (see Section 7.2),

$$f(x)=\frac{1}{2\pi}\sum_{m=-\infty}^{\infty} a_n e^{imx},$$

where

$$a_m=\int_0^{2\pi} f(x)e^{-imx}dx.$$

Obviously $a_0=I$, the value of the integral. Note that at the end points, $x=0$ and $x=2\pi$ the Fourier series converges to $\frac{1}{2}\left(f(0)+f(2\pi)\right)$, in case of a discontinuity. But this is exactly the way that the trapezoidal rule approximates the boundary values. And this way of handling the end points has a remarkable effect on the accuracy of the trapezoidal rule—handling the end points correctly turns out to be the key. Let us compare the trapezoidal approximation I_N with the exact value.

It is easy to evaluate I_N, lumping together $\frac{1}{2}\left(f(0)+f(2\pi)\right)$ we find that

$$\begin{aligned} I_N &= h\sum_{n=0}^{N-1} f(x_n) \\ &= h\sum_{n=0}^{N-1}\frac{1}{2\pi}\sum_{m=-\infty}^{\infty} a_m e^{imx_n} \\ &= \sum_{m=-\infty}^{\infty} a_m \frac{1}{N}\sum_{n=0}^{N-1} e^{i2\pi mn/N}. \end{aligned}$$

This inner sum simplifies a lot since it is zero unless m is an integer multiple of N. Thus we get for integer s,

$$\begin{aligned} I_N &= a_0 + \sum_{s\neq 0} a_{sN}. \\ &= a_0 + \sum_{s=1}^{\infty}(a_{sN}+a_{-sN}). \end{aligned}$$

For periodic functions the error in the trapezoidal rule is therefore given by

$$\left| E_N := I - I_N = \sum_{s=1}^{\infty}(a_{sN}+a_{-sN}) \right|$$

and the question becomes, how quickly decay the Fourier expansion coefficients of the periodic function $f(x)$? It is not hard to derive the necessary estimates, integration by parts is all that is required, but the key observation is that it depends on the smoothness of $f(x)$. If for example $f(x)$ is ℓ times differentiable with $f^{(\ell)}(x)$ piecewise continuous with some jump discontinuities, then

$$a_{sN} = O\left(\frac{1}{(sN)^{\ell+1}}\right)$$

in which case the error becomes

$$E_N = O(h^{\ell+1}).$$

In the case that $f(x)$ is analytic in a region that includes the real axis, ℓ goes to infinity in which case the trapezoidal rule is of spectral accuracy, i.e. the Fourier

coefficients decay exponentially fast and the error becomes

$$E_N \propto e^{-\gamma/h},$$

where γ depends on the location of the nearest pole of $f(x)$ in the complex plane. This unbeatable by any 'higher order' scheme.

This is an example of a rather common theme, how the location of the singularities in the complex plane directly influences the behavior on the real axis.

We are so close that we might as well derive a similar result for integrals over the whole real line,

$$I = \int_{-\infty}^{\infty} f(x)dx,$$

where we again assume that the integrand decays sufficiently fast for the integral to converge. In this case the trapezoidal rule takes the form

$$I_N = h \sum_{n=-\infty}^{\infty} f(nh),$$

and we want to calculate the error

$$E_N = |I - I_N| .$$

This follows directly from the Poisson sum formula of Section 7.3, here written in the form,

$$h \sum_{n=-\infty}^{\infty} f(nh) = \sum_{k=-\infty}^{\infty} \widehat{f}\left(\frac{k}{h}\right),$$

where $\widehat{f}(\omega)$ is the Fourier transform of $f(x)$ defined by (see Section 7.3),

$$\widehat{f}(\omega) = \int_{-\infty}^{\infty} f(x)e^{-2\pi i\omega x}dx.$$

Thus

$$I := \int_{-\infty}^{\infty} f(x)dx = \widehat{f}(0),$$

so that the trapezoidal rule is related to the integral by

$$\begin{aligned} I_N &= h\sum_{n=-\infty}^{\infty} f(nh) \\ &= \sum_{k=-\infty}^{\infty} \widehat{f}\left(\frac{k}{h}\right) \\ &= \widehat{f}(0) + \sum_{k\neq 0} \widehat{f}\left(\frac{k}{h}\right) \\ &= I + \sum_{k\neq 0} \widehat{f}\left(\frac{k}{h}\right). \end{aligned}$$

The error in the trapezoidal rule is therefore again determined by the rate of decay of the Fourier terms,

$$E_N \leq \sum_{k\neq 0} \left|\widehat{f}\left(\frac{k}{h}\right)\right|.$$

If $f(x)$ is analytic then the Fourier terms decay exponentially fast, again leading to spectral accuracy in the trapezoidal rule.

There is one other interesting situation. If we are dealing with a *band limited* signal, i.e. a function $f(x)$ for which the Fourier transform vanishes outside a band limited by $|\omega| < T$, then the trapezoidal rule is *exact,* provided we choose $h < 1/T$. This is an important result for engineers to decide how densely they need to sample a signal in order to capture it perfectly. If they want to reproduce all frequencies up to T, they sample according to the Nyquist rate,

$$h < \frac{1}{T}. \tag{20.3}$$

EXAMPLE 84. Approximate $\int_{-1}^{1} e^x dx = e^1 - e^{-1}$ using the trapeziodal rule. The results are given in Table .

Note the typical quadratic convergence.

□

EXAMPLE 85. Approximate $\int_{-\infty}^{\infty} e^{-\frac{1}{2}x^2} dx = \sqrt{2\pi}$ using the trapezoidal rule. Inthis case we need to integrate far enough out so that the boundaries play no

Number of nodes	Error
8	1.2e-02
16	3.1e-03
32	7.7e-04
64	1.9-04

TABLE 1. Error in the trapezoidal approximation of $\int_{-1}^{1} e^x dx$.

Number of nodes	Error
8	5.6e-01
16	7.8e-04
32	3.6e-15

TABLE 2. Error in the trapezoidal approximation of $\int_{-\infty}^{\infty} e^{-\frac{1}{2}x^2} dx$.

role in the accuracy. We therefore apply the trapezoidal rule to $\int_{-12}^{12} e^{-\frac{1}{2}x^2} dx$. The results are summarized in Table 2.

The spectral convergence is particularly noticeable.

20.3. Gaussian Quadrature.

In this section we consider integrals of the form

$$I = \int_a^b w(x) f(x) dx, \tag{20.1}$$

where the limits a and b are allowed to be $-\infty$ or ∞ respectively. The weight function $w(x)$ is non-negative, $w(x) \geq 0$ and zero only at isolated points. In order to explain Gaussian Quadrature (GQ), we first need to say something about orthogonal polynomials, briefly encountered in Section 12.4 by way of the Chebyshev polynomials.

20.3.1. Orthogonal polynomials. Let us start with a specific example. The family of Legendre polynomials can be defined recursively,

$$(n+1)P_{n+1}(x) = (2n+1)xP_n(x) - nP_{n-1}(x), \tag{20.2}$$

with $P_0(x) = 1, \;\; P_1(x) = x$. It should be easy to see that each Legendre polynomial $P_n(x)$ is a polynomial of degree n. We list the first few:

$$\begin{aligned} P_0(x) &= 1 \\ P_1(x) &= x \\ P_2(x) &= \frac{1}{2}\left(3x^2 - 1\right) \\ P_3(x) &= \frac{1}{2}x\left(5x^2 - 3\right) \\ P_4(x) &= \frac{1}{8}\left(35x^4 - 30x^2 + 3\right) \\ &\vdots \quad . \end{aligned}$$

The property that is most important for our purposes is their orthogonality,

$$\int_{-1}^{1} P_m(x)P_n(x)dx = \begin{cases} 0 & \text{if } m \neq n \\ \frac{2}{2n+1} & \text{if } m = n \end{cases} . \tag{20.3}$$

Note that this means that $P_n(x)$ is orthogonal to *all* polynomials of degree less than n. The orthogonality ensures that $P_n(x)$ has n distinct real roots, all lying in the interval $(-1, 1)$.

In general a family of polynomials $P_n(x), \;\; n = 0, 1, \ldots$ of degree n, is orthogonal with respect to a weight function $w(x) \geq 0$ (zero only at isolated points), and an interval $[a, b]$ if

$$\int_a^b w(x)P_n(x)P_m(x)dx = 0, \;\; \text{if } m \neq n. \tag{20.4}$$

The limits a and b are allowed to be $-\infty$ or ∞ respectively. Also in the general case one can show that the polynomial $P_n(x)$ of degree n has exactly n isolated real roots, lying in the interval (a, b). In Table 3we list some of the common families, see **[2, 15]**, for example for more detail.

Family	(a, b)	$w(x)$	Recursion
Legendre	$(-1, 1)$	1	$(n+1)P_{n+1}(x) = (2n+1)xP_n(x) - nP_{n-1}(x)$ $P_0(x) = 1, \quad P_1(x) = x$
Chebyshev	$(-1, 1)$	$\frac{1}{\sqrt{1-x^2}}$	$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x)$ $T_0(x) = 1, \quad T_1(x) = x$
Hermite	$(-\infty, \infty)$	e^{-x^2}	$H_{n+1}(x) = 2xH_n(x) - 2nH_{n-1}(x)$ $H_0(x) = 1, \quad H_1(x) = 2x$
Laguerre	$(0, \infty)$	e^{-x}	$(n+1)\,L_{n+1}(x) = (2n+1-x)\,H_n(x) - nL_{n-1}(x)$ $L_0(x) = 1, \quad L_1(x) = -x+1$

TABLE 3. A few common orthogonal polynomial families.

20.3.2. Gaussian Quadrature. The idea is to approximate the integral (20.1) with a quadrature formula of the form,

$$\int_a^b w(x)f(x)dx \approx \sum_{n=0}^{N} w_n f(x_n). \tag{20.5}$$

The question that we need to answer is how to choose the weights w_n and nodes x_n in order to find the best approximation. Recall that in our discussion of the trapezoidal rule, we mentioned that a good criterion is to find the weights and nodes to exactly integrate polynomials up to a certain order. Also in that discussion we chose the nodes to be equidistant, a choice common to all Newton-Cotes formulas. Since we chose $N = 1$ we could do not better than integrating a linear polynomial exactly. For the more general case of (20.5) it is straightforward to derive a scheme that integrates polynomials up to degree N exactly. Given the nodes x_n, $n = 0, 1, \ldots, N$ (at this point any choice of node distribution is acceptable), write down the Lagrange interpolation polynomial (see Section 12.2) of degree N,

$$f(x) \approx L_N(x) = \sum_{n=0}^{N} \ell_n(x) f(x_n).$$

Integrating the Lagrange polynomial instead of $f(x)$ gives,

$$\int_a^b w(x)L_N(x)dx = \sum_{n=0}^{N} f(x_n) \int_a^b w(x)\ell_n(x)dx.$$

Comparing with the quadrature formula (20.5) gives us the weights,

$$w_n = \int_a^b w(x)\ell_n(x)dx, \tag{20.6}$$

that ensures that the quadrature formula is exact for all polynomials up to degree N, for a given node distribution x_n. However, the number of free parameters in the quadrature formula is $2(N+1)$, indicating the possibility that, with a good choice of all the parameters, polynomials up to degree $2N+1$ might be integrated exactly. We are happy with the choice (20.6) for the weights, the problem is to choose the nodes. The answer is both beautiful and simple. Recall that we mentioned that the $P_n(x)$ the member of degree n of the family of orthogonal polynomials with respect to the weight function $w(x)$ and interval $[a, b]$, has exactly n isolated real roots inside the interval (a, b). These roots are our choice of nodes x_n. Let us see how these choices of weights w_n and nodes x_n allow us to integrate polynomials up to degree $2N + 1$ exactly.

Let $Q_{2N+1}(x)$ be any polynomial of degree $2N+1$. Next we divide this polynomial by the orthogonal polynomial $P_{N+1}(x)$ of degree $N + 1$,

$$Q_{2N+1}(x) = Q_N(x)P_{N+1}(x) + R_N(x),$$

where $Q_N(x)$ is a polynomial of degree N, and $R_N(x)$ the remainder of degree less or equal to N. Let us do the calculation,

$$\begin{aligned} \int_a^b w(x)Q_{2N+1}(x)dx &= \int_a^b w(x)\left[Q_N(x)P_{N+1}(x) + R_N(x)\right]dx \\ &= \int_a^b w(x)R_N(x)dx, \end{aligned}$$

where the first integral on the right side vanishes because of the orthogonality of $P_{N+1}(x)$. But we choose the weights to integrate polynomials of degree N exactly,

$\pm x_k$	w_k
0.00000 00000	0.56888 88889
0.53846 93101	0.47862 86705
0.90617 98459	0.23692 68851

TABLE 4. The weights and nodes for the five point Gauss-Legendre rule.

consequently,

$$\int_a^b w(x)R_N(x)dx = \sum_{n=0}^{N} w_n R_N(x_n).$$

Since the nodes x_n are the zeros of $P_{N+1}(x)$, it follows that

$$\begin{aligned} Q_{2N+1}(x_n) &= Q_N(x_n)P_{N+1}(x_n) + R_N(x_n) \\ &= R_N(x_n). \end{aligned}$$

Putting all this together we find that

$$\int_a^b w(x)Q_{2N+1}(x)dx = \sum_{n=0}^{N} w_n Q_{2N+1}(x_n).$$

This is exact, no approximations involved.

In order to apply the GQ rule one has to find the weights and nodes for the appropriate orthogonal polynomial (note that a transformation of variables is often required in order to get the integral in one of the standard forms). Although this can be done using the formulas above, e.g. using Matlab's polynomial root-finding routine. This however quickly becomes unstable. A numerical stable procedure is obtained by relating the nodes to the eigenvalues of a tridiagonal matrix, which can be stably computed. Fortunately this can be done once and for all and the values tabulated for future use.

EXAMPLE 86. Find an approximation for $\int_{-1}^{1} e^x dx = e^1 - e^{-1}$, using the Gauss-Legendre rule. In order to use the Gauss-Legendre rule, the necessary weights and nodes are required. A good electronic source is http://dlmf.nist.gov/3.5#v. The data for the five point formula is given in Table 4.

The five point Gauss-Legendre approximation is therefore given by

$$\begin{aligned}\int_{-1}^{1} e^x dx &\approx 0.23692\,68851\left(e^{-0.90617\,98459}+e^{0.90617\,98459}\right)\\ &\quad +0.47862\,86705\left(e^{-0.53846\,93101}+e^{0.53846\,93101}\right)\\ &\quad +0.56888\,88889e^{0.0000000000}\\ &= 2.35040\,23866.\end{aligned}$$

Compare this with the analytical value, $e^1 - e^{-1} = 2.35040\,23873$.

□

EXAMPLE 87. Approximate $\int_0^\infty e^{-x}dx = 1$ using the five point Gauss-Laguerre rule. Since the weight function is $w(x) = e^{-x}$ the function $f(x) = 1$, and in this case the formula becomes particularly simple. The necessary values are given in Table 5. The five point Gauss-Laguerre approximation is therefore given by

$$\begin{aligned}GL_5 &= w_1 + w_2 + w_3 + w_4 + w_5\\ &= 0.10000\,00000 \times 10^1\end{aligned}$$

which is exact to the number of decimal places used. This is to be expected, all polynomials up to degree $N = 9$, should be integrated exactly.

□

EXAMPLE 88. Approximate $\int_0^\infty e^{-\frac{1}{2}x^2}dx = \sqrt{\pi/2}$ using the five point Gauss-Laguerre rule. In this case the weight function is $w(x) = e^{-x}$, we therefore integrate $\int_0^\infty e^{-x}e^{x-\frac{1}{2}x^2}dx$, with the result that $f(x) = e^{x-\frac{1}{2}x^2}$. The necessary values are given in Table 5.

x_k	w_k
0.26356 03197	0.52175 56106
$0.14134\ 03059\times10^{1}$	0.39866 68111
$0.35964\ 25771\times10^{1}$	$0.75942\ 44968\times10^{-1}$
$0.70858\ 10006\times10^{1}$	$0.36117\ 58680\times10^{-2}$
$0.12640\ 80084\times10^{2}$	$0.23369\ 97239\times10^{-4}$

TABLE 5. The weights and nodes for the five point Gauss-Laguerre rule.

The five point Gauss-Laguerre approximations is therefore given by

$$\begin{aligned} GL_5 &= 0.5217556106 \times f(0.2635603197) \\ &\quad +0.3986668111 \times f(0.1413403059 \times 10^1) \\ &\quad +0.7594244968 \times 10^{-1} \times f(0.3596425771 \times 10^1) \\ &\quad +0.3611758680 \times 10^{-2} \times f(0.7085810006 \times 10^1) \\ &\quad +0.2336997239 \times 10^{-4} \times f(0.1264080084 \times 10^2) \\ &= 1.2636725238561697, \end{aligned}$$

which should be compared with the analytical value, $\sqrt{\pi/2} = 1.2533141373155001$. One should not expect too much using only five function evaluations. For the ten point scheme the approximation improves to $GL_{10} = 1.2532905795030598$, still requiring only ten function evaluations.

□

20.4. Gregory's Method.

Gregory's method is among the very first quadrature formulas ever described in the literature, dating back to James Gregory (1638–1675). It seems to have been highly regarded for centuries, but is less often seen these days. Fröberg [**7**] offers an operator algebra type derivation but also made the puzzling comment, "Gregory's formula, as a rule, is not used for numerical quadrature but sometimes for summation". Later authors usually ignore Gregory's method. Nevertheless we believe it is one of the real gems from long ago that should find its way back into textbooks.

For simplicity we start by considering the approximations of a convergent integral,

$$I = \int_0^\infty f(x)dx, \tag{20.1}$$

based on the function values only at the integer locations, $x_n = n, \quad n = 0, 1, \ldots$. A crude approximations is

$$I \approx \sum_{n=0}^{\infty} f(n) = f(0) + f(1) + \cdots . \tag{20.2}$$

In the next four subsections, we will

(1) Develop a sequence of 'end conditions' at $x = 0$ for increasing the accuracy of (20.2),
(2) Generalize the node locations from $x_n = n$ to $x_n = nh$, where h denotes an arbitrary node spacing.
(3) Change the integration interval from $[0, \infty]$ to an arbitrary finite interval $[a, b]$. In that step the assumption of the integral converging over the infinite interval can be dropped.
(4) Discuss Gregory's method and, in particular, compare it to Simpson's method.

20.4.1. End corrections. We know from the trapezoidal rule discussed in Section 19.2 that the coefficient of $f(0)$ should be $\frac{1}{2}$ instead of 1. If we use a node spacing of h instead of $h = 1$ as we do above, then this simple change in the boundary coefficient reduces the error from $O(h)$ to $O(h^2)$. It is then natural to ask whether additional end corrections can reduce the error, in some systematic way, to any desired order. This is not at all implausible, given that the trapezoidal rule is spectrally accurate over $(-\infty, \infty)$ for analytic functions that decay sufficiently fast in both directions. The situation is similar to the one discussed in Section 19.2 where spectral accuracy is achieved for periodic analytic functions. All algebraic error terms must therefore come from what happens at the $x = 0$ boundary. These error terms ought to be possible to identify and removed. Euler-Maclaurin, discussed in Section 10.5 but requires us to know high derivatives at the end point. Gregory's method is conceptually similar but does not require any such extra information.

Using the notation

$$\Delta f(x_n) = f(x_{n+1}) - f(x_n)$$

we obtain

$$\begin{aligned}
\Delta^0 f(0) &= f(0) \\
\Delta^1 f(0) &= f(1) - f(0) \\
\Delta^2 f(0) &= f(2) - 2f(1) + f(0) \\
\Delta^3 f(0) &= f(3) - 3f(2) + 3f(1) - f(0) \\
&\vdots
\end{aligned}$$

where we recognize the coefficients as coming from the successive lines of Pascal's triangle. We next try to 'correct' (20.2) by adding yet unknown amounts of the terms above,

$$I \approx \sum_{n=0}^{\infty} f(n) + \left(b_0\Delta^0 + b_1\Delta^1 + b_2\Delta^2 + \cdots\right) f(0). \tag{20.3}$$

At this point we recall that an arbitrary periodic function over $[0, 2\pi]$ can be written as a combination of e^{ikx}, $k \in \mathbb{Z}$ (Fourier series), and that functions that decay fast enough over $(-\infty, \infty)$ similarly can be written as a combinations of $e^{i\omega x}$, $\omega \in \mathbb{R}$ (Fourier transform). One might therefore guess that functions $f(x)$ that decay for increasing values x similarly can be written as combinations of e^{-zx} where $\mathrm{Re} z > 0$. Substituting $f(x) = e^{-zx}$ into (20.3) gives, after a few straightforward simplifications,

$$\frac{1}{z} = \frac{1}{1-z} + \left(b_0 - b_1(1-e^{-z}) + b_2(1-e^{-z})^2 - b_3(1-e^{-z})^3 + \cdots\right).$$

After changing variables $w = 1 - e^{-z}$, i.e. $z = -\log(1-w)$, we get

$$\frac{1}{\log(1-w)} + \frac{1}{w} = -b_0 + b_1 w - b_2 w^2 + b_3 w^3 - \cdots .$$

Since the right hand side is of the form of the Taylor series expansion of the left hand side, we can read off the coefficients from the Taylor expansion of $\frac{1}{\log(1-w)} + \frac{1}{w}$ around $w = 0$. Multiplying by the denominators, utilizing the well-known Taylor expansion of $\log(1-w)$, and equating coefficients lead to a simple recursion relation for the coefficients b_n. Expressed in matrix form, the coefficients are obtained by solving a

triangular Toeplitz system

$$\begin{bmatrix} 1 & & & & \\ -\frac{1}{2} & 1 & & & \\ \frac{1}{3} & -\frac{1}{2} & 1 & & \\ -\frac{1}{4} & \frac{1}{3} & -\frac{1}{2} & 1 & \\ \vdots & \vdots & & \ddots & \ddots \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ \vdots \end{bmatrix} = \begin{bmatrix} -\frac{1}{2} \\ \frac{1}{3} \\ -\frac{1}{4} \\ \frac{1}{5} \\ \vdots \end{bmatrix}$$

from which follows,

$$b_0 = -\frac{1}{2},\ \ b_1 = \frac{1}{12},\ \ b_2 = -\frac{1}{24},\ \ b_3 = \frac{19}{720},\ \ b_4 = -\frac{3}{160},\ \ b_5 = \frac{863}{60480},\ \ b_6 = -\frac{2751}{24192}, \cdots .$$

Analysis, which we do not present here, shows that, for each extra coefficient that is included, the overall accuracy increases with one power of h (in the case of using a spacing h rather than just $h = 1$).

20.4.2. Change of grid spacing. Changing the grid spacing from $h = 1$ to arbitrary h, we first rewrite Gregory's formula (20.3) as

$$\int_0^\infty f(x)dx \approx \sum_{n=0}^{\infty} c_n f(n)$$

where the 'boundary' c_n's are no longer equal to one. A change of variables $n \to nh =: x_n$ inserts an h in front of the sum,

$$\int_0^\infty f(x)dx \approx h \sum_{n=0}^{\infty} c_n f(x_n). \tag{20.4}$$

If we include k corrective boundary terms, the coefficients $c_n,\ \ n = 0, \ldots, k-1$ are modified while the rest remain ones, i.e. $c_n = 1,\ \ n = k, k+1, \ldots$.

20.4.3. Using a finite interval. This change is also trivial. Instead of having one boundary on the left, we have another one on the right. All that is required is to apply the same corrections to the right hand boundary as what we have just arrived for the left hand boundary, except from symmetry, the order is reversed at the right hand boundary. When all is said and done, the first and last $c_n$coefficients are the same, the second and next-to-last are the same, etc.

20.4.4. Brief discussion. Everyone knows that the trapezoidal rule is usually pretty bad with an error of $O(h^2)$. However, important exceptions arise if the integrand is analytic, either or the infinite interval, or periodic, in which case the accuracy becomes spectral. In the general case for fixed, bounded intervals, modern text book typically advise the use of

(1) Newton-Cotes formulas (including Simpson's rule), and
(2) Gaussian Quadrature (GQ) formulas, discussed in Section 20.3.

If GQ formulas can be used, it is often the best option. Assuming however, as we do here, that the function values are only available at equidistant node locations, GQ is no longer an option.

Newton-Cotes formula, including Simpson's rule have several shortcomings:

(1) The accuracy of Simpson's rule is $O(h^4)$. Higher orders are sufficiently awkward to reach that they are seldom used.
(2) Recalling (the composite) Simpson's rule,

$$\int_a^b f(x)dx \approx \frac{h}{3}\left[f(x_0) + 4f(x_1) + 2f(x_2) + 4f(x_3) + \cdots + 4f(x_{N-1}) + f(x_N)\right]$$

where $x_n = a + hn, \quad n = 0, 1, \ldots, N$, with $h = \frac{b-a}{N}$, it is necessary to have an odd number of nodes. Higher order versions have even more awkward node number restrictions—it is not trivial to just jack up the order step-by-step.
(3) The oscillating Simpson/Newton-Cotes weights across the whole interval is a very clumsy way of handling errors that have their cause in what happens at the boundaries. Note for instance (explored further in the Exercises), that Simpson's rule can be obtained on a grid with spacing h, by using Richardson extrapolation on two trapezoidal approximations based on grids $2h$ and h. In the important cases where the trapezoidal rule is spectrally accurate, contaminating the $2h$ approximation with an h approximations, severely damages the accuracy. The Gregory approach involves no such accuracy destruction for smooth functions.

EXAMPLE 89. Experiment approximating $\int_{-1}^{1} e^x dx = e^1 - e^{-1}$, using Gregory's method for different number nodes and different orders of accuracy by including more boundary correction terms. The results are summarized in Table 6

order	number of nodes=11	21	31
2	7.8e-03	2.0e-3	8.7e-04
3	9.7e-04	1.3e-04	3.7e-05
4	8.0e-05	5.6e-06	1.1e-06
5	1.6e-05	5.3e-07	7.2e-08
6	1.3e-06	2.7e-08	2.5e-09
7	3.4e-07	3.0e-09	1.8e-10
8	2.4e-08	1.5e-10	6.7e-12

TABLE 6. Error in Gregory's method applied to $\int_{-1}^{1} e^x dx$.

EXAMPLE 90. $\int_0^\infty e^{-\frac{1}{2}x^2} dx = \sqrt{\pi/2}$

Part 4

PROBABILISTIC MODELING

Uncertainties are part of life. Observations can never be perfect, and the processes we need to investigate are often so complex that it is impossible to model it with any accuracy. In this part we therefore change emphasis. Instead of modeling physical processes using the fundamental laws of physics, we now build probabilistic models directly from data obtained from observations or measurements. Fundamentally we assume that the data is generated by a specific process, reflected by patterns and regularities that we are able to learn from the data. In order to illustrate this point, let me share with you my personal discomfort in searching for patterns in stock market trends (bearing in mind that many, if not most researchers do not share this discomfort with me). The process generating the stock market data is truely complex and there is no hope to do any detailed modeling, therefore a prime candidate for the type of modeling we'll be studying. So what is the problem? Let us suppose that we have been able to extract the patterns of market movements from the data, even if there is still significant uncertainty, and start to exploit this knowledge. It is inevitable that our exploitation has an effect on the mechanism that will eventually break the process, in which case it is necessary to start learning the patterns all over again. If we don't break the system, there is the potential of gaining umlimited wealth, and that does not make sense to me. Said in a different way, what we really need to do is to be able to predict the break-downs of the mechanism, i.e. when the patterns we have learned are no longer of any use to us. In which case the same objection applies.

Be it as it may, it is important that the data we study is generated by some orderly process, producing patterns and regularities that we can identify.

In this manuscript the emphasis is on parametric modeling, where the data is used to estimate the model parameters. The model itself depends on the questions one wants to ask, i.e. on the problem one has to solve. One may, for example, be interested in knowing whether a given handwritten signature is a forgery or not, or one may want to identify the person speaking on the other side of a telephone. In yet another application you may want to identify the corresponding points in a stereo vision pair. It is also possible to repair badly damaged images by estimating the most likely pixel values, based on the information conveyed by the undamaged part of the image.

In all examples like these we proceed in the same basic way. First the probabilistic model is constructed based on all available data. This is the learning phase, and it often boils down to estimating the parameters of a model using something called maximum likelihood. Note that finding the parameters is probably the easy part. The actual modeling process is much harder, and determines the efficacy of the system. In this part we'll introduce powerful techniques, as well as a number of real-life examples, always with the understanding that proficiency in the modeling process only comes with experience.

The second part of the process is to address queries to the model. This is where we actually find the answers to the questions we would like ask, such as those mentioned above. If the modeling is done carefully, the main issue that remains is probably computational speed. If it is our task to estimate the position and velocity of an aircraft during take-off and landing as part of an automated control system, we had better get the results in real time.

As it stands the reader will not fail to recognize our debt to the book by Chris Bishop [**4**], we have freely borrowed from his images that he has generously made available on-line. Other useful references include [**1, 3, 10, 11**]

CHAPTER 21

BASIC PROBABILITY

21.1. Introduction.

One needs surprisingly little background in probability in order to follow the discussion in this part of the manuscript. Everything we do is based on just two basic rules: the sum rule, and the product rule. In this chapter we develop the essential ideas.

21.2. Discrete Probability.

Everything we do derives from a few basic rules of probability. Assuming that you are at least familiar with the basic ideas we go straight to a more general discussion.

We need to specify a a triple $(X, \mathcal{A}_X, \mathcal{P}_X)$, the discrete random variable, X, the set of possible values, or outcomes, or realizations of X, $\mathcal{A}_X = \{a_1, \ldots, a_I\}$, and the relative frequencies with which we observe the different values of X, $\mathcal{P}_X = \{p_1, \ldots, p_I\}$, where $P(X = a_i) = p_i,\ p_i \geq 0$ and $\sum_{a_i \in \mathcal{A}_\mathcal{X}} P(X = a_i) = 1$. The two conditions, $p_i \geq 0$, and $\sum_{i=1}^{I} p_i = 1$, ensure that that the p_i are proper probabilities.

Instead of $P(X = a_i)$ the shorthand $P(a_i)$ or $P(X)$ will often be used but make sure you understand the distinction between a random variable X, written in upper case, and its value or realization, x or a_i, written in lower case. (Often we don't make the distinction, hoping it will be clear from the context.)

If T is a subset of $\mathcal{A}_X$ then, noting that the values are mutually exclusive,

$$P(T) = p(X \in T) = \sum_{a_i \in T} P(a_i).$$

A *joint ensemble* XY is an ensemble in which each outcome is an ordered pair, $(x,\ y)$ with $x \in \mathcal{A}_X$ and $y \in \mathcal{A}_Y$. We call $P(X, Y)$ the joint probability of X and Y. More intuitively, $P(X = x, Y = y)$ is the probability of observing the values x and y simultaneously.

The *conditional* probability is given by

$$P(X = a_i|Y = b_j) = \frac{P(X = a_i, Y = b_j)}{P(Y = b_j)}.$$

Note that the conditional probability is not defined if $P(Y = b_j) = 0$. The conditional probability $P(X|Y)$ is read as 'the probability of X, given Y'. It will be necessary to discuss this in more depth in order to develop an intuition. Let it suffice for now to point out that the conditional probability tells us what we can learn about the probability of X, given information about Y. Also note that $P(X|Y)$ is a probability distribution over X, and is therefore properly normalized, i.e.

$$\sum_{X \in \mathcal{A}_X} P(X|Y) = 1,$$

for *any* value of $Y = y$. This leads us to the product rule

$$P(X, Y) = P(X|Y)P(Y) = P(Y|X)P(X), \tag{21.1}$$

where the second equality derives from the fact that the joint probability is symmetric (we make no distinction in the order when we observe X and Y simultaneously). Together the two factorizations lead to the important Bayes' theorem

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}. \tag{21.2}$$

Before we proceed with a detailed discussion of Bayes' theorem, we derive another useful result, the *marginal* probability, or the *sum* rule. Since the conditional probability is normalized,

$$1 = \sum_{Y \in \mathcal{A}_Y} P(Y|X) = \frac{1}{P(X)} \sum_{Y \in \mathcal{A}_Y} P(X, Y)$$

it follows that the denominator $P(X)$ is given by the marginal probability,

$$P(X) = \sum_{Y \in \mathcal{A}_Y} P(X, Y), \tag{21.3}$$

and,

$$P(Y) = \sum_{X \in \mathcal{A}_X} P(X, Y).$$

It might be useful to note that the joint probability $P(X, Y)$ is the most fundamental of the quantities mentioned above. Knowing the joint probability one derives the marginals as well as the conditional probabilities.

We alluded to the denominator as a normalization constant (as far as Y is concerned), ensuring that the posterior probability is properly normalized. Using the sum and product rules one can also write the denominator as

$$P(X) = \sum_{Y \in \mathcal{A}_Y} P(X|Y)P(Y). \tag{21.4}$$

For reasons that will become clear later, it is also called the *evidence*.

Note: If $P(X|Y) = P(X)$, i.e. if the conditional probability does not depend on Y, the two variables are *statistically independent*. In particular this implies that for statistically independent variables

$$P(X, Y) = P(X)P(Y).$$

In practice this means that any knowledge about Y has no effect on what we know about X.

It is important to note that dependencies do not imply any causal relationship between the variables. Statistical dependence means that new information about one variable changes the state of our knowledge of the other. For example, if we hear a cock crow, there is a distinct probability that the sun is about to rise. It does not however, imply that the rising of the sun is *caused* by the cock crowing[1]. Although we do not study causal relationships, it is of considerable interest. It is for example, of extreme importance to discover if a desired effect is actually caused by the medicine administered to the patient, or due to some other serendipitous effect. Establishing causal effects is beyond the scope of this work, see [**16**].

In the next example we use Bayes' theorem to investigate the practical efficacy of a signature verification system.

EXAMPLE 91. Suppose you have developed a signature verification system with error curves shown in Figure 21.2.1. For further background you may want to read Chapter .

[1]Let us not blame the cock; politicians also believe they make the world go round.

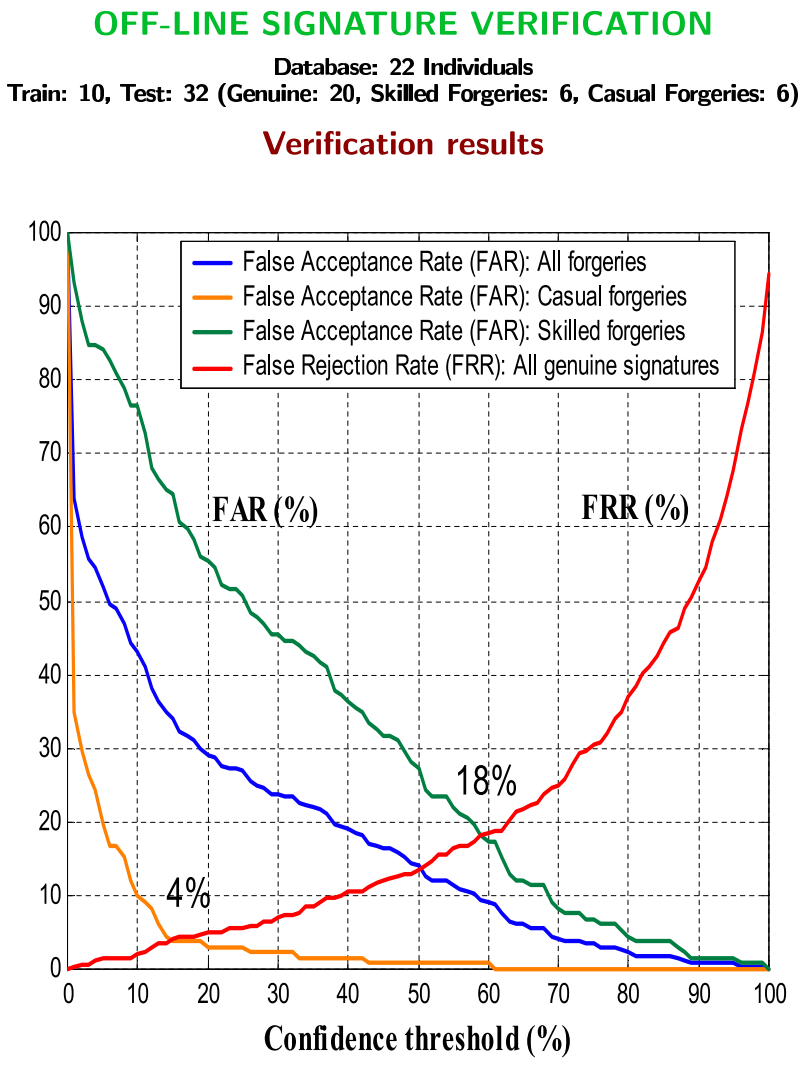


FIGURE 21.2.1. Error rates of a static signature verification system.

It is clear from the error curves that your system is not perfect—there does not exist a perfect system. It will on occasion accepts a forgery, or reject a genuine signature. Suppose you have installed your system in a bank and your system rejects a signature. What is the probability that it is actually a forgery? This is an important question. If your system rejects a large percentage of genuine signatures, it is basically useless. From the Figure is is clear that you cannot expect any better than a 4% equal error rate (when the false acceptance rate equals the false rejection rate). Crime statistics are normally given as a number per one hundred thousand of the population. We don't have exact statistics of the number of forgers in a society but in countries with high crime rates, a number of forty per hundred thousand is typical. Use these numbers to calculate the probability that a signature is a forgery if rejected by the system. What should the error rate be before your system becomes useful, if by 'useful' you mean that every second signature that is rejected is actually a forgery? It is worth spending some time on this example.

First of all, let us introduce our random variables. Let S denote the signatures, i.e. S can take on two values, $S = t$ indicating a genuine signature, and $S = f$ a forgery. Similarly, let M indicates the measurement (the output of the system) so that $M = t$ and $M = f$ indicate a genuine signature and a forgery respectively, according to the system.

Let us start by considering what we can say about a signature without the benefit of our system, i.e. if we are presented with a signature how much confidence do we have that it is genuine? The important thing to realize is that we actually have some faith that it is genuine based on prior knowledge about the behavior of people in our community. Using the crime statistics quoted above, let us suppose that the probability that it is a forgery is $\frac{40}{100\,000}$, i.e. $P(S = f) = \frac{40}{100\,000} = 0.0004$. Without the benefit of the system we conclude that it is quite unlikely to encounter a forgery. Let us now consider how an actual measurement of the system affects this prior belief. Suppose the system indicates a forgery, we are interested in the posterior $P(S = f|M = f)$. Using Bayes' rule we write

$$P(S = f|M = f) = \frac{P(M = f|S = f)P(S = f)}{P(M = f)}.$$

The likelihood, $P(M = f|S = f)$ is an indication of how well the system describes the signatures, and is used as a correction of our prior assumption. According to the error curves we have $P(M = f|S = f) = 0.96$, and the normalization constant $P(M = f) = P(M = f|S = f)P(S = f) + P(M = f|S = t)P(S = t) = 0.96 \times 0.0004 + 0.04 \times 0.9996 = 0.0404$. The posterior is therefore given by $P(S = f|M = f) = \frac{0.96 \times 0.0004}{0.0404} = 0.0095$, or about 0.95%. Although the likelihood has changed our prior belief, we note that our system performs very poorly indeed—in the vast majority of cases we simply cannot trust the system even if it does indicate a forgery. Can you think of the reason for this?

If one really does not have any prior idea of the prevalence of forgeries, one can use an uninformative prior $P(S = f) = 0.5$. In this case the posterior becomes

$$P(S = f|M = f) = \frac{0.96 \times 0.5}{0.96 \times 0.5 + 0.04 \times 0.5} = 0.96$$

which is just the output from the system. Using this prior really means that in the absence of any information from the system one believes that half the population forge signatures. Hopefully this is not the case.

If you find the next exercise somewhat vague, it is on purpose. We cannot do any inference without making assumptions. In this exercise we want you to make what you think are reasonable assumptions, state your assumptions, and explore the consequences of those assumptions. It will be even better if you explore the consequences of different sets of assumptions. There is always some subjective element in the assumptions we make. If however, we agree on the assumptions, we should always come to the same conclusion. Our conclusions should always be objective in that sense.

EXAMPLE 92. You have lost the last digit of the phone number of your friend. So you decide to try a number at random and see whether you get through to your friend. Since you have a random choice between 10 different digits, the chances that you get connected is $\frac{1}{10}$. What are your chances if you allow yourself a second chance?

If you ever encounter a problem where you are not sure how to proceed it is a good idea to list all the possibilities. In this case you may want to first do it for fewer digits, say four. We go straight for ten digits.

Since you are not going to repeat the number already dialed, you may dial any of the following combinations: $(0,1),\ (0,2),\ldots,\ (0,9),\ (1,2),\ldots,\ (1,9),\ldots,\ (7,8),\ (7,9),\ (8,9)$ for a total of $9+8+\cdots+1=45$ combinations. Looking at this list we see that each number appears in nine different combinations. Thus the probability of hitting the right number is $\frac{9}{45}=\frac{1}{5}$.

Let us now reason in a different way. Choosing the first number out of ten possibilities gives us a probability of $\frac{1}{10}$ of being right. For the second number we choose a different one, that is one out of nine possibilities. If we knew that the correct number is among those, it would have given a probability of $\frac{1}{9}$. However, the correct number might have been the first one (the mathematics doesn't care whether the first number was the correct one), the probability that the correct number is among the remaining nine is $\frac{9}{10}$. The total probability of dialing the correct number is therefore $\frac{1}{10}+\frac{1}{9}\frac{9}{10}=\frac{1}{5}$.

Note that you need to dial all ten numbers to ensure that you will get the right one.

EXERCISE 93. We now want you to model the situation and think carefully about your modeling assumptions by approaching the problem in the following systematic manner. Introduce three binary random variables, S and $X_1,\ X_2$. $S=1$ describes a successful connection with your friend. X_1 and X_2 describe the two trials. For example, $P(S=1|X_1=1)=1$. From the joint probability $P(S,X_1,X_2)$ you find the total probability of success

$$\begin{aligned} P(S) &= \sum_{X_1,X_2\in\{0,1\}} P(S,X_1,X_2) \\ &= \sum_{X_1,X_2\in\{0,1\}} P(S|X_1,X_2)P(X_2|X_1)P(X_1). \end{aligned}$$

You are interested in $P(S=1)$. Assign values to the different terms on the right hand side, and note in particular how they derive from your modeling assumptions.

Let us think about specific assumptions. A very reasonable assumption is that you are a rational person, and having discovered that the first number is not the correct one, choose another one among the remaining ones. A less reasonable assumption but perhaps not impossible, is that it simply does not occur to you to discard the

FIGURE 21.2.2. (a) a and b dependent (b) a and b independent, given c.

wrong number, and that you select the second number again from all ten possibilities. Finally, it might just be possible that your friend has a heavy cold and that you are unable to tell, even if you have reached the right number.

Maybe you can think of more scenarios. But once you have decided on a specific scenario, you should be able to come to a definite conclusion.

21.2.1. Conditional independence. It is interesting, and important, to note that additional knowledge may change dependencies into independencies and vice versa. The main idea is that of *conditional* independence . Variables X and Y are conditionally independent given Z, if

$$P(X,Y|Z) = P(X|Z)P(Y|Z), \tag{21.5}$$

or equivalently,

$$P(X|Y,Z) = P(X|Z) \tag{21.6}$$

also written as

$$X \amalg Y|Z.$$

EXERCISE 94. Show that $P(X,Y|Z) = P(X|Z)P(Y|Z) \iff P(X|Y,Z) = P(X|Z)$.

Let us look at few examples illustrating how knowledge about Z can influence our knowledge about the dependencies between X and Y.

EXAMPLE 95. Suppose we have a model where the joint distribution factorizes as

$$P(A,B,C) = P(B|C)P(C|A)P(A). \tag{21.7}$$

One can illustrate these dependencies as in Figure 21.2.2.

Assuming that $C = c$ is observed, it follows from the standard factorizations $P(A, B, C = c) = P(A|B, C = c)P(B|C = c)P(C = c)$ and (21.7) that

$$\begin{aligned} P(A|B, C = c) &= \frac{P(C = c|A)P(A)}{P(C = c)} \\ &= P(A|C = c). \end{aligned}$$

Thus the moment that $C = c$ is observed in this particular model, A and B become conditinally independent.

A practical example of this scenario is, where your gnome depends on the gnomes of your grand parents. Once you know the gnome of your parents, you have all the available knowledge of your own gnome and it becomes independent of that of your grand parents.

EXAMPLE 96. For this model the joint distribution factorizes as follows (see Figure 21.2.3),

$$P(A, B, C) = P(C|A, B)P(A)P(B).$$

If C is unobserved we can marginalize and write

$$\begin{aligned} P(A, B) &= \sum_C P(A, B, C) \\ &= \sum_C P(C|A, B)P(A)P(B) \\ &= P(A)P(B). \end{aligned}$$

This means that as long as C is unobserved, A and B are statistically independent. The moment C is observed we can no longer marginalize (C is locked to its observed value, $C = c$), and A and B have become conditionally dependent.

Again a practical situation is where the genomes of you and your husband are independent. The moment you know the genome of your children, you can infer something about your own genome from the genome of your husband—your genome has become linked to your husband's through knowledge of your children's.

The last example is a particularly vivid illustration of the fact that there need not be any causal relationship between statistically dependent variables. The fact that

(a) (b)

FIGURE 21.2.3. (a) a and b independent. (b) a and b dependent, given c.

you know your children's genome, does not alter the relationship between your or your wife's genome in any sense; physically they remain independent. In probability we are talking about a *logical* relationship. You can infer something about your own genome via the link through your children, by knowing your wife's and your children's genome.

EXAMPLE 97. Let us return to the signature verification example, Example 91, above. The question arises of what happens when your systems rejects a signature as a forgery? Do you immediately call security? What if it is the system that erroneously rejects the signature? In that case you have just lost what might have been a valued customer. Another approach is to verify your system by asking the customer to sign again. Let us analyze this situation.

In general if we have two signatures, we are interested in

$$P(S|M_1, M_2) = \frac{P(M_1, M_2|S)P(S = f)}{P(M_1, M_2)}.$$

So far we have not made any assumptions; in order to make further progress we now need to make an assumption. We assume that the two measurements, M_1 and M_2 are *conditionally* independent, given S. This means that

$$P(M_1, M_2|S) = P(M_1|S)P(M_2|S)$$

in which case we have

$$P(S|M_1, M_2) = \frac{P(M_1|S)P(M_2|S)P(S = f)}{P(M_1, M_2)}.$$

Using the same numerical values as in Example 91, we can now calculate the probability of the signature being a forgery, if the system indicates a forgery both time,

$$P(S = f|M_1 = f, M_2 = f) = \frac{P(M_1 = f|S = f)P(M_2 = f|S = f)P(S = f)}{P(M_1 = f, M_2 = f)}.$$

Since

$$\begin{aligned} P(M_1, M_2) &= \sum_S P(M_1, M_2, S) \\ &= \sum_S P(M_1, M_2|S)P(S) \\ &= \sum_s P(M_1|S)P(M_2|S)P(S), \end{aligned}$$

we find that

$$\begin{aligned} P(S = f|M_1 = f, M_2 = f) &= \frac{0.96 \times 0.96 \times 0.0004}{0.96 \times 0.96 \times 0.0004 + 0.04 \times 0.04 \times 0.9996} \\ &= 0.1873 \end{aligned}$$

which is about 19%. Although we still cannot be too sure, our confidence that it is a forgery has increased after the signature has been rejected twice.

This example is a useful illustration of the difference between independence and *conditional* independence. The fact that we assumed *conditional* independence

$$P(M_1, M_2|S) = P(M_1|S)P(M_2|S)$$

does *not* imply independence, i.e.

$$P(M_1, M_2) = P(M_1)P(M_2)$$

is *not* necessarily true.

Let us first give the intuitive reason. If we know S, i.e. we know whether the signature is a forgery or not, then there is no way that a second measurement can add to what we already know. It means the two measurements are conditionally independent. On the other hand if we don't know whether the signature is a forgery, then our only information comes from the measurements. Thus if the first measurement indicates a forgery, we have a reasonable expectation that the second measurement

(a) (b)

FIGURE 21.2.4. Conditional probability for Example 97.(a) S is not observed. (b) S is observed.

will also indicate a forgery. If not, if the measurements are completely random with no dependencies between the signatures, surely your system is utterly useless. Thus, in the absence of any knowledge about S the measurements are dependent, and $P(M_1, M_2) \neq P(M_1)P(M_2)$.

Let us now discuss the formal explanation, illustrated by the appropriate graphical model. Assuming conditional independence as above, the joint distribution factorizes as follows,

$$P(M_1, M_1, S) = P(M_1|S)P(M_2|S)P(S)$$

with the graphical model given by Figure 21.2.4.

It is an easy exercise to show that, given this graphical model, M_1 and M_2 are conditionally independent given S, and that $M_1$and M_2 are not independent.

21.3. Probability Densities.

In the previous section we looked at discrete random variables. In this section we turn to continuous random variables. With this also comes a change in natation: continous random variables are denoted by lower case; leaving it to the context in to determine whether we are working with a random variable or its realization.

Since the state space is now continous, i.e. the random variable x can now assume values drawn from the continuous set, we need to replace the idea of a probability distribution. After all, the probability of drawing a specific number from a continuum is zero. Thus instead of a probability distribution over a discrete random variable, $P(X)$, we now introduce a continous random variable described by a probability density function $p(x)$. If the probability of a real-valued variable x falling in the interval $(x, x + \delta x)$ is given by $p(x)\delta x$ for $\delta x \to 0$,then $p(x)$ is called the probability

density over x. The probability that x will lie in the interval (a, b) is therefore given by

$$P(x \in (a, b)) = \int_a^b p(x)dx.$$

Note: $p(x)$ satisfies

- $p(x) \geq 0$
- $\int_{-\infty}^{\infty} p(x)dx = 1.$

The sum (marginal)– and product rules become

$$\begin{aligned} p(x) &= \int p(x, y)dy \\ p(x, y) &= p(x|y)p(y). \end{aligned}$$

Change of variables: If $x = g(y)$ we want to figure out how the density functions transform, i.e. how $p_x(x)$ is transformed to $p_y(y)$. For simplicity we only consider the case where $g(y)$ is monotonic, either increasing or decreasing. First consider the case where $g(y)$ monotoninc increasing. The probability that x falls in the range $(x, x+\delta x)$ has to be the same as the probability that y falls in the range $(y, y+\delta y)$, where $g(y)$ maps $(y, y + \delta y)$ to $(x, x + \delta x)$. Accordingly we require that $p_x(x)\delta x = p_y(y)\delta y$, or

$$\begin{aligned} p_y(y) &= p_x(x)\frac{dx}{dy} \\ &= p_x(g(y))g'(y). \end{aligned}$$

If $g(y)$ is monotonic decreasing then $(y, y + \delta y)$ is again mapped to $(x, x + \delta x)$ but in this case $x + \delta x < x$ if $\delta y > 0$. In this case we therefore require that $-p_x(x)\delta x = p_y(y)\delta y$, or

$$\begin{aligned} p_y(y) &= -p_x(x)\frac{dx}{dy} \\ &= - p_x(g(y))g'(y). \end{aligned}$$

Since $g(y)$ is monotonic decreasing, $\frac{dx}{dy} < 0$. Thus for monotonic functions is follows that

$$\begin{aligned} p_y(y) &= p_x(x)\left|\frac{dx}{dy}\right| \\ &= p_x(g(y))\left|g'(y)\right|. \end{aligned}$$

EXAMPLE 98. A transformation function of particular importance is given by

$$x = g(y) = \int_0^y p_y(r)dr. \tag{21.1}$$

It now follows that

$$\begin{aligned} \frac{dx}{dy} &= \frac{dg}{dy} \\ &= p_y(y). \end{aligned}$$

Let us calculate the transformed density function,

$$\begin{aligned} p_x(x) &= p_y(y)\left|\frac{dy}{dx}\right| \\ &= p_y(y)\left|\frac{1}{p_y(y)}\right| \\ &= 1. \end{aligned}$$

Let us say you have available to you a method that draws unbiased random samples from a uniform distribution, this transformation then allows you to draw unbiased random samples from any other density function.

In image processing this can be used to enhance the contrast in an image, through histogram equilization. Given a histogram $p_y(y)$ (normalized so that $\sum_y p_y(y) = 1$) the idea is to transform it so that $p_x(x) = 1$. This just the transformation given above. If the image has L gray levels, i.e. y is the discrete variable with values $y_k,\ k = 0, \ldots, L-1$, and there are n pixels in the image, then $p_y(y_k) = \frac{n_k}{n}$ where n_k is the number of pixels that have gray level y_k. The discrete version of the transform

(a) (b)

FIGURE 21.3.1. (a) Original image. (b) After histogram equalization.

(21.1) is then given by

$$\begin{aligned} x_k &= g(y_k) \\ &= \sum_{r=0}^{k} p_y(r) \\ &= \sum_{r=0}^{k} \frac{n_r}{n}, \ k = 0, \ldots, L-1. \end{aligned}$$

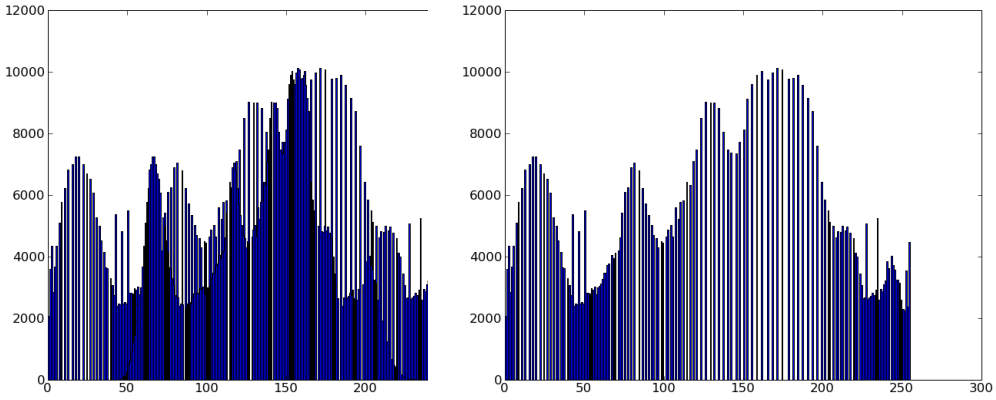
The transformed image is obtained by mapping each pixel with gray level y_k to a pixel with gray level x_k. The result is shown in Figure 21.3.1. Notice how the contrast is enhanced by the histogram equalization.

The histograms are given by the following Figure. Do you have any explanation why the histogram after equalization does not show a flat curve?

21.4. Expectation and Covariances.

Discrete expectation:

$$\mathbb{E}[f] = \sum_{x} P(x) f(x).$$



(a) (b)

FIGURE 21.3.2. (a) Histogram of original image. (b) Histogram of equalized image.

Continuous expectation:

$$\mathbb{E}[f] = \int p(x)f(x)dx.$$

Note: If we have N points $x_n,\ n = 1, \ldots, N$ drawn from the probability density $p(x)$ the expectation can be approximated by-

$$E[f] \approx \frac{1}{N}\sum_{n=1}^{N} f(x_n).$$

This approximation becomes exact for $N \to \infty$. Note that formally what we are doing is to approximate $p(x)$ with samples drawn from it. There are different strategies for doing this, see for example Chapter 11 of Bishop's book.

Conditional expectation:

$$\mathbb{E}[f|y] = \sum_{x} P(x|y)f(x).$$

Variance:

$$\text{var}[f] = \mathbb{E}[(f(x) - \mathbb{E}[f(x)])^2].$$

Covariance:

$$\begin{aligned} cov[x,y] &= \mathbb{E}_{x,y}\left[\{x-\mathbb{E}[x]\}\{y-\mathbb{E}[y]\}\right] \\ &= \mathbb{E}_{x,y}[xy]-\mathbb{E}[x]\mathbb{E}[y]. \end{aligned}$$

For vector variables this becomes

$$\text{cov}[\mathbf{x},\mathbf{y}] = \mathbb{E}_{x,y}\left[\{\mathbf{x}-\mathbb{E}[\mathbf{x}]\}\{\mathbf{y}-\mathbb{E}[\mathbf{y}]\}^T\right] = \mathbb{E}_{x,y}[\mathbf{x}\mathbf{y}^T]-\mathbb{E}[\mathbf{x}]\mathbb{E}[\mathbf{y}^T].$$

A particularly important case of the covariance matrix is $\text{cov}[\mathbf{x},\mathbf{x}]$. If $\mathbf{x}$ is a d-dimensional vector then $\text{cov}[\mathbf{x},\mathbf{x}]$ is a $d\times d$, symmetric, positive semi-definite matrix. This implies that all its eigenvalues are real, moreover all its eigenvalues are non-negative.

The only 'problem' with this definition is that we seldom know the probability density function $p(\mathbf{x})$. In fact the learning part of machine learning is to estimate or approximate the pdf from data. In general we observe samples that we assume are generated by the pdf, from these samples we have to estimate, ideally the pdf, or perhaps the mean and covariance. We have already given the expression for the sample estimate for the mean. The (biased) sample estimate for the covariance matrix, given N samples $\mathbf{x}_n, n=1,\ldots,N$ is

$$\Sigma = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n-\mu)(\mathbf{x}_n-\mu)^T,$$

where μ is the mean.

EXERCISE 99. Show that the sample covariance matrix Σ is symmetric and positive semi-definite, i.e. that it is symmetric and all its eigenvalues are non-negative.

21.5. Decision Theory.

In the signature verification example we referred to a decision boundary that allows one to accept a signature, or reject it as a forgery. The question is how does one find the optimal decision boundary. We classify a given signature x into one of two classes $C_k,\ k=0,1$ and we are interested in the probability of class C_k given a signature x, i.e. we are interested in $P(C_k|x)$. Using Bayes theorem we can write

this as

$$P(C_k|x) = \frac{p(x|C_k)P(C_k)}{p(x)}.$$

The prior probability $P(C_k)$ tells us the probability of class C_k in the absence of any further information. It simply tells us how likely it is that any given signature is a forgery. This represents our (subjective) belief in the honesty of the population and may for example be based on crime statistics. $p(C_k|x)$ is the revised, posterior estimate, based on the actual measurement x. The class-based likelihood $p(x|C_k)$ requires knowledge about our system.

We want to minimize the chances of assigning x to the wrong class. Intuitively we assign x to the class with the highest posterior probability. Let us make this precise.

21.5.1. Minimizing misclassification. Two types of errors are possible, accepting a forgery, or rejecting a genuine signature. In general an input vector belonging to C_0 can be wrongly assigned to class C_1, or vice versa. The probability of a mistake is therefore given by

$$P(\text{mistake}|x) = \begin{cases} P(C_0|x) & \text{if we choose } C_1 \\ P(C_1|x) & \text{if we choose } C_0 \end{cases}. \tag{21.1}$$

It may be even better to minimize the probability of a mistake, averaged over all possible input values, i.e. minimize the marginal probability,

$$\begin{aligned} P(\text{mistake}) &= \int P(\text{mistake}, x)dx \\ &= \int_{R_0} P(\text{mistake}, x)dx + \int_{R_1} P(\text{mistake}, x)dx \\ &= \int_{R_0} P(\text{mistake}|x)p(x)dx + \int_{R_1} P(\text{mistake}|x)p(x)dx \end{aligned}$$

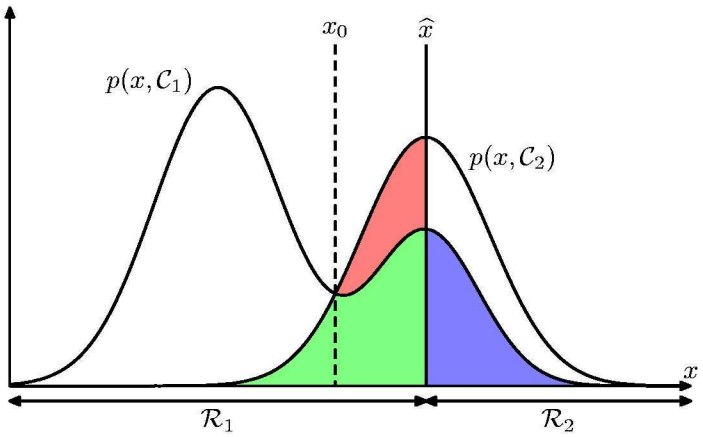


FIGURE 21.5.1. Optimal decision boundary.

where R_k is the region where all input vectors x are assigned to C_k, see Figure 21.5.1. Making use of (21.1) this can be written as

$$\begin{aligned} P(\text{mistake}) &= \int_{R_0} P(C_1|x)p(x)dx + \int_{R_1} P(C_0|x)p(x)dx \\ &= \int_{R_0} p(x|C_1)P(C_1)dx + \int_{R_1} p(x|C_0)P(C_0)dx \end{aligned}$$

where the last line follows from Bayes' theorem. This is minimized if x is assigned to the region for which the integrand is the smallest, as illustrated in Figure 21.5.1.

Note that it is not possible to make an informed decision in the absence of knowledge of our system. In fact, following this approach we need to know quite a lot: essentially we need to know the joint densities $p(x, C_k)$. The 'learning' part of machine learning is about inferring joint densities or class conditional densities from the data. Signature verification is hard because we do not have a model for forgeries—a forgery is everything that is not a genuine signature. The reason why it is at all possible to design efficient systems is because we are able to design a good models for the genuine signatures.

21.5.2. Minimizing expected loss. If there is a penalty involved in a particular misclassification it can be modeled by a loss function. Commercial banks for instance would rather accept more forgeries than alienating their customers by

rejecting genuine signatures—there is a heavy penalty for rejecting a genuine signature.

Let us generalize a little and assume that there are N different classes (or states of nature), and a different actions $\alpha_1, \ldots, \alpha_N$ every time we observe x. Action α_j for example, may indicate: put x in class C_j. The posterior is given as usual by

$$P(C_j|x) \propto p(x|C_j)P(C_j).$$

Let λ_{ij} expresses the penalty for taking action α_i when the true state is C_j. The loss or penalty for taking action α_i having observed x, is the average penalty

$$R(\alpha_i|x) = \sum_{j=1}^{N} P(C_j|x)\lambda_{ij}.$$

Our decision rule is a function $\alpha(x)$ that maps each x to an action α_i. The average penalty is minimized if, for each x choose

$$\alpha(x) = \arg\min_{\alpha_i} R(\alpha_i|x).$$

Note that this also minimizes the overall risk, defined as the expected loss for a given decision rule,

$$R = E\left[R(\alpha(x)|x)\right] = \int R(\alpha(x)|x)p(x)dx.$$

21.5.3. Reject option. If it is essential that a correct classification be made, one might consider a reject option for samples falling in the ambiguous region close to the decision boundary.

21.5.4. Inference and decision. Instead of two different stages, inference and decision, one might opt for a *discriminant* function. Three approaches

(1) Calculate the posterior density $p(C_K|x)$ from the class conditional densities $p(x|C_k)$. Then use decision theory, i.e. incorporate the possible risk. This is a so-called *generative* model. The name stems from the fact that, since one builds a probabilistic model of each class, it is possible to use the model to generate data by drawing samples from the probability distributions. The main drawback of this approach is the danger of overkill. We may not be

interested in a detailed model of each class, in which case it might be better to use the available data to directly achieve your aim.

(2) Find $p(C_k|x)$ directly, then use decision theory. This is a so-called *discriminative* model. Instead of using the data to build a model of each class, the available data is used to estimate parameters that directly gives the probability of class membership. Note that any penalty funtions are built into the model.

(3) Find a function $f(x)$ that maps x directly onto a class label. This is perhaps the simplest approach. The main drawback is that one does not get a confidence value with the estimate and therefore has no indication to what extent the class assignment can be trusted.

Combining models. Suppose we have two independent test of a quantity, for example using both a signature and fingerprint for personal identification. If the two tests are (conditionally) independent we write

$$p(x_1, x_2|C_k) = p(x_1|C_k)p(x_2|C_k).$$

the posterior density is then given by

$$\begin{aligned} p(C_k|x_1, x_2) &\propto p(x_1, x_2|C_k)p(C_k) \\ &= p(x_1|C_k)p(x_2|C_k)p(C_k) \\ &\propto \frac{p(C_k|x_1)p(C_k|x_2)}{p(C_k)}. \end{aligned}$$

It is always a good idea to use different, independent models. Note that this also works if one obtains a second conditionally independent measurement using the same model.

CHAPTER 22

PROBABILITY DENSITY FUNCTIONS

22.1. Introduction.

We introduce various probability density functions, the most important one is the Gaussian or normal density function. It does show up regularly in practical situations but its main attraction is its analytical properties. Operations that are otherwise intractable can be evaluated analytically in case of Gaussian.

22.2. Binary Variables.

22.2.1. The Bernoulli distribution. If $X \in \{0, 1\}$ is a discrete random variable such that $P(X = 1) = \mu$, implying $P(X = 0) = 1 - \mu$, then the probability distribution $P(X|\mu)$ is given by the Bernoulli distribution,

$$P(X|\mu) = \text{Bern}(X|\mu) = \mu^X (1 - \mu)^{1-X}. \tag{22.1}$$

EXERCISE 100. Show that $\mathbb{E}[X] = \mu$, and $\text{var}[X] = \mu(1 - \mu)$.

The goal is to estimate μ from data, $\mathcal{D} = \{x_1, \ldots, x_N\}$. If we draw the samples independently, the likelihood function of μ is given by

$$P(\mathcal{D}|\mu) = \prod_{n=1}^{N} P(x_n|\mu) = \prod_{n=1}^{N} \mu^{x_n} (1 - \mu)^{1-x_n}. \tag{22.2}$$

If we observe $X = 1$, m times, then the likelihood becomes

$$P(\mathcal{D}|\mu) = \mu^m (1 - \mu)^{N-m}.$$

Maximizing the log-likelihood,

$$\ln P(\mathcal{D}|\mu) = m \ln \mu + (N - m) \ln(1 - \mu),$$

gives

$$\mu_{ML} = \frac{m}{N}.$$

Note: If $N = 3$ and we observe $x = 1$ for all three samples, then the maximum likelihood estimate gives $\mu_{ML} = 1$, which is nonsense. The problem is that we are working with a point estimate of μ, and not a probability distribution over μ. We return to this problem later when we introduce a prior distribution over μ.

22.2.2. The binomial distribution. If we observe $x = 1$, m times out of N observations, it follows from (22.2) that the probability of m ones out of a sequence of N trials is $\mu^m(1-\mu)^{N-m}$. Since there are $\begin{pmatrix} N \\ m \end{pmatrix} = \frac{N!}{(N-m)!m!}$ ways of getting m ones out of a sequence of N trials, the total total probability of getting m ones out of a sequence of N trials, is given by the *binomial* distribution

$$\text{Bin}(m|N,\mu) = \begin{pmatrix} N \\ m \end{pmatrix} \mu^m(1-\mu)^{N-m}. \tag{22.3}$$

Note that $\sum_{m=0}^{N} \text{Bin}(m|N,\mu) = 1$.

It can be shown that

$$\begin{aligned} \mathbb{E}[m] &= N\mu \\ \text{var}[m] &= N\mu(1-\mu). \end{aligned}$$

The following example asks questions based directly on the data that is observed. The answers are therefore obtained directly from the data.

EXAMPLE 101. An urn contains K balls, of which B are black and $W = K - B$ are white. You draw a ball at random, and replaces it, N times. Let us calculate the probability distribution of the number of times a black ball is drawn n_B, as well as the expectation and variance of n_B,

$$P(n_B|\mathcal{D}) = \begin{pmatrix} N \\ n_B \end{pmatrix} \mu^{n_B}(1-\mu)^{N-n_B}$$

where $\mu = \frac{B}{K}$,

$$\mathbb{E}[n_B] = N\mu \text{ and } \text{var}[n_B] = N\mu(1-\mu).$$

You might also, for example, be interested in the quantity

$$z = \frac{(n_B - N\mu)^2}{N\mu\,(1-\mu)}.$$

Since n_B is a random variable, z is also a random variable and we can calculate its expected value,

$$\mathbb{E}\,[z] = \frac{\mathbb{E}\left[(n_B - N\mu)^2\right]}{N\mu\,(1-\mu)} = 1$$

since the numerator is the variance of n_B.

In the next example we are still using a generative model, i.e. we again use a model that describes a process that generates data, but this time we ask questions about latent quantities, i.e. quantities not directly observed. For this Bayes' theorem is invariably required.

EXAMPLE 102. We now have eleven urns $\mathcal{C}_j,\ \ j = 0, \ldots, 10$, each one containing ten balls, either white or black. Suppose the jth urn contains j black balls, and $10 - j$ white balls. One urn is now selected at random and N balls are drawn from it, with replacement. You don't know which urn is selected but you know that n_B black balls, and $N - n_B$ white balls are drawn. What is the probability that the balls are drawn from urn $\mathcal{C}_u$?

Each urn can be modeled as,

$$P(n_B|\mathcal{C}_j) = \left(\begin{array}{c} N \\ n_B \end{array} \right) \mu_j^{n_B} \left(1 - \mu_j\right)^{N - n_B}, \ \ j = 0, \ldots, 10,$$

where $\mu_j = \frac{j}{10}$. In this case however, the information we need is not directly accessible from the data. The question is, given n_B, what is the probability of $\mathcal{C}_u$, i.e. what is $P(\mathcal{C}_u|n_B)$? Using Bayes' theorem we can invert the posterior to get

$$P(\mathcal{C}_u|n_B) = \frac{P(n_B|\mathcal{C}_u)P(\mathcal{C}_u)}{P(n_B)}.$$

The prior probability $P(\mathcal{C}_u) = \frac{1}{11}$, and the marginal,

$$P(n_B) = \sum_j P(n_B|\mathcal{C}_j)P(\mathcal{C}_j).$$

EXERCISE 103. Assuming that in the previous problem $N = 10$ and $n_B = 3$, calculate $P(\mathcal{C}_u|n_B)$ for $u = 0, \ldots, 10$. Now you draw another ball from the same urn. What is the probability that the next ball is black? Hint: You need to calculate $P(\text{ball}_{N+1} \text{ is black}|n_B, N)$. Write this as the marginal

$$P(\text{ball}_{N+1} \text{ is black}|n_B, N) = \sum_u P(\text{ball}_{N+1} \text{ is black}, u|n_B, N).$$

Factorize this in the usual way and note that

$$P(\text{ball}_{N+1} \text{ is black}|u, n_B, N) = P(\text{ball}_{N+1} \text{ is black}|u).$$

This means that ball_{N+1} is black is conditionally independent of n_B and N, given that we know it is drawn from urn $\mathcal{C}_u$.

Please note carefully the reasoning in this problem—by marginalizing you are taking the average over all urns. Let us see how your answer differs if instead, you simply use the fact that $n_B = 3$, to determine the most likely urn, finding that it is urn $u = 3$. Drawing the next ball from urn $u = 3$, gives a probability of 0.3 of drawing a black ball. This answer is not as good as the one you calculated above, because it does not take the uncertainty of the urn into account.

22.2.3. The beta distribution. Given the data $\mathcal{D}$ in (22.2) we calculated the maximum likelihood estimate of μ using the likelihood $p(\mathcal{D}|\mu)$. We noted that this can run into problems for a small number of samples. Let us therefore examine the possibility of maximizing the posterior probability $p(\mu|\mathcal{D})$ instead. This also has a more philosophical interpretation. Instead of the point estimate we get from maximizing the likelihood, we now end up with a probability distribution over μ, given the data. In order to do so a 'subjective' element is introduced in the form of a prior over μ. To wit, according to Bayes' theorem the posterior is proportional to $p(\mathcal{D}|\mu)p(\mu)$, where $p(\mu)$is the prior. The question is how to choose a suitable prior—the subjective element. In the absence of other considerations, it is *convenient* to choose it in such a way so that the posterior distribution has the same functional form as the prior, referred to as a *conjugate* prior. A suitable conjugate prior is the beta distribution

$$\text{Beta}(\mu|a, b) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}\mu^{a-1}(1-\mu)^{b-1} \tag{22.4}$$

where the gamma function is defined as

$$\Gamma(x) = \int_0^\infty u^{x-1} e^{-u} du,$$

and

$$\begin{aligned} \mathbb{E}[\mu] &= \frac{a}{a+b} \\ \text{var}[\mu] &= \frac{ab}{(a+b)^2(a+b+1)}. \end{aligned}$$

Given that $x = 1$ is observed m times in a trial of N samples, and setting $l = N - m$ (the number of times $x = 0$ is observed) the posterior is proportional to

$$p(\mu|m, l, a, b) \propto \mu^{m+a-1}(1-\mu)^{l+b-1}.$$

This is another (unormalized) beta distribution and properly normalized (using (22.4)) the posterior becomes,

$$p(\mu|m, l, a, b) = \frac{\Gamma(m+a+l+b)}{\Gamma(m+a)\Gamma(l+b)} \mu^{m+a-1}(1-\mu)^{l+b-1}.$$

Starting with the prior note how it is modified by the subsequent observations: a is increased by the number of times $X = 1$ is observed, and b is increased by the number of times $X = 0$ is observed. The effect of the prior is as if we had a certain number of prior observations of $X = 0$ and $X = 1$. This prior 'belief' is then modified in view of the actual observations received.

Note (important): The posterior can be used as a prior if subsequent observations are made. This allows for sequential estimates.

Let us be a little more precise and compute the probability of observing $X = 1$ in view of the data $\mathcal{D}$, by marginalizing over the joint distribution $P(X = 1, \mu|\mathcal{D})$,

$$P(X = 1|\mathcal{D}) = \int_0^1 P(X = 1|\mu) p(\mu|\mathcal{D}) d\mu = \int_0^1 \mu p(\mu|\mathcal{D}) d\mu = \mathbb{E}[\mu|\mathcal{D}].$$

Using the mean of the beta distribution, we get

$$P(X = 1|\mathcal{D}) = \frac{m+a}{m+a+l+b}.$$

The interpretation of the prior is now clear: one can think of a and b as fictitious observations prior to any actual observations (of $X = 1$, and $X = 0$ respectively).

The probability of observing observing $X = 1$ is the fraction of the total number of times $m + a$, including the prior 'observations', out of a total of $m + a + l + b$ observations, again including the prior 'observations'.

It is important to note the shift in point of view: when we wrote down the Bernoulli distribution, all the way through to the maximum likelihood estimate of μ from the binomial distribution, μ was just a parameter with $P(X = 1|\mu) = \mu$. Having introduced the prior, we get a probability distribution over μ. μ has become a random variable itself, and in order to find $P(X = 1|\mathcal{D})$ it has become necessary to marginalize over μ, as we did above.

Finally note that our estimate of $P(X = 1|\mathcal{D})$ (using a prior) agrees with the maximum likelihood estimate as $m,\ l \to \infty$.

EXERCISE 104. Suppose that you observe $X = 1$ three times out of three trials. Using maximum likelihood, we find that $\mu = 1$. Introducing a prior, investigate how the estimate $P(X = 1|\mathcal{D})$ is modified by the presence of data. Suppose you have reason to believe that your coin is balanced, how do you assign values to the parameters of the prior? Suppose this prior belief turns out to be wrong, how does subsequent observations correct this mistaken prior belief? Any idea how many observations are required in order to correct a mistaken prior belief?

22.3. Multinomial Variables.

Variables that can take on one of K mutually exclusive states can be represented by a K-dimensional vector of the form,

$$\mathbf{x} = [0 \ \cdots \ 0 \ 1 \ 0 \ \cdots \ 0]^T$$

where the position of the 1 indicates the specific state. If $P(x_k = 1) = \mu_k$, then

$$P(\mathbf{x}|\boldsymbol{\mu}) = \prod_{k=1}^{K} \mu_k^{x_k}$$

where $\boldsymbol{\mu} = [\mu_1, \ldots, \mu_K]^T,\ \mu_k \geq 0$ and $\sum_k \mu_k = 1$.

Note:

$$\sum_{\mathbf{x}} P(\mathbf{x}|\boldsymbol{\mu}) = \sum_{k=1}^{K} \mu_k = 1$$

and

$$\mathbb{E}[\mathbf{x}|\boldsymbol{\mu}] = \sum_{\mathbf{x}} p(\mathbf{x}|\boldsymbol{\mu})\mathbf{x} = [\mu_1, \ldots, \mu_K]^T.$$

For a data set $\mathcal{D}$ consisting of N independent observations, the likelihood is,

$$p(\mathcal{D}|\boldsymbol{\mu}) = \prod_{n=1}^{N}\prod_{k=1}^{K}\mu_k^{x_{nk}} = \prod_{k=1}^{K}\mu_k^{(\sum_n x_{nk})} = \prod_{k=1}^{K}\mu_k^{m_k},$$

where

$$m_k = \sum_n x_{nk},$$

the number of observations of $x_k = 1$. Maximizing the log-likelihood $\ln p(\mathcal{D}|\boldsymbol{\mu})$ subject to the constraint $\sum_k \mu_k = 1$, is achieved through a Lagrange multiplier, i.e. by maximizing

$$\sum_{k=1}^{K} m_k \ln \mu_k + \lambda\left(\sum_{k=1}^{K}\mu_k - 1\right).$$

Setting the partial derivative with respect to μ_k equal to zero, gives

$$\mu_k = -\frac{m_k}{\lambda},$$

and the constraint gives

$$\lambda = -N.$$

Thus

$$\mu_k^{ML} = \frac{m_k}{N},$$

and the multinomial distribution is given by

$$\text{Mult}(m_1, \ldots, m_K|\boldsymbol{\mu}, N) = \begin{pmatrix} N \\ m_1 \cdots m_K \end{pmatrix}\prod_{k=1}^{K}\mu_k^{m_k},$$

where

$$\sum_{k=1}^{K} m_k = N.$$

22.3.1. Dirichlet distribution. The Dirichlet distribution is the conjugate prior for the multinomial distribution

$$\mathrm{Dir}(\boldsymbol{\mu}|\boldsymbol{\alpha}) = \frac{\Gamma(\alpha_0)}{\Gamma(\alpha_1)\cdots\Gamma(\alpha_K)}\prod_{k=1}^{K}\mu_k^{\alpha_k-1}$$

where

$$\alpha_0 = \sum_{k=1}^{K}\alpha_k,$$

and

$$\sum_{k=1}^{K}\mu_k = 1.$$

22.4. Model comparison.

It is not always clear what underlying process generates the observed data. There might be competing hypotheses and one would like a systematic procedures to make an informed decision.

Suppose we observe N trials of a binary process, i.e. we are given a sequence $\mathbf{s}$ of N zero's and ones. One hypothesis, let us call it $\mathcal{H}_1$, is that the sequence is generated by a biased coin. We are not given the bias and need to estimate it from the data. We also want to predict the outcome of the next trial. This is of course the problem already encountered in Section 22.2.

The probability of observing m ones and $n = N - m$ zero's is given by

$$P(\mathbf{s}|\mu, N, \mathcal{H}_1) = \mu^m(1-\mu)^n,$$

where the dependence on our hypothesis is explicitly stated. Let us also assume a uniform prior, $p(\mu|\mathcal{H}_1) = 1$. There is nothing too special about this choice—the choice of prior is part of our assumptions and we cannot do inference without assumptions. Here we make this choice more as a matter of convenience; we might have chosen a conjugate prior, the beta distribution. We want to infer μ where $P(X = 1|\mu, \mathcal{H}_1) = \mu$, and predict the outcome of the next trial. Again it might be useful to note that, since predictions are never certainties, they are always expressed in terms of probabilities. Assuming that $\mathcal{H}_1$ is true, the posterior probability is given

by

$$\begin{aligned} p(\mu|\mathbf{s}, N, \mathcal{H}_1) &= \frac{P(\mathbf{s}|\mu, N, \mathcal{H}_1)p(\mu|\mathcal{H}_1)}{P(\mathbf{s}|N, \mathcal{H}_1)} \\ &= \frac{P(\mathbf{s}|\mu, N, \mathcal{H}_1)}{P(\mathbf{s}|N, \mathcal{H}_1)}, \end{aligned} \tag{22.1}$$

using the uniform prior. Since the likelihood of the parameter μ is known, the posterior is given by

$$p(\mu|\mathbf{s}, N, \mathcal{H}_1) = \frac{\mu^m(1-\mu)^n}{P(\mathbf{s}|N, \mathcal{H}_1)},$$

where the normalization constant (also called the *evidence*, for reasons that will become clear shortly) is given by

$$\begin{aligned} P(\mathbf{s}|N, \mathcal{H}_1) &= \int_0^1 \mu^m(1-\mu)^n d\mu \\ &= \frac{\Gamma(m+1)\Gamma(n+1)}{\Gamma(m+n+2)} \\ &= \frac{m!n!}{(m+n+1)!}. \end{aligned}$$

Given our hypothesis and the observed data, the posterior probability distribution of μ is therefore given by

$$p(\mu|\mathbf{s}, N, \mathcal{H}_1) = \frac{(m+n+1)!}{m!n!}\mu^m(1-\mu)^n.$$

EXERCISE 105. Recall that the maximum likelihood estimate gives $\mu = \frac{m}{N}$. Calculate the most probable value of μ from the posterior, i.e. calculate the value of μ that maximizes the posterior (this is also called the maximum a posteriori (MAP) estimate). What is the mean of μ under this distribution? How does it compare with the maximum likelihood estimate?

Turning to prediction, we now calculate the probability $P(X = 1|\mathbf{s}, N)$. The sum rule gives,

$$\begin{aligned} P(X = 1|\mathbf{s}, N, \mathcal{H}_1) &= \int_0^1 p(X = 1, \mu|\mathbf{s}, N, \mathcal{H}_1)d\mu \\ &= \int_0^1 P(X = 1|\mu, \mathbf{s}, N, \mathcal{H}_1)p(\mu|\mathbf{s}, N, \mathcal{H}_1)d\mu. \end{aligned}$$

Since the probability of observing $X = 1$ is μ, the prediction for observing $X = 1$ is given by

$$\begin{aligned} P(X = 1|\mathbf{s}, N, \mathcal{H}_1) &= \int_0^1 \mu \frac{(m+n+1)!}{m!n!} \mu^m (1-\mu)^n d\mu \\ &= \frac{(m+n+1)!}{m!n!} \int_0^1 \mu^{m+1} (1-\mu)^n d\mu \\ &= \frac{(m+n+1)!}{m!n!} \frac{(m+1)!n!}{(m+n+2)!} \\ &= \frac{m+1}{m+n+2}. \end{aligned}$$

Now suppose we have a second hypothesis $\mathcal{H}_0$ of how the sequence $\mathbf{s}$ is generated. Suppose this second hypothesis states that the sequence is generated by casting a normal, six-sided dice, with five sides painted 'zero' and one side painted 'one'. How do we decide between these two alternative hypotheses, is $\mathcal{H}_0$ more probable that $\mathcal{H}_1$?

Also note that $\mathcal{H}_0$ does not have any free parameters whereas $\mathcal{H}_1$ has one, namely $\mu \in [0, 1]$.

Using Bayes' theorem again,

$$P(\mathcal{H}_1|\mathbf{s}, N) = \frac{P(\mathbf{s}|N, \mathcal{H}_1) P(\mathcal{H}_1)}{P(\mathbf{s}|N)}$$

and

$$P(\mathcal{H}_0|\mathbf{s}, N) = \frac{P(\mathbf{s}|N, \mathcal{H}_0) P(\mathcal{H}_0)}{P(\mathbf{s}|N)}.$$

The normalization constant is the same for both models, and if we only consider these two models it is given by

$$P(\mathbf{s}|N) = P(\mathbf{s}|N, \mathcal{H}_0) P(\mathcal{H}_0) + P(\mathbf{s}|N, \mathcal{H}_1) P(\mathcal{H}_1).$$

In order to evaluate the posterior probabilities we need to assign the prior probabilities $P(\mathcal{H}_0)$ and $P(\mathcal{H}_1)$. Without any additional information we might set both to $\frac{1}{2}$. The two data dependent terms indicate how much the data favor the two hypotheses; this is also referred to as the *evidence* for the model. Note that it appeared as the normalization constant in (22.1)when we inferred μ from the data.

The evidence for model $\mathcal{H}_0$ is very simple because it has no free parameters

$$P(\mathbf{s}|N, \mathcal{H}_0) = \mu_0^m (1-\mu_0)^n$$

where $\mu_0 = \frac{1}{6}$. The ratio of the two models becomes,

$$\begin{aligned} \frac{P(\mathcal{H}_1|\mathbf{s}, N)}{P(\mathcal{H}_0|\mathbf{s}, N)} &= \frac{P(\mathbf{s}|N, \mathcal{H}_1)}{P(\mathbf{s}|N, \mathcal{H}_0)} \\ &= \frac{m!n!}{(m+n+1)!} / \mu_0^m (1-\mu_0)^n. \end{aligned}$$

EXERCISE 106. Suppose you observe the following sequence, which one of the two models is the more probable,

$$01110001010011100101011010010010010100110101101100010?$$

It should be no surprise that the model $\mathcal{H}_1$with the one free parameter is favored over the model $\mathcal{H}_0$ with no free parameters. The reason is that free parameters can be adjusted to favor the data at least to some extent. Such models are therefore seldom totally unlikely.

EXAMPLE 107. A tale of three prisoners.

In a cruel, far-away country three prisoners find themselves in a cell awaiting their lot. Two of them will be set free the next morning but the third one will go free. The warden draws the name of the prisoner to be executed from a uniform distribution, i.e. each prisoner has a $\frac{1}{3}$ chance of being selected for execution. The warden draws the name but it will only be announced the next morning.

The first prisoner quite agitated, reasons that at least one of the other two prisoners will be released, and he begs the warden to tell him the name of one of the others that will be released. The warden gives him a name. Then the first prisoner realizes that he might have made a mistake because, according to his reasoning reasoning, there are only two of them left, with equal probabilities of being executed. His chances of being executed has just gone up from $\frac{1}{3}$ to $\frac{1}{2}$. Or is he wrong?

Let us introduce three different hypotheses, $\mathcal{H}_1$, $\mathcal{H}_2$ and $\mathcal{H}_3$ where $\mathcal{H}_i$ is the hypothesis that prisoner i, $\quad i = 1,\ 2\ , 3$ is executed. Let $\mathcal{D}$ denotes the available data, i.e. the answer provided by the warden. The three hypotheses have the same prior probabilities, $P(\mathcal{H}_i) = \frac{1}{3}$, $\quad i = 1, \ldots, 3$. Since prisoner one asked the question,

we are interested in $P(\mathcal{H}_1|\mathcal{D})$. Again using Bayes' theorem we find that

$$P(\mathcal{H}_1|\mathcal{D}) = \frac{P(\mathcal{D}|\mathcal{H}_1)P(\mathcal{H}_1)}{P(\mathcal{D})}.$$

The response $\mathcal{D}$ from the warden is either $\mathcal{D} = 2$ or $\mathcal{D} = 3$. For hypothesis $\mathcal{H}_1$ (both prisoners 2 and 3 are being released) the warden has a choice. Let us assume that he chooses randomly between them (no bias). This means that $P(\mathcal{D} = 2|\mathcal{H}_1) = \frac{1}{2} = P(\mathcal{D} = 3|\mathcal{H}_1)$. In the case of $\mathcal{H}_2$ or $\mathcal{H}_3$, the warden does not have a choice, and it follows that

$$P(\mathcal{D} = 2|\mathcal{H}_2) = 0 = P(\mathcal{D} = 3|\mathcal{H}_3),$$
$$P(\mathcal{D} = 3|\mathcal{H}_2) = 1 = P(\mathcal{D} = 2|\mathcal{H}_3).$$

This allows us to compute

$$\begin{aligned} P(\mathcal{D}) &= \sum_{i=1}^{3} P(\mathcal{D}|\mathcal{H}_i)P(\mathcal{H}_i) \\ &= \frac{1}{3}\sum_{i=1}^{3} P(\mathcal{D}|\mathcal{H}_i). \end{aligned}$$

Thus

$$\begin{aligned} P(\mathcal{D} = 2) &= \frac{1}{3}\left[P(\mathcal{D} = 2|\mathcal{H}_1) + P(\mathcal{D} = 2|\mathcal{H}_2) + P(\mathcal{D} = 2|\mathcal{H}_3)\right] \\ &= \frac{1}{3}\left[\frac{1}{2} + 0 + 1\right] \\ &= \frac{1}{2}, \end{aligned}$$

similarly

$$P(\mathcal{D} = 3) = \frac{1}{2}.$$

Thus we find that

$$\begin{aligned} P(\mathcal{H}_1|\mathcal{D}) &= \frac{\frac{1}{2} \times \frac{1}{3}}{\frac{1}{2}} \\ &= \frac{1}{3}, \end{aligned}$$

for either $\mathcal{D} = 2$ or $\mathcal{D} = 3$. Thus, perhaps surprisingly, the additional information does not affect the chances of prisoner 1 to be released at all. This does not mean that he slept any easier.

EXERCISE 108. Let us say the prisoners learn that prisoner 3 is being released. How does this affect the chances of prisoner 2?

EXAMPLE 109. At a crime scene blood samples from two individuals are found. After testing, the samples turn out to be of type O, a relative common blood type (about 60% of the population), and type AB, a rather rare blood type (about 1% of the population). A suspect is tested and his blood type is found to be of type O. What is the probability, based on blood types, that he was at the crime scene?

We are dealing with two different hypotheses. Let us denote the hypothesis, 'the suspect and one other unknown person was at the crime scene', by S, and the alternative, 'two unknown people were present at the crime scene', by $\overline{S}$. In this case the priors are the prior probabilities of the two scenarios, of which we might not know much. Let us therefore concentrate on what can be learned from the data or, equivalently, assign the same prior probabilities to the two scenarios. Given S, i.e. the suspect was present at the crime scene, the probability of the data is

$$P(\mathcal{D}|S) = p_{AB},$$

the probability of the blood type AB. (We are given that the suspect was present, thus the presence of blood type O is a given.) The probability of the data given $\overline{S}$, is

$$P(\mathcal{D}|\overline{S}) = 2p_O p_{AB}.$$

The presence of the factor 2 might cause some confusion. We need to take into account all possible scenarios that explain the two different blood types, subject to the hypothesis. Here the hypothesis is that two unknown individuals are present, one with blood type O and one with blood type AB. There are two ways in which that can happen, given two individuals.

The likelihood ration is therefore

$$\frac{P(\mathcal{D}|S)}{P(\mathcal{D}|\overline{S})} = \frac{1}{2p_O} = 0.83.$$

This analysis provides weak evidence, *against* the suspect being present at the crime scene.

This result looks reasonable. If, for instance, the analysis provided evidence *for* the suspect being present at the crime scene, the same analysis would then indicate that *all* persons with blood type are suspect, 60% of the population.

EXERCISE 110. Suppose the suspect has blood type AB, what is the probability that he was present at the crime scene.

EXERCISE 111. Consider the following (actual) statement:

> When spun on edge 250 times, a Belgian one-euro coin came up heads 140 times and tails 110. 'It looks very suspicious to me', said Barry Blight, a statistics lecturer at the London School of Economics. 'If the coin were unbiased the chance of getting a result as extreme as that would be less than 7%'.

What do you think, does this data provide evidence that the coin is biased rather than fair?

EXERCISE 112. You visit a family with three children, but you don't know their sexes. Each child has his/her own bedroom, and you stumble by chance into one of the bedrooms. It is clear that it is the bedroom of a girl. Then you observe a letter from the school addressed to parents of the boys in the school.

Now you know that the family has at least one boy and one girl. What is more likely, that the family has one boy and two girls, or one girl and two boys? State your answer in terms of probabilities.

EXERCISE 113. The Monte Hall problem. (For this problem it is a good idea to study Example 107 first.)

The rules of a game show are as follows. The contestant is shown three door, behind one door is a prize, behind the other two doors, nothing. The contestant chooses a door and indicate it to the game show host. At lease one of the two remaining doors, is empty. The host, who knows where the prize is, chooses an

empty door between the other two, and open it. The contestant is now faced with two closed doors, his original choice and one other, and one open, empty door. The contestant now gets the opportunity of changing his/her original choice.

Calculate the probabilities of winning the prize when (a) the contestant changes his/her original choice, (b) sticks with his/her original choice.

If you know this problem, chances are that you know an easy argument to find the probabilities. If you don't know the argument, try and find it. What I want you to do, is to argue systematically, based on conditional probabilities. In what way does the opening of an empty door, adds information to the system that you can exploit?

EXERCISE 114. This problem is exactly like the previous one. Except that after the contestant made his/her initial choice, there is a small earthquake, causing one of the remaining empty doors to fly open. Assuming that earthquakes know nothing about the game, how does this change the situation? What is the probabilities of winning the prize, using both strategies?

EXERCISE 115. You play a game with your friend consisting of three identical cards, except that both sides of one card are painted black, both sides of the second card are painted red, and the two sides of the third card are painted red and black respectively. Your friend picks a card at random, then show you, also randomly, one of the sides. You therefore see either a red or black side. The game consists of you guessing the color of the other, unseen, side of the card. What is your best strategy to guess right as often as possible? How often do you expect to guess right?

You can again approach the problem by listing all possibilities. Alternatively, you can use Bayes' theorem. Try comparing three hypotheses, you see the black-black card, you see the red-red card, you see the red-black card. If C is the random variable describing your observation, i.e. $C \in \{\text{red},\ \text{black}\}$, you need to calculate the posterior probabilities, $P(\mathcal{H}|C)$. Answer: $P(\mathcal{H} = \text{red-red}|C = \text{red}) = \frac{2}{3}$, $P(\mathcal{H} = \text{red-black}|C = \text{red}) = \frac{1}{3}$.

Now that we have seen the examples, and you have worked through a few exercises, let us look at the general situation. Suppose have are observing a generative process providing us with observations $\mathcal{D}$, assumed to be statistically independent. We now *model* the process, resulting in a generative model. That is we hypothesize that the observed data is generated by a pdf $p(\mathbf{x}|\theta_a, \mathcal{H}_a)$, where we now explicitly indicate that the model is conditioned on our hypothesis $\mathcal{H}_a$, and the model depends on parameters θ_a that can be adjusted to the data. Given the observations, we form the likelihood $p(\mathcal{D}|\theta_a, \mathcal{H}_a)$. Because the observations are statistically independent, this joint distribution factorizes over the individual observations.

The posterior distribution over the parameters requires a prior $p(\theta_a|\mathcal{H}_a)$ over the parameters, and is given by

$$p(\theta_a|\mathcal{D}, \mathcal{H}_a) = \frac{p(\mathcal{D}|\theta_a, \mathcal{H}_a)p(\theta_a|\mathcal{H}_a)}{p(\mathcal{D}_a|\mathcal{H}_a)}, \tag{22.2}$$

where we recall that the denominator is also referred to as the *evidence.* Given the posterior of the parameters, we make predictions. For example, the probability of making a specific observation $\mathbf{x}$, is given by

$$p(\mathbf{x}|\mathcal{D}, \mathcal{H}_a) = \int p(\mathbf{x}|\theta, \mathcal{H}_a)p(\theta|\mathcal{D}, \mathcal{H}_a)d\theta,$$

where we have made use of the conditional independence of $\mathbf{x}$ on the data $\mathcal{D}$, given the parameters θ.

We can also ask how well the data supports the hypothesis. Introducing the prior $P(\mathcal{H}_a)$, it follows that

$$P(\mathcal{H}_a|\mathcal{D}) = \frac{p(\mathcal{D}|\mathcal{H}_a)P(\mathcal{H}_a)}{p(\mathcal{D})},$$

where $p(\mathcal{D}|\mathcal{H}_a)$ is given by the *evidence* in (22.2). The normalization term $p(\mathcal{D})$ poses a problem. In order to do the normalization we need an exhaustive set of hypotheses, which may not be available. In practice however, a *comparison* of different hypotheses is often all that is required. Accordingly, suppose we have an alternative hypothesis $\mathcal{H}_b$ with model parameters θ_b. Following the same reasoning as above, we find that

$$\frac{P(\mathcal{H}_a|\mathcal{D})}{P(\mathcal{H}_b|\mathcal{D})} = \frac{p(\mathcal{D}|\mathcal{H}_a)P(\mathcal{H}_a)}{p(\mathcal{D}|\mathcal{H}_b)P(\mathcal{H}_b)}.$$

Taking the log, gives

$$\log \frac{P(\mathcal{H}_a|\mathcal{D})}{P(\mathcal{H}_b|\mathcal{D})} = \log \frac{p(\mathcal{D}|\mathcal{H}_a)}{p(\mathcal{D}|\mathcal{H}_b)} + \log \frac{P(\mathcal{H}_a)}{P(\mathcal{H}_b)}.$$

This nicely separates the contributions of two quantities. The *odds* in favor of hypothesis $\mathcal{H}_a$ against hypothesis $\mathcal{H}_b$ is just the ratio $\frac{P(\mathcal{H}_a)}{P(\mathcal{H}_b)}$. Thus, the posterior log-odds equals the prior log-odds plus the log-likelihood ratio.

If you are wondering about the meaning of the term 'odds', it derives from betting where the odds in favor of an event E is given by

$$\text{odds}(E) = \frac{P(E)}{1 - P(E)}.$$

For example, if $P(E) = 0.2$, then the odds in favor of event E, is given by $\text{odds}(E) = 0.25$. In betting the bookmakers give their odds *against* the event taking place. Thus, if $P(E) = 0.2$, the odds against it taking place is $1/\text{odds}(E) = 4$. This means that betting R1, you win R4 if the event realizes, against the bookie who wins your R1 if the event does not take place.

22.5. Gaussian Distribution.

22.5.1. Univariate Gaussian Distributions. One of the most important distributions is the Gaussian– or normal distribution

$$N(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\}. \tag{22.1}$$

The Gaussian distribution is governed by exactly two parameters, the mean μ and the variance σ^2. The square root of the variance σ is called the standard deviation and the reciprocal of the variance $1/\sigma^2$ is called the precision—the smaller the variance, the greater the precision, see Figure 22.5.1.

Note that

$$\mathbb{E}[x] = \int_{-\infty}^{\infty} N(x|\mu, \sigma^2) x dx = \mu$$

and

$$\mathbb{E}[x^2] = \int_{-\infty}^{\infty} N(x|\mu, \sigma^2) x^2 dx = \mu^2 + \sigma^2.$$

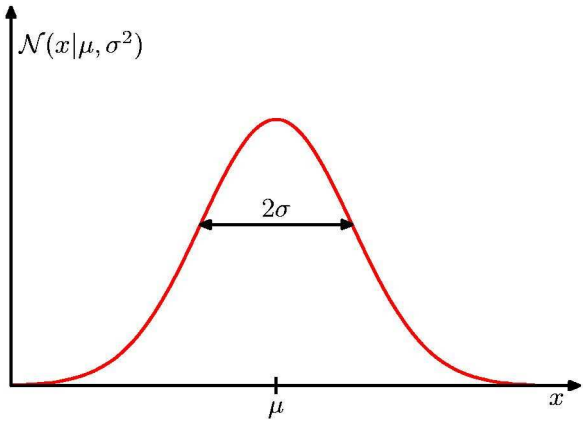


FIGURE 22.5.1. Gaussian density function.

Thus the variance is given by

$$\text{var}[x] = \mathbb{E}[x^2] - \mathbb{E}[x]^2 = \sigma^2.$$

Suppose we have N independent observations $\mathbf{x} = [x_1 \ \cdots \ x_N]^T$ all drawn from the same Gaussian distribution $N(x|\mu,\sigma^2)$. Such data points are said to be *independent* and i*dentically* distributed, or i.i.d. Furthermore, suppose that we do not know the values of μ and σ^2 and want to infer them from the observations. Because the data set $\mathbf{x}$ is i.i.d. we can factorize the joint distribution as

$$p(\mathbf{x}|\mu,\sigma^2) = \prod_{n=1}^{N} N(x_n|\mu,\sigma^2).$$

Viewed as a function of μ and σ^2 this is the likelihood function for the Gaussian parameters, see Figure 22.5.2.

In order to infer the values of μ and σ^2 from the data, one might maximize the likelihood function, or rather, the log likelihood,

$$\ln p(x|\mu,\sigma^2) = -\frac{1}{2\sigma^2}\sum_{n=1}^{N}(x_n-\mu)^2 - \frac{N}{2}\ln\sigma^2 - \frac{N}{2}\ln(2\pi).$$

Maximizing with respect to μ gives the *maximum likelihood* solution,

$$\mu_{ML} = \frac{1}{N}\sum_{n=1}^{N} x_n,$$

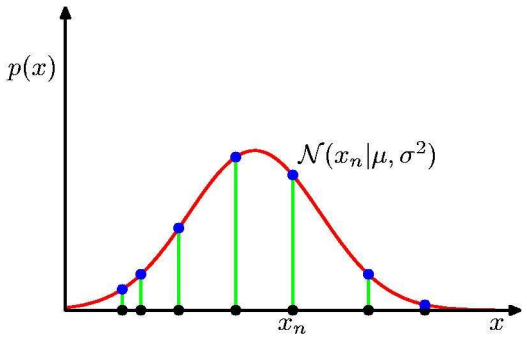


FIGURE 22.5.2. Likelihood function.

which is the *sample* mean. The maximum likelihood solution of the variance is similarly obtained as

$$\sigma^2_{ML} = \frac{1}{N}\sum_{n=1}^{N}(x_n - \mu_{ML})^2.$$

EXERCISE 116. The sample mean and sample variance are also random variables and one can calculate their mean. Show that

- $\mathbb{E}[\mu_{ML}] = \mu$
- $\mathbb{E}[(\mu_{ML} - \mu)^2] = \frac{\sigma^2}{N}$. Hint: It is the simplest to write $\delta_n = \mu - x_n$, so that $\mathbb{E}[\delta] = 0$ and $\mathbb{E}[\delta^2] = \sigma^2$.
- $\mathbb{E}[\sigma^2_{ML}] = \frac{N-1}{N}\sigma^2$. Hint: Write

$$\begin{aligned}\sum_{n=1}^{N}(x_n - \mu_{ML})^2 &= \sum_{n=1}^{N}\left[(x_n - \mu) - (\mu_{ML} - \mu)\right]^2 \\ &= \sum_{n=1}^{N}(x_n - \mu)^2 - 2(\mu_{ML} - \mu)\sum_{n=1}^{N}(x_n - \mu) + N(\mu_{ML} - \mu)^2 \\ &= \sum_{n=1}^{N}(x_n - \mu)^2 - N(\mu_{ML} - \mu)^2.\end{aligned}$$

From Exercise 116 it is clear that a bias is introduced into the sample variance. This is due to the fact that the variance is calculated with respect to the sample mean and not the real mean. An unbiased estimate is easily obtained (show it yourself!)

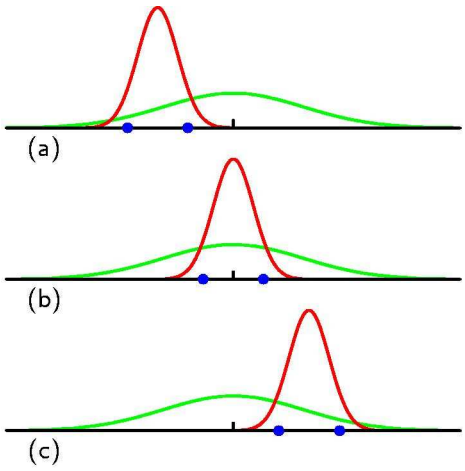


FIGURE 22.5.3. Illustration of how a bias arises. Averaged over three data sets the mean is correct but the variance is under-estimated.

by

$$\sigma^2_{UB} = \frac{1}{N-1}\sum_{n=1}^{N}(x_n - \mu_{ML})^2.$$

The bias can also be explained in the following way. First note that the data points are distributed around the true mean μ with mean squared error σ^2. The sample mean μ_{ML} minimizes the mean squared error of the data. Thus, unless μ and μ_{ML} coincide, the data has a larger sum-squared deviation from the the true mean μ than from the sample mean μ_{ML}. The expected mean squared deviation from the sample mean is therefore smaller than the mean squared deviation from the true mean. This is illustrated in Figure 22.5.3.

EXERCISE 117. Assume that you are observing a generative process that generates observations from a univariate Gaussian distribution, $\mathcal{N}(x|0, \sigma^2 = 1)$. You are asked to write a computer program to generate samples from this distribution. You may assume that you have access to an algorithm that draws samples from a uniform distribution. These are readily available and you may use whatever is available in your favorite software library. Explain, based on the discussion of Section˜30.3 how you can use samples drawn from a uniform distribution to generate samples from a Gaussian distribution. Use this to write a program that will generate an arbitrary number of data points from the Gaussian distribution. Plot the data together with

the Gaussian distribution. How can you easily adapt your code in order to generate data from a Gaussian distribution with arbitrary mean and covariance?

22.5.2. Multivariate Gaussian Distributions. In D dimensions the Gaussian distribution is given by,

$$N(\mathbf{x}|\boldsymbol{\mu},\Sigma) = \frac{1}{|2\pi\Sigma|^{\frac{1}{2}}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\}, \tag{22.2}$$

where $|\cdot|$ denotes the determinant, and the covariance matrix Σ is an $D\times D$ symmetric matrix.

EXERCISE 118. Show that there is no loss in generality to assume that the covariance matrix is symmetric. Hint: Assume that Σ^{-1} is not symmetric and write it as $\Sigma^{-1} = \Sigma_s^{-1} + \Sigma_{as}^{-1}$, i.e. write it as the sum of a symmetric and an anti-symmetric matrix. Now show that the anti-symmetric part cancels and draw the appropriate conclusion.

The Mahalanobis distance is defined as,

$$\Delta^2 = (\mathbf{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}). \tag{22.3}$$

Note:

- $\Delta = \text{const}$ defines the curves of constant probability.

If we write $U = [\mathbf{u}_1 \ \cdots \ \mathbf{u}_D]^T$ and $\Lambda = \text{diag}(\lambda_1,\ldots,\lambda_D)$ where the $\mathbf{u}_j$ are the eigenvectors, and the λ_j the corresponding eigenvalues of the covariance matrix, i.e.,

$$\Sigma\mathbf{u}_j = \lambda_j\mathbf{u}_j,\ j = 1,\ldots,D,$$

then we can write

$$\left[\begin{array}{ccc} \Sigma\mathbf{u}_1 & \ldots & \Sigma\mathbf{u}_D \end{array}\right] = \left[\begin{array}{ccc} \lambda_1\mathbf{u}_1 & \ldots & \lambda_D\mathbf{u}_D \end{array}\right]$$

or

$$\Sigma U = U\Lambda.$$

This means that

$$\Sigma = U\Lambda U^T$$

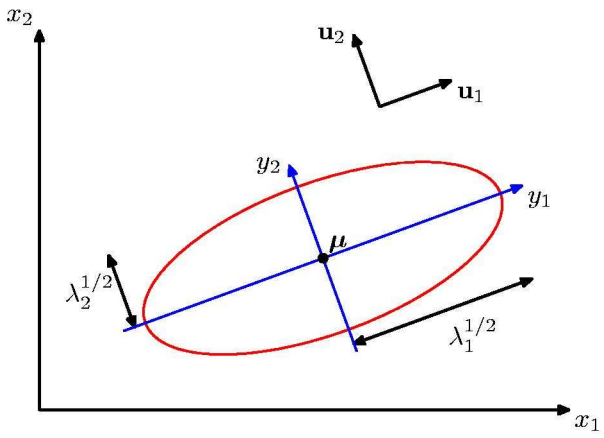


FIGURE 22.5.4. Rotating the principle axes.

or

$$\Sigma^{-1} = U\Lambda^{-1}U^T$$

assuming, of course, that the eigenvectors are normalized. Multiplying out this gives,

$$\Sigma = \sum_{i=1}^{D} \lambda_i \mathbf{u}_i \mathbf{u}_i^T,$$

and

$$\Sigma^{-1} = \sum_{i=1}^{D} \frac{1}{\lambda_i} \mathbf{u}_i \mathbf{u}_i^T.$$

If we now change variables

$$\mathbf{y} = U^T(\mathbf{x} - \boldsymbol{\mu}) \tag{22.4}$$

in (22.3) we get

$$\begin{aligned} \Delta^2 &= \mathbf{y}^T U^T \Sigma^{-1} U \mathbf{y} \\ &= \mathbf{y}^T \Lambda^{-1} \mathbf{y} \\ &= \sum_{i=1}^{D} \frac{y_i^2}{\lambda_i} \end{aligned}$$

The transformation (22.4) is simply a rotation so that the principle axes coincide with the coordinate axes, see Figure 22.5.4.

In these coordinates the Gaussian becomes

$$p(\mathbf{y}) = N(\mathbf{y}|\mathbf{0}, \Lambda).$$

This is magical: The $\mathbf{y}$ coordinates, obtained by a rotation of the $\mathbf{x}$ coordinates, are statistically independent since the covariance matrix is diagonal. Thus for normally distributed variables one easily removes any statistical dependence through a simple rotation around the mean.

Note: For the normal distribution (22.2) the mean and covariance are given by

- $\mathbb{E}[\mathbf{x}] = \boldsymbol{\mu}$.
- $\text{cov}[\mathbf{x}] = \mathbb{E}[(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^T] = \Sigma$.

The following exercise prepares the way for what is to follow.

EXERCISE 119. Jointly Gaussian. Assume that two Gaussian variables x and y are jointly Gaussian in the sense that

$$p(x, y) = N(\mathbf{x}|\boldsymbol{\mu}, \Sigma),$$

where $\mathbf{x} = [x\ y]^T$ and $\Sigma = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{bmatrix}$.

- Show that the marginal distributions $p(x)$ and $p(y)$ are Gaussian and calculate their mean and covariance.
- Show that $p(x|y)$ is Gaussian and calculate its mean and covariance.
- Now show that x and y are independent if and only if $\sigma_{xy} = 0$.

Although the next section might seem a little technical, the results discussed below is one of the reasons why we use Gaussian distributions extensively.

Conditional Gaussian distribution. The following identity will be used extensively,

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1} = \begin{bmatrix} M & -MBD^{-1} \\ -D^{-1}CM & D^{-1} + D^{-1}CMBD^{-1} \end{bmatrix} \tag{22.5}$$

where

$$M = \left(A - BD^{-1}C\right)^{-1}.$$

EXERCISE 120. Verify the identity by showing that

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \times \begin{bmatrix} M & -MBD^{-1} \\ -D^{-1}CM & D^{-1} + D^{-1}CMBD^{-1} \end{bmatrix} = I.$$

Suppose that $\mathbf{x} \in \mathbb{R}^D$ is a Gaussian random variable $\mathbf{x} \sim N(\mathbf{x}|\boldsymbol{\mu}, \Sigma)$. Partition

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_a \\ \mathbf{x}_b \end{bmatrix}$$

where $\mathbf{x}_a$ is the first M components of $\mathbf{x}$. We also partition the mean and covariance matrix,

$$\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_a \\ \boldsymbol{\mu}_b \end{bmatrix},$$

and

$$\Sigma = \begin{bmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{bmatrix}$$

where $\Sigma_{aa} = \Sigma_{aa}^T,\ \Sigma_{bb} = \Sigma_{bb}^T,\ \Sigma_{ab} = \Sigma_{ba}^T$. Also partition the precision matrix,

$$\Lambda = \Sigma^{-1},$$

as

$$\Lambda = \begin{bmatrix} \Lambda_{aa} & \Lambda_{ab} \\ \Lambda_{ba} & \Lambda_{bb} \end{bmatrix}.$$

We are looking for $p(\mathbf{x}_a|\mathbf{x}_b)$. It turns out that it is again a Gaussian distribution in $\mathbf{x}_a$. Therefore we only need to find the mean and covariance of $p(\mathbf{x}_a|\mathbf{x}_b)$. In principle this is easy to calculate. All we need to do is to fix $\mathbf{x}_b$ in the joint distribution, and to extract the quadratic form of the remaining $\mathbf{x}_a$. A tedious but essentially straightforward calculation gives,

$$\Sigma_{a|b} = \Lambda_{aa}^{-1},$$

and

$$\boldsymbol{\mu}_{a|b} = \boldsymbol{\mu}_a - \Lambda_{aa}^{-1}\Lambda_{ab}(\mathbf{x}_b - \boldsymbol{\mu}_b).$$

Making use of the matrix identity this can also be expressed in terms of the partitioned covariance matrix

$$\Sigma_{a|b} = \Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba} \tag{22.6}$$

and

$$\boldsymbol{\mu}_{a|b} = \boldsymbol{\mu}_a + \Sigma_{ab}\Sigma_{bb}^{-1}(\mathbf{x}_b - \boldsymbol{\mu}_b). \tag{22.7}$$

Note that the conditional mean takes the form

$$\boldsymbol{\mu}_{a|b} = A\mathbf{x}_b + \mathbf{b}$$

and that the conditional covariance is independent of $\mathbf{x}_a$. This is an example of a linear Gaussian model.

22.5.3. Marginal Gaussian distribution. We want to compute the marginal distribution

$$p(\mathbf{x}_a) = \int p(\mathbf{x}_a, \mathbf{x}_b) d\mathbf{x}_b.$$

Again it turns out to be a Gaussian distribution,

$$p(\mathbf{x}_a) = N(\mathbf{x}_a|\boldsymbol{\mu}_a, \Sigma_{aa}).$$

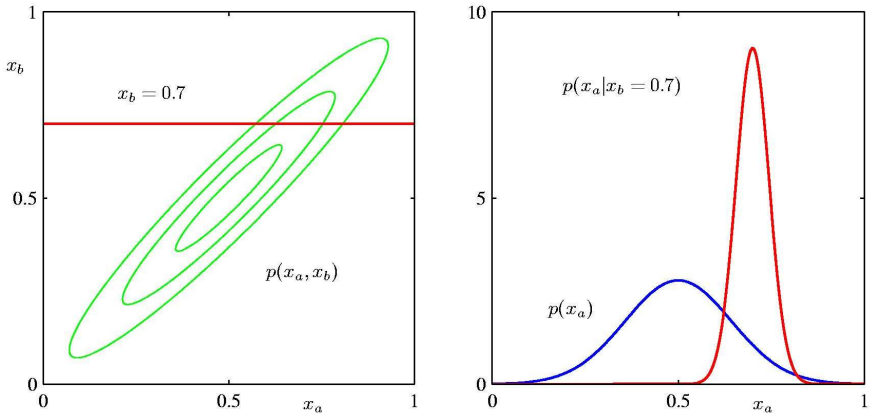
This means that the marginal distribution can be simply read off the partitioned mean and covariance. This illustrated in Figure 22.5.5

22.5.4. Bayes' theorem for Gaussian variables. Assume we are given a Gaussian marginal distribution $p(\mathbf{x})$ and a Gaussian conditional distribution $p(\mathbf{y}|\mathbf{x})$ that has a mean that is linear in $\mathbf{x}$ and a covariance independent of $\mathbf{x}$. We wish to find the marginal distribution $p(\mathbf{y})$ and the conditional distribution $p(\mathbf{x}|\mathbf{y})$. Let

$$\begin{aligned} p(\mathbf{x}) &= N(\mathbf{x}|\boldsymbol{\mu}, \Lambda^{-1}) \\ p(\mathbf{y}|\mathbf{x}) &= N(\mathbf{y}|A\mathbf{x} + \mathbf{b}, L^{-1}). \end{aligned}$$

If $\mathbf{x} \in \mathbb{R}^M$ and $\mathbf{y} \in \mathbb{R}^{\mathbb{D}}$ then A is an $D \times M$ matrix. Note that we are essentially given the joint distribution

$$p(\mathbf{x}, \mathbf{y}) = p(\mathbf{y}|\mathbf{x})p(\mathbf{x}).$$



(a) (b)

FIGURE 22.5.5. (a) $2D$ Gaussian distribution. (b) Marginal and conditional distributions.

Since it is the product of Gaussian distributions, it is also a Gaussian distribution. Therefore the marginal $p(\mathbf{y})$ can be obtained through marginalization. Again an essentially straightforward but tedious calculation gives the mean and covariance of the marginal,

$$\begin{aligned} \mathbb{E}[\mathbf{y}] &= A\boldsymbol{\mu} + \mathbf{b} \\ \mathrm{cov}[\mathbf{y}] &= L^{-1} + A\Lambda^{-1}A^T. \end{aligned}$$

The posterior distribution is now obtained using Bayes' theorem,

$$p(\mathbf{x}|\mathbf{y}) = \frac{p(\mathbf{y}|\mathbf{x})p(\mathbf{x})}{p(\mathbf{y})}.$$

Not surprisingly this is a Gaussian distribution with mean and covariance given by,

$$\begin{aligned} \mathbb{E}[\mathbf{x}|\mathbf{y}] &= \left(\Lambda + A^T L A\right)^{-1} \left(A^T L(\mathbf{y} - \mathbf{b}) + \Lambda\boldsymbol{\mu}\right) \\ \mathrm{cov}[\mathbf{x}|\mathbf{y}] &= \left(\Lambda + A^T L A\right)^{-1}. \end{aligned}$$

Summary:

Given a marginal distribution $p(\mathbf{x})$ and a conditional distribution $p(\mathbf{y}|\mathbf{x})$ in the form

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \Lambda^{-1}) \tag{22.8}$$

$$p(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{y}|A\mathbf{x} + \mathbf{b}, L^{-1}) \tag{22.9}$$

the marginal $p(\mathbf{y})$ and conditional $p(\mathbf{x}|\mathbf{y})$ distributions are given by

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y}|A\boldsymbol{\mu} + \mathbf{b}, L^{-1} + A\Lambda^{-1}A^T) \tag{22.10}$$

$$p(\mathbf{x}|\mathbf{y}) = \mathcal{N}(\mathbf{x}|\Sigma\left\{A^T L(\mathbf{y} - \mathbf{b}) + \Lambda\boldsymbol{\mu}\right\}, \Sigma) \tag{22.11}$$

where

$$\Sigma = (\Lambda + A^T L A)^{-1}.$$

22.5.5. Maximum likelihood. Given an i.i.d. data set, $X = [\mathbf{x}_1 \dots \mathbf{x}_N]$ drawn from a multivariate Gaussian distribution, the log likelihood is given by

$$\ln p(X|\boldsymbol{\mu}, \Sigma) = -\frac{ND}{2}\ln(2\pi) - \frac{N}{2}\ln|\Sigma| - \frac{1}{2}\sum_{n=1}^{N}(\mathbf{x}_n - \boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}_n - \boldsymbol{\mu}).$$

The sufficient statistics are given by $\sum_{n=1}^{N}\mathbf{x}_n$ and $\sum_{n=1}^{N}\mathbf{x}_n\mathbf{x}_n^T$. Maximizing gives,

$$\boldsymbol{\mu}_{ML} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n$$

and

$$\Sigma_{ML} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \boldsymbol{\mu}_{ML})(\mathbf{x}_n - \boldsymbol{\mu}_{ML})^T.$$

The unbiased estimate is given by

$$\Sigma_{ML} = \frac{1}{N-1}\sum_{n=1}^{N}(\mathbf{x}_n - \boldsymbol{\mu}_{ML})(\mathbf{x}_n - \boldsymbol{\mu}_{ML})^T.$$

22.5.6. Bayesian inference for the univariate Gaussian. Suppose that the variance σ^2 is known. We need to infer the mean μ given data $\mathcal{D} = \{x_1, \dots, x_N\}$. The likelihood is given by

$$p(\mathcal{D}|\mu) = \prod_{n=1}^{N} p(x_n|\mu) = \frac{1}{(2\pi\sigma^2)^{N/2}}\exp\left\{-\frac{1}{2\sigma^2}\sum_{n=1}^{N}(x_n - \mu)^2\right\}.$$

Maximizing the likelihood gives a point estimate of μ. A systematic Bayesian approach treats μ like a random variable and looks for a pdf over μ, given the data. This means that we we are after the posterior

$$p(\mu|\mathcal{D}) = \frac{p(\mathcal{D}|\mu)p(\mu)}{p(\mathcal{D})}.$$

Here one has to provide a suitable prior $p(\mu)$. Note that the likelihood as a function of μ, assumes the form of a Gaussian distribution albeit not properly normalized. This however, allows us to choose a conjugate prior $p(\mu)$ in the form of a Gaussian,

$$p(\mu) = \mathcal{N}(\mu|\mu_0, \sigma_0^2)$$

so that the posterior distribution becomes

$$p(\mu|\mathcal{D}) \propto p(\mathcal{D}|\mu)p(\mu).$$

Completing the square it follows that

$$p(\mu|\mathcal{D}) = \mathcal{N}(\mu|\mu_N, \sigma_N^2)$$

where

$$\begin{aligned} \mu_N &= \frac{\sigma^2}{N\sigma_0^2 + \sigma^2}\mu_0 + \frac{N\sigma_0^2}{N\sigma_0^2 + \sigma^2}\mu_{ML} \\ \frac{1}{\sigma_N^2} &= \frac{1}{\sigma_0^2} + \frac{N}{\sigma^2} \end{aligned}$$

where μ_{ML} is the sample mean.

Note that the prior plays the role of a fictitious observation followed by N 'real' observations. The observations serve to 'correct' or modify the prior assumption, see Figure 22.5.6. Also note that one can manipulate the relative importance of the prior by adjusting the prior variance σ_0^2. A small value of σ_0^2 indicates high confidence in the prior. In fact $\sigma_0^2 = 0$ shows absolute confidence with the result that the measurements have no effect. Small values means that its effect is persistent and only disappears as the number of observations $N \longrightarrow \infty$.

Note that this leads naturally to a sequential view where is observation is used to correct our current best estimate.

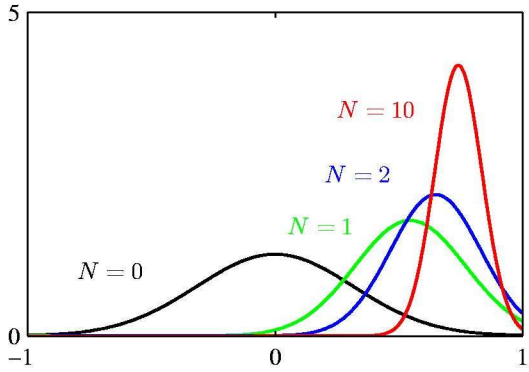


FIGURE 22.5.6. Bayesian inference for the mean μ.

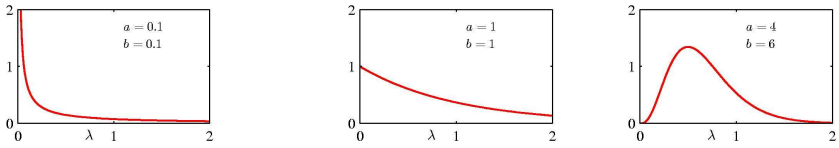


FIGURE 22.5.7. The gamma distribution.

Now suppose that the mean is known and we want to infer the variance, or more conveniently the precision $\lambda = \frac{1}{\sigma^2}$. The likelihood is given by

$$p(\mathbf{x}|\lambda) = \prod_{n=1}^{N} \mathcal{N}(x_n|\mu, \lambda^{-1}) \propto \lambda^{N/2} \exp\left\{-\frac{\lambda}{2}\sum_{n=1}^{N}(x_n-\mu)^2\right\}.$$

Following the same reasoning as above, the conjugate prior is a gamma distribution

$$\text{Gam}(\lambda|a,b) = \frac{1}{\Gamma(a)} b^a \lambda^{a-1} \exp(-b\lambda),$$

with mean and variance given by,

$$\begin{aligned} \mathbb{E}[\lambda] &= \frac{a}{b} \\ \text{var}[\lambda] &= \frac{a}{b^2}. \end{aligned}$$

The gamma distribution is illustrated in Figure 22.5.7

Using the prior $p(\lambda) = \text{Gam}(\lambda|a_0, b_0)$ the posterior becomes

$$p(\lambda|\mathcal{D}) \propto \lambda^{a_0-1}\lambda^{N/2} \exp\left\{-b_0\lambda - \frac{\lambda}{2}\sum_{n=1}^{N}(x_n-\mu)^2\right\}$$

which is a Gamma distribution of the form $\text{Gam}(\lambda|a_N, b_N)$ where

$$\begin{aligned} a_N &= a_0 + \frac{N}{2} \\ b_N &= b_0 + \frac{1}{2}\sum_{n=1}^{N}(x_n-\mu)^2 = b_0 + \frac{N}{2}\sigma^2_{ML}. \end{aligned}$$

Note that N observations increase a_0 by $\frac{N}{2}$. One therefore interprets a_0 as being the equivalent of $2a_0$ prior observations. The N observations also increase the value of b_0 by $N\sigma^2_{ML}/2$. If we want to interpret b_0 as being the result of the $2a_0$ prior observations, we have to write $b_0 = a_0\frac{b_0}{a_0}$ so that we conclude that the prior is the equivalent of $2a_0$observations with variance $\frac{b_0}{a_0}$.

If both the mean and precision are unknown, we are looking for an expression for the posterior $p(\mu, \lambda|\mathcal{D}) \propto p(\mathcal{D}|\mu, \lambda)p(\mu, \lambda)$. It is worth writing the likelihood again,

$$\begin{aligned} p(\mathcal{D}|\mu,\lambda) &= \prod_{n=1}^{N}\left(\frac{\lambda}{2\pi}\right)^{\frac{1}{2}} \exp\left\{-\frac{\lambda}{2}(x_n-\mu)^2\right\} \\ &\propto \left[\lambda^{\frac{1}{2}}\exp\left(-\frac{\lambda\mu^2}{2}\right)\right]^N \exp\left\{\lambda\mu\sum_{n=1}^{N}x_n - \frac{\lambda}{2}\sum_{n=1}^{N}x_n^2\right\}. \end{aligned}$$

A conjugate prior takes the same functional form as the likelihood and should therefore be of the form

$$\begin{aligned} p(\mu,\lambda) &\propto \left[\lambda^{\frac{1}{2}}\exp\left(-\frac{\lambda\mu^2}{2}\right)\right]^\beta \exp\{c\lambda\mu - d\lambda\} \\ &= \exp\left(-\frac{\beta\lambda\mu^2}{2} + c\lambda\mu\right)\left[\lambda^{\frac{\beta}{2}}\exp(-d\lambda)\right] \\ &= \exp\left\{-\frac{\beta\lambda}{2}\left(\mu - \frac{c}{\beta}\right)^2\right\}\left[\lambda^{\frac{\beta}{2}}\exp\left\{-\left(d - \frac{c^2}{2\beta}\right)\lambda\right\}\right]. \end{aligned}$$

Factorized the prior as $p(\mu, \lambda) = p(\mu|\lambda)p(\lambda)$, and compare with the expression above to find that

$$\begin{aligned} p(\mu, \lambda) &= p(\mu|\lambda)p(\lambda) \\ &= \mathcal{N}\left(\mu|\mu_0, (\beta\lambda)^{-1}\right) \mathrm{Gam}(\lambda|a, b) \end{aligned}$$

with

$$\mu_0 = \frac{c}{\beta}, \quad a = 1 + \frac{\beta}{2}, \quad b = d - \frac{c^2}{2\beta}.$$

This means that the posterior has a normal-gamma distribution.

EXERCISE 121. In Exercise 117 you developed a program that samples from a univariate Gaussian pdf. Generalize your program to generate samples from a multivariate Gaussian distribution. Plot your results for a 2D Gaussian distribution. Plot the data points together with the contours of the pdf (this might require some thought). Choose covariance matrices so that the two components are (i) independent, (ii) dependent. Also use a covariance matrix that is a multiple of the identity matrix. Note how the shape of the contours change. And of course your data should be consistent with these curves.

22.6. Linear Transformations of Gaussians and the central limit theorem.

There is another important and useful property of Gaussian pdf's namely that a linear transformation results in another Gaussian. Let

$$\mathbf{x} = A\mathbf{y}$$

where A is an invertible matrix. Substituting into (22.2) gives,

$$\begin{aligned} \mathcal{N}(\mathbf{y}|\boldsymbol{\mu}_y, \Sigma_y) &\propto \exp(-\frac{1}{2}(A\mathbf{y} - \boldsymbol{\mu})^T\Sigma^{-1}(A\mathbf{y} - \boldsymbol{\mu})) \\ &= \exp(-\frac{1}{2}(\mathbf{y} - A^{-1}\boldsymbol{\mu})^T A^T\Sigma^{-1}A(\mathbf{y} - A^{-1}\boldsymbol{\mu})). \end{aligned}$$

It now easily follows that $\boldsymbol{\mu}_y = A^{-1}\boldsymbol{\mu}$ and $\Sigma_y = A^{-1}\Sigma A^{-T}$. Although it is a linear transformation, it is a very special kind of linear transformation. Consider for instance the following transformation

$$z = x_1 + x_2$$

where both x_1 and x_2 are Gaussian random variables, let us say with means μ_1 and μ_2 and variances σ_1^2 and σ_2^2. This is also a linear transformation but in this case the transformation matrix is not invertible. The previous derivation does not cover this result, and the question remains, is a linear transformation of a Gaussian again a Gaussian? The next exercise guides you through a simple construction.

EXERCISE 122. Let $z_1 = x_1 + x_2$ and define $z_2 = x_2$. Write down the linear transformation $\mathbf{z} = A\mathbf{x}$ where $\mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$ and $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$. Show that A is invertible. Assume that x_1 and x_2 are jointly Gaussian and write down the joint distribution for $\mathbf{x}$. From this you should be able to calculate the Gaussian for $\mathbf{z}$ from which you can obtain the distribution of z_1 by marginalization. Consider the separate cases when x_1 and x_2 are statistically independent, and when they are dependent.

What about random variables that are not Gaussian? The next exercise guides you through an example.

EXERCISE 123. Let $z = x_1 + x_2$ and assume that x_1 and x_2 are statistically independent random variables with, in this case, the pdf's given by the general functions $p_1(x_1)$ and $p_2(x_2)$ respectively. Try the same trick as in the previous exercise, i.e. write $z_1 = x_1 + x_2$ and $z_2 = x_2$. If $p(x_1, x_2)$ is the joint pdf we can now transform to the z-coordinates, $p(z_1, z_2) = p(x_1(z_1, z_2), x_2(z_1, z_2)) = p_1(x_1(z_1, z_2))p_2(x_2(z_1, z_2)) = p_1(z_1 - z_2)p_2(z_2)$ (noting that the determinant of the transformation matrix equals 1). If we do the transformation, and marginalize, $p(z_1) = \int p(z_1, z_2)dz_2$, we can write the pdf of z in the form of a convolution.

EXERCISE 124. Given two statistically independent random variables x and y with means $\overline{x}$ and $\overline{y}$ respectively, calculate the mean of $z = x + y$. Show that the variance of z is the sum of the variances of x and y.

EXERCISE 125. This exercise, together with the next one should help you understand the idea behind the central limit theorem.

Assume that you throw two fair six-sided dice, i.e. assume that the six sides are drawn from a uniform distribution every time you throw. Write down the probability

distribution of the sum of the two dice. How does this relate to Exercise 123? What is the probability distribution of the sum of the numbers if you throw three dice?

Suppose that you throw 100 dice, what is the probability distribution of the sum of the numbers? Write a computer program that simulates this. Also describe how one can calculate the distribution theoretically, at least in principle (you may find it tedious to do it in practice).

EXERCISE 126. The central limit theorem states that if independent random variables $x_1, x_2, \ldots, x_N$ have means μ_n and variances σ_n^2, then, in the limit of large N, the sum $\sum_n x_n$ has a distribution that tends to a Gaussian distribution with mean $\sum_n \mu_n$ and variance $\sum_n \sigma_n^2$.

Calculate the mean and variance of one die (assume a uniform distribution). Use the results of the previous problems and calculate the mean and variance of the sum one hundred dice. In the previous question you simulated the probability distribution of the sum of 100 dice. Compare your simulation with a Gaussian distribution with mean and variance you have just calculated. Do you find confirmation for the central limit theorem in the result?

Since you have shown that the probability distribution of the sum of two independent variables is given by a convolution, another way of approaching this problem is to calculate the convolution of two uniform distributions (giving the distribution of two dice). Show that by taking convolutions of the result with a uniform distribution recursively, generates the probability distributions of more and more dice (you need to ensure that the probability distributions are properly normalized of course).

CHAPTER 23

LINEAR MODELS FOR REGRESSION

23.1. Introduction.

The regression problem is related to the interpolation problem we studied before in Chapter 8. The main difference is that we no longer assume that the data is known with infinite precision. This implies that it is no longer feasible to require the approximant to pass through the data—the data itself is imprecisely known. In addition, we want to make use of all available knowledge in a principled way in order to obtain the best possible estimates.

From a modeling point of view, whenever your model requires the output in terms of a real number, you are probably dealing with a regression problem. The modeling process should become more familiar by now. Using a parametric generative model, training data is used to estimate the model parameters. Once the parameters are known one can query the model, given input, the model provides a response in terms of a continuous variable. We first start by finding point estimates for the parameters, moving to a more systematic Bayesian approach where the model parameters are treated as random variables.

23.2. Curve Fitting.

Suppose we are given the data points, $\mathbf{x} = \{x_1, \ldots, x_N\}$, together with corresponding observations $\mathbf{t} = \{t_1, \ldots, t_N\}$. Think of the x_n as different machine settings and the t_n as observations corresponding to the setting. Alternatively, one can think of $\mathbf{x}, \mathbf{t}$ as a training set with $\mathbf{x}$ the observations and $\mathbf{t}$ the corresponding known prediction values. The idea is to be able to predict a value for $\widehat{t}$, given a corresponding, previously unseen, value for $\widehat{x}$. One way of approaching this problem is to view it as an interpolation problem, i.e. we fit a polynomial $t = P_N(x)$ (here N refers to the

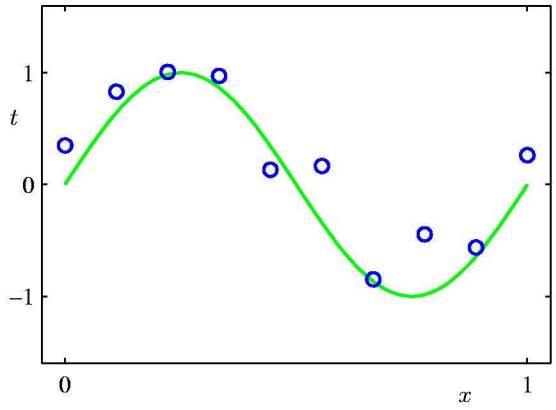


FIGURE 23.2.1. Noisy data.

order of the polynomial) of degree $N-1$ through the N data points. The predicted value then becomes $\widehat{t} = P_N(\widehat{x})$.

This is familiar but not very satisfactory. One reason is that it does not make much sense to force the model through the observations t_n if these are contaminated by noise, as shown in Figure 23.2.1 One can revert to a least squares approximation where the approximation polynomial is no longer required to pass through the data points. In particular one might want to fit a polynomial of the form

$$y(x, \mathbf{w}) = \sum_{j=0}^{M} w_j x^j, \tag{23.1}$$

where M is the degree of the polynomial (one less than the order of the polynomial). The question is how to determine the coefficients $\mathbf{w}$ in order to get a good approximation. First note that if $M \geq N-1$ then we can fit $y(x, \mathbf{w})$ exactly through the data points, and we are back at the interpolation problem. Let us therefore assume that $M < N-1$ in which case we minimize the sum of the squares of the errors between the predictions $y(x_i, \mathbf{w})$ for each data point x_i and the corresponding target values t_i, i.e. we minimize

$$E(\mathbf{w}) = \frac{1}{2} \sum_{i=1}^{N} \left(y(x_i, \mathbf{w}) - t_i \right)^2 . \tag{23.2}$$

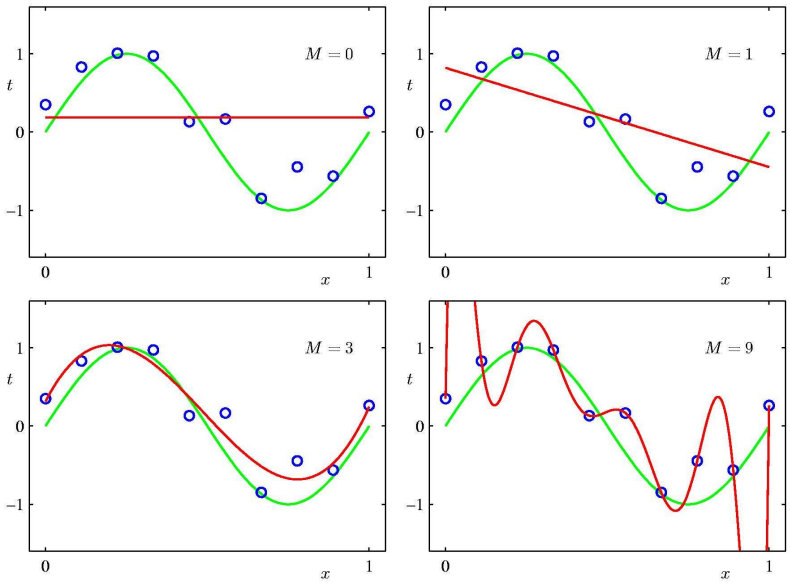


FIGURE 23.2.2. Approximating polynomial (red) of $\sin(2\pi x)$ (green) for various values of M.

EXAMPLE 127. Let us generate data from the function $\sin(2\pi x)$ by first choosing random values of $x \in [0, 1]$. The t values are obtained by evaluating $\sin 2\pi x$ at these random values, corrupted by Gaussian noise. The results are shown in Figure 23.2.2.

The most important observation from this Figure is how bad the approximation becomes for $M = 9$. If one investigates the values of $\mathbf{w}$ for $M = 9$ one finds that the values become very large and oscillatory. One idea therefore is to introduce a penalty or regularization term to the error function in order to prevent large oscillatory values of $\mathbf{w}$,

$$\widetilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}\{y(x_i, \mathbf{w}) - t_i\}^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2 \tag{23.3}$$

where λ governs the relative importance of the regularization term.

From Figure 23.2.3it is clear that the situation has improved dramatically.

A number of question remain:

- How does one determine the value of λ?
- How does one find the optimal value for M ?

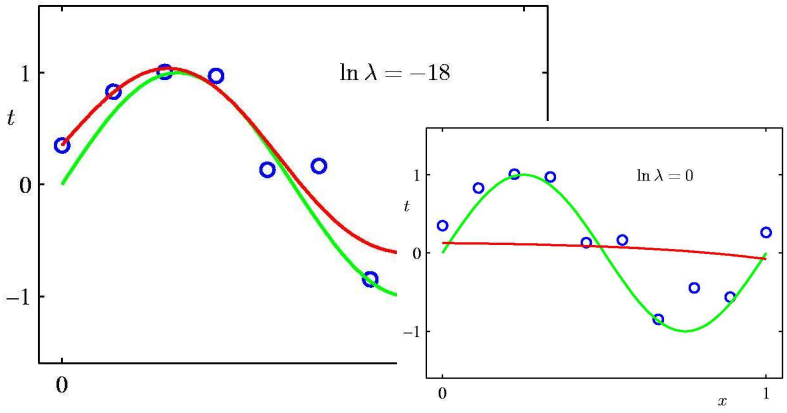


FIGURE 23.2.3. Using $M = 9$ and different values of the regularization constant.

- Are polynomials really the best approximating functions?

23.2.1. A probabilistic view. Moving towards a probabilistic description, we assume that the process we need model is able to provide us with completely observed training data in the form of N input values $\mathbf{x} = \{x_1 \cdots x_N\}$ and corresponding target values $\mathbf{t} = \{t_1 \cdots t_N\}$. The model is generative and parametric where the parameters are inferred from the training data. The model can then be used to predict a target value t for a given input x. Assuming a Gaussian model,

$$p(t|x, \mathbf{w}, \beta) = N(t|y(x, \mathbf{w}), \beta^{-1}),$$

where for the time being it is safe to think of $y(x, \mathbf{w})$ as the polynomial (23.1); β is the precision of the observation noise, see Figure 23.2.4. We'll generalize $y(x, \mathbf{w})$ in a moment. For now, note how this model allows us to generate predictions for given x, once the model parameters $\mathbf{w}$ and β are known.

Assuming the observations to be independent, the likelihood is given by

$$p(\mathbf{t}|\mathbf{x}, \mathbf{w}, \beta) = \prod_{n=1}^{N} N(t_n|y(x_n, \mathbf{w}), \beta^{-1}),$$

and the log-likelihood by

$$\ln p(\mathbf{t}|\mathbf{x}, \mathbf{w}, \beta) = -\frac{\beta}{2} \sum_{n=1}^{N} \{y(x_n, \mathbf{w}) - t_n\}^2 + \frac{N}{2} \ln \beta - \frac{N}{2} \ln(2\pi).$$

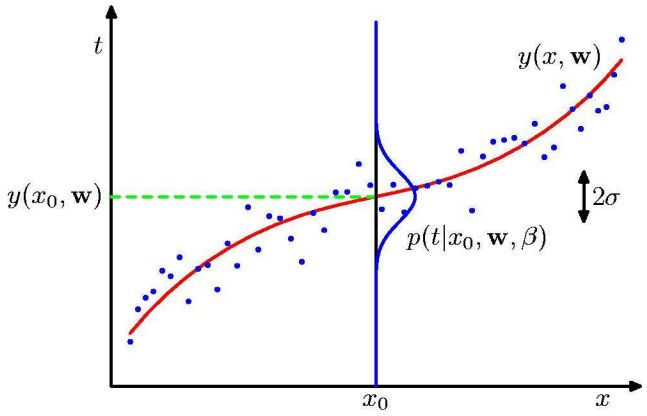


FIGURE 23.2.4. Gaussian distribution for t given x.

The parameters are now obtained by maximizing the log-likelihood. Note that maximizing the log-likelihood with respect to $\mathbf{w}$ is the same as minimizing the sum-of-squares error function (23.2). This means that we get the same solution as (23.2).

Maximizing log-likelihood with respect to β, gives

$$\frac{1}{\beta_{ML}} = \frac{1}{N}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}_{ML}) - t_n\}^2 .$$

Having determined $\mathbf{w}$ and β we can now use these point-wise estimates for the parameters to predict the output t for given input x, using the probability distribution over t,

$$p(t|x, \mathbf{w}_{ML}, \beta_{ML}) = N(t|y(x, \mathbf{w}_{ML}), \beta_{ML}^{-1}).$$

Thus for any given x we are able to generate samples of the target t.

Changing views, the parameters are now treated as random variables, allowing us to write down the posterior distribution over the parameters. Assuming that the observation noise precision β is known, the posterior is given by

$$p(\mathbf{w}|x, t, \beta) \propto p(t|x, \mathbf{w}, \beta)p(\mathbf{w}),$$

where we note the prior over $\mathbf{w}$. Since the likelihood as a function of $\mathbf{w}$ assumes the form of a Gaussian, a conjugate prior is given by the following simplified Gaussian

distribution,

$$p(\mathbf{w}|\alpha) = N(\mathbf{w}|0, \alpha^{-1}I) = \left(\frac{\alpha}{2\pi}\right)^{(M+1)/2} \exp\left\{-\frac{\alpha}{2}\mathbf{w}^T\mathbf{w}\right\},$$

where α is the precision and $M+1$ the order of the polynomial. Although we can make predictions when the posterior is known, it is instructive to calculate a point estimated for $\mathbf{w}$. Taking negative log, the maximum posterior (MAP) estimate is the minimum of

$$\frac{\beta}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2 + \frac{\alpha}{2}\mathbf{w}^T\mathbf{w}.$$

This should look familiar—maximizing the posterior is the same as minimizing the regularized sum-of-squares error (23.3). Also note that it is not necessary to find a point estimate for $\mathbf{w}$—it may be better to use a softer approach and average over the distribution of $\mathbf{w}$, as shown in the next section.

23.2.2. Bayesian curve fitting. In a Bayesian approach we take the uncertainty in $\mathbf{w}$ into account by marginalizing over it. Given the data $\mathbf{x}$, $\mathbf{t}$ and the test point x we want to predict t, i.e. we need $p(t|x, \mathbf{x}, \mathbf{t})$. Using the product and sum rules, we get (omitting the α and β dependence for simplicity),

$$\begin{aligned} p(t|x, \mathbf{x}, \mathbf{t}) &= \int p(t, \mathbf{w}|x, \mathbf{x}, \mathbf{t})d\mathbf{w} \\ &= \int p(t|x, \mathbf{w}, \mathbf{x}, \mathbf{t})p(\mathbf{w}|x, \mathbf{x}, \mathbf{t})d\mathbf{w} \\ &= \int p(t|x, \mathbf{w})p(\mathbf{w}|\mathbf{x}, \mathbf{t})d\mathbf{w}, \end{aligned}$$

noting the conditional independence of $p(t|x, \mathbf{w}, \mathbf{x}, \mathbf{t})$ on $\mathbf{x}$ and $\mathbf{t}$, once $\mathbf{w}$ is known. It turns out (and it is not hard to show) that this is again a Gaussian of the form,

$$p(t|x, \mathbf{x}, \mathbf{t}) = N(t|m(x), s^2(x))$$

with the mean and covariance given by

$$m(x) = \beta\boldsymbol{\phi}(x)^T S \sum_{n=1}^{N}\boldsymbol{\phi}(x_n)t_n$$

$$s^2(x) = \beta^{-1} + \boldsymbol{\phi}(x)^T S\boldsymbol{\phi}(x). \tag{23.4}$$

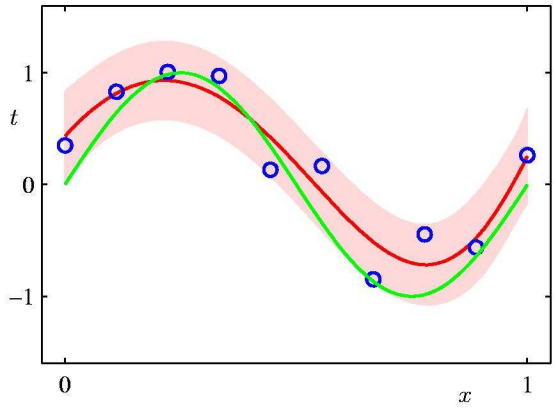


FIGURE 23.2.5. Predictive distribution resulting from a Bayesian treatment.

Here the matrix S is given by

$$S^{-1} = \alpha I + \beta \sum_{n=1}^{N} \boldsymbol{\phi}(x_n)\boldsymbol{\phi}(x_n)^T$$

where I is the identity matrix, and the vector $\boldsymbol{\phi}$ is defined to have the components $\phi_i(x) = x^i,\ i = 1, \ldots, M$.

Note: The first term in (23.4) is the uncertainty in the predicted value of t due to the noise on the target values. The second term is because of the uncertainty in the parameters $\mathbf{w}$ and is a consequence of the Bayesian treatment, see Figure 23.2.5. Also note that in this formulation we bypass the parameters $\mathbf{w}$ and move straight to prediction, given the data. This is a consequence of the convenient mathematical properties of the Gaussian, in particular the fact that marginalizing a Gaussian, produces another Gaussian.

23.3. Linear Models

Consider a linear combinations of fixed, nonlinear functions

$$y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(\mathbf{x})$$

where the ϕ_j are known as the basis functions. This is written in more compact form as

$$y(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}).$$

23.3.1. Maximum likelihood and least squares. Assume that the target value is given by a deterministic function plus noise,

$$t = y(\mathbf{x}, \mathbf{w}) + \epsilon$$

where ϵ is a Gaussian random variable with zero mean and precision β, i.e.

$$p(t|\mathbf{x}, \mathbf{w}, \beta) = \mathcal{N}(t|y(\mathbf{x}, \mathbf{w}), \beta^{-1}).$$

Given data $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ with corresponding target values $t_1, \ldots, t_N$ (drawn independently from a Gaussian distribution), the likelihood is given by,

$$p(\mathbf{t}|X, \mathbf{w}, \beta) = \prod_{n=1}^{N} \mathcal{N}(t_n|\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_n), \beta^{-1})$$

and

$$\begin{aligned} \ln p(\mathbf{t}|X, \mathbf{w}, \beta) &= \sum_{n=1}^{N} \ln \mathcal{N}(t_n|\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_n), \beta^{-1}) \\ &= \frac{N}{2} \ln \beta - \frac{N}{2} \ln(2\pi) - \beta E_D(\mathbf{w}) \end{aligned} \tag{23.1}$$

where

$$E_D(\mathbf{w}) = \frac{1}{2} \sum_{n=1}^{N} \left\{ t_n - \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_n) \right\}^2.$$

This can be rewritten as

$$\begin{aligned} E_D(\mathbf{w}) &= \frac{1}{2} (\mathbf{t} - \boldsymbol{\Phi}\mathbf{w})^T (\mathbf{t} - \boldsymbol{\Phi}\mathbf{w}) \\ &= \frac{1}{2} (\mathbf{t}^T\mathbf{t} - \mathbf{w}^T \boldsymbol{\Phi}^T \mathbf{t} - \mathbf{t}^T \boldsymbol{\Phi}\mathbf{w} + \mathbf{w}^T \boldsymbol{\Phi}^T \boldsymbol{\Phi}\mathbf{w}) \end{aligned}$$

where $\boldsymbol{\Phi}$ is the matrix

$$\boldsymbol{\Phi} = \begin{bmatrix} \boldsymbol{\phi_0}(\mathbf{x_1}) & \cdots & \boldsymbol{\phi_{M-1}}(\mathbf{x_1}) \\ \vdots & \ddots & \vdots \\ \boldsymbol{\phi_0}(\mathbf{x_N}) & \cdots & \boldsymbol{\phi_{M-1}}(\mathbf{x_N}) \end{bmatrix}.$$

Minimizing E_D with respect to $\mathbf{w}$ gives the maximum log-likelihood

$$\nabla_w E_D = -\mathbf{t}^T\mathbf{\Phi} + \mathbf{w}^T\mathbf{\Phi}^T\mathbf{\Phi} = 0,$$

so that

$$\mathbf{w}_{ML} = (\mathbf{\Phi^T\Phi})^{-1}\mathbf{\Phi}^T\mathbf{t}.$$

We also find that

$$\frac{1}{\beta_{ML}} = \frac{1}{N}\sum_{n=1}^{N}\left\{t_n - \mathbf{w}_{ML}^T\boldsymbol{\phi}(\mathbf{x}_n)\right\}^2.$$

23.3.2. Regularized least squares. In order to control over-fitting one can add a regularization term so that the total error term becomes,

$$E_D(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\left\{t_n - \mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_n)\right\}^2 + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w}.$$

Minimizing the error with respect to $\mathbf{w}$ gives

$$\mathbf{w} = \left(\lambda I + \mathbf{\Phi}^T\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}^T\mathbf{t}. \tag{23.2}$$

23.4. Bayesian Linear Regression.

Instead of the likelihood we now turn to the posterior, as ususal. This requires a prior over the parameters. Assuming the noise precision parameter β is known, the conjugate prior is a Gaussian,

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{m}_0, S_0).$$

Since the posterior is given by

$$p(\mathbf{w}|\mathbf{t}) \propto p(\mathbf{t}|\mathbf{w})p(\mathbf{w})$$

with

$$p(\mathbf{t}|\mathbf{w}) = \mathcal{N}(\mathbf{t}|\mathbf{\Phi w}, \beta^{-1}),$$

it follows that

$$p(\mathbf{w}|\mathbf{t}) = \mathcal{N}(\mathbf{w}|\mathbf{m}_N, S_N),$$

where

$$\begin{aligned}\mathbf{m}_N &= S_N(S_0^{-1}\mathbf{m}_0 + \beta\mathbf{\Phi}^T\mathbf{t}) \\ S_N^{-1} &= S_0^{-1} + \beta\mathbf{\Phi}^T\mathbf{\Phi}.\end{aligned}$$

For simplicity consider the prior

$$p(\mathbf{w}|\alpha) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}I)$$

so that

$$\begin{aligned}\mathbf{m}_N &= \beta S_N\mathbf{\Phi}^T\mathbf{t} \\ S_N^{-1} &= \alpha I + \beta\mathbf{\Phi}^T\mathbf{\Phi}.\end{aligned}$$

This form of the prior simply adds a term to the posterior so that we have from (23.1)

$$\ln p(\mathbf{w}|\mathbf{t}) = -\frac{\beta}{2}\sum_{n=1}^{N}\left\{t_n - \mathbf{w}^T\boldsymbol{\phi}(x_n)\right\}^2 - \frac{\alpha}{2}\mathbf{w}^T\mathbf{w} + \text{const.}$$

Maximizing the posterior distribution with respect to $\mathbf{w}$ is therefore equivalent to the minimization of the sum-of-squares error with a quadratic regularization. Note that we have already obtained the solution that minimizes the regularized error term with respect to $\mathbf{w}$, namely (23.2) above.

EXAMPLE. Let $y(x, \mathbf{w}) = w_0 + w_1 x$. Generate data from $y = -0.3 + 0.5x$ by first choosing x from a uniform distribution and then adding Gaussian noise with standard deviation $\frac{1}{\sqrt{\beta}} = 0.2$ (β is the precision). We assume that this is known, i.e. the value of β in the equations above is set to $\beta = \frac{1}{0.2^2} = 25$. The prior (over $\mathbf{w}$) has zero mean and is given the precision matrix $\frac{1}{2.0}I$. The data points are obtained sequentially with the posterior of the previous estimate acting as the prior for the next estimate, illustrated in Figure 23.4.1.

23.4.1. Predictive distribution. In practice we are not interested in the value of $\mathbf{w}$ itself but in predicting t for unseen values of $\mathbf{x}$. Thus we marginalize over $\mathbf{w}$, to get the predictive distribution,

$$p(t|\mathbf{t}, \alpha, \beta) = \int p(t|\mathbf{w}, \beta)p(\mathbf{w}|\mathbf{t}, \alpha, \beta)d\mathbf{w}$$

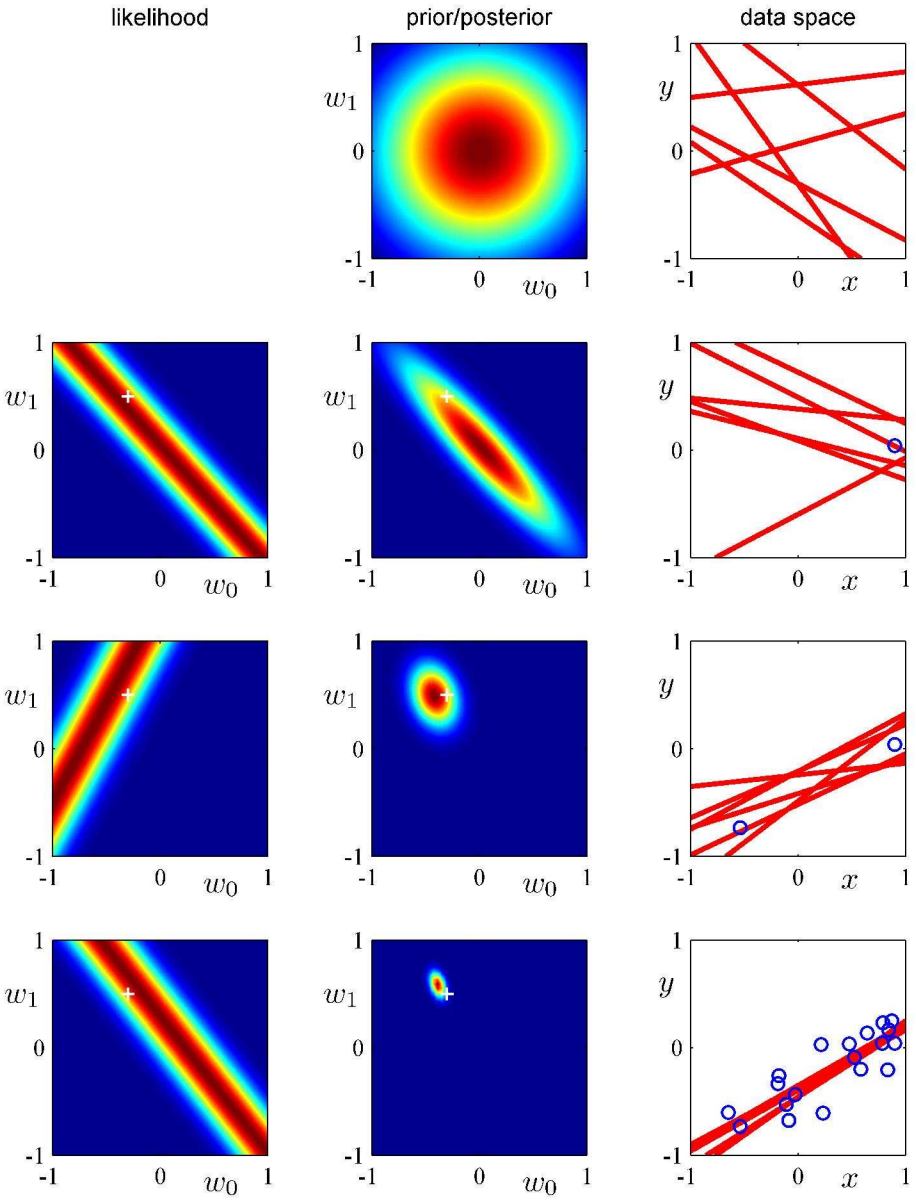


FIGURE 23.4.1. Sequential Bayesian learning.

where $\alpha$is the precision of the prior over $\mathbf{w}$ and β is the precision of the noise in the data. Note that

$$p(t|\mathbf{w}, \beta) = \mathcal{N}(\mathbf{t}|\mathbf{\Phi}^T\mathbf{w}, \beta^{-1})$$

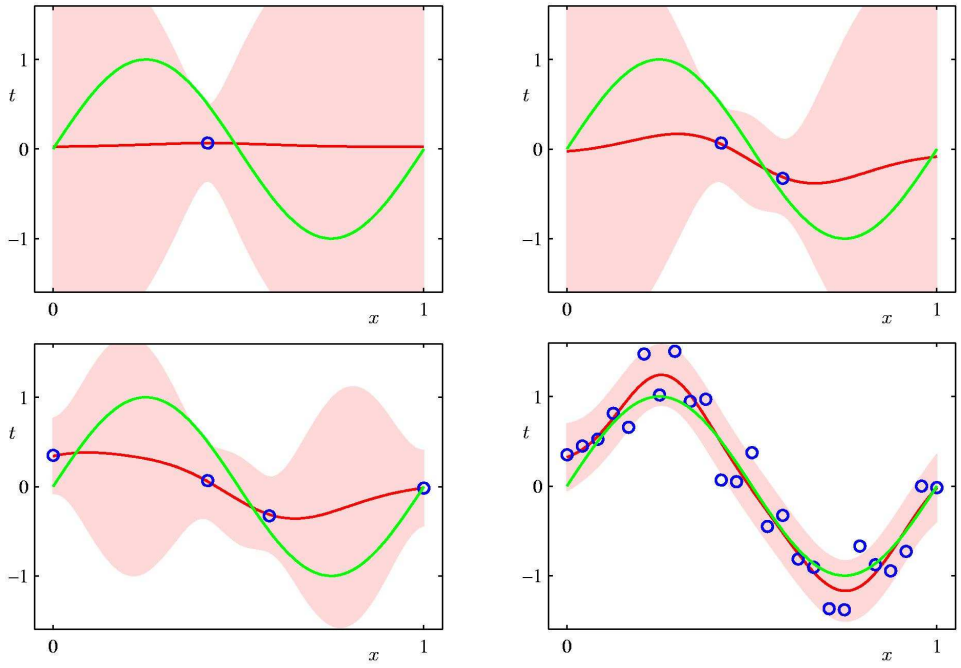


FIGURE 23.4.2. Predictive distribution.

and

$$p(\mathbf{w}|\mathbf{t}, \alpha, \beta) = \mathcal{N}(\mathbf{w}|\mathbf{m}_N, S_N).$$

Since $p(t|\mathbf{t}, \alpha, \beta)$ is the marginalization of two Gaussians we get

$$p(t|\mathbf{t}, \alpha, \beta) = \mathcal{N}(t|\mathbf{m}_N^T \boldsymbol{\phi}(\mathbf{x}), \sigma_N^2(\mathbf{x}))$$

where the variance is given by

$$\sigma_N^2(\mathbf{x}) = \frac{1}{\beta} + \boldsymbol{\phi}(\mathbf{x})^T S_N \boldsymbol{\phi}(\mathbf{x}).$$

This illustrated in Figure 23.4.2

The green curve shows the synthetic (sinusoidal) data set, using a linear combination of Gaussian basis functions (details not specified). This Figure shows the mean of the predictive distribution in red for $N = 1$, $N = 2$, $N = 4$ and $N = 24$ data points. The mean is of course the best estimate that we have for the given number of data points, at any given value of x. We also see (shaded) the standard deviation around the mean as a function of the input value x. We can also look at

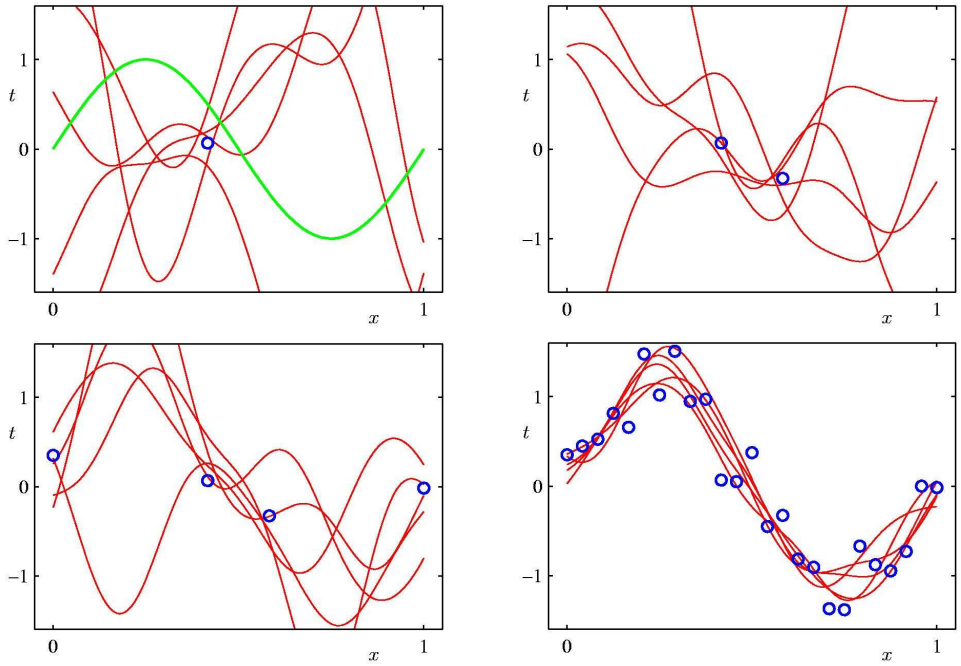


FIGURE 23.4.3. Samples from the posterior distribution of $\mathbf{w}$.

the variations in the predictive function. In order to do this we draw samples of $\mathbf{w}$ from its posterior distribution (for the given data points). For each sample we then draw the predictive curve $y(x, \mathbf{w})$. This is shown in Figure 23.4.3.

23.4.2. Equivalent kernel. There is a powerful alternative interpretation of the posterior mean. Writing the posterior mean as $\mathbf{m}_N$ we get

$$\begin{aligned} y(\mathbf{x}, \mathbf{m}_N) &= \mathbf{m}_N^T \boldsymbol{\phi}(\mathbf{x}) \\ &= \beta \boldsymbol{\phi}(\mathbf{x})^T S_N \boldsymbol{\Phi}^T \mathbf{t} \\ &= \sum_{n=1}^{N} \beta \boldsymbol{\phi}(\mathbf{x})^T S_N \boldsymbol{\phi}(\mathbf{x}_n) t_n. \end{aligned}$$

Defining the kernel

$$k(\mathbf{x}, \mathbf{x}_n) = \beta \boldsymbol{\phi}(\mathbf{x})^T S_N \boldsymbol{\phi}(\mathbf{x}_n)$$

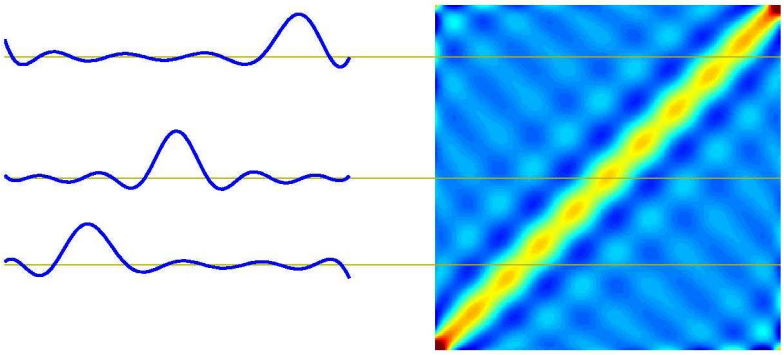


FIGURE 23.4.4. Linear smoother.

we get

$$y(\mathbf{x}, \mathbf{m}_N) = \sum_{n=1}^{N} k(\mathbf{x}, \mathbf{x}_n) t_n.$$

We can think of the kernel as a smoother of the data t_n. Plotting the kernel as in Figure 23.4.4 we see that it gives more weight to the data closest to $\mathbf{x}$ where we want to estimate the value of t.

This provides a general procedure: In general one has to find an appropriate kernel.

Note that

$$\sum_{n=1}^{N} k(\mathbf{x}, \mathbf{x}_n) = 1.$$

This can be seen intuitively as follows. Let us say that all out input values $t_n = 1$, then we need the prediction to equal 1 as well for all input values $\mathbf{x}$.

23.5. Bayesian Model Comparison.

Given observed data $\mathcal{D}$ we consider L different models $\mathcal{M}_i,\ i = 1, \ldots, L$, where each model is a probability observation over the data $\mathcal{D}$. The idea is to select the best model. Our uncertainty in the exact nature of the model is expressed in the prior probability, $p(\mathcal{M}_i)$. Given a training set we are looking for the posterior distribution

$$p(\mathcal{M}_i|\mathcal{D}) \propto p(\mathcal{D}|\mathcal{M}_i) p(\mathcal{M}_i).$$

If we assume equal prior probabilities the interesting term is the model evidence $p(\mathcal{D}|\mathcal{M}_i)$ which describes the preference shown by the data for different models. Once we know the posterior distribution of the model, the predictive distribution is obtained from marginalizing over the different models,

$$\begin{aligned} p(t|\mathbf{x},\mathcal{D}) &= \sum_{i=1}^{L} p(t,\mathcal{M}_i|\mathbf{x},\mathcal{D}) \\ &= \sum_{i=1}^{L} p(t|\mathbf{x},\mathcal{M}_i,\mathcal{D})p(\mathcal{M}_i|\mathbf{x},\mathcal{D}) \\ &= \sum_{i=1}^{L} p(t|\mathbf{x},\mathcal{M}_i,\mathcal{D})p(\mathcal{M}_i|\mathcal{D}), \end{aligned}$$

since the model only depends on the data.

Note that this is an example of how one can combine different models into a single predictive model, hopefully more accurate than any of the individual models. An approximation to this model averaging is to choose the single most probable model alone to make predictions. This is known as model selection.

If the model is governed by a set of parameters $\mathbf{w}$, as we have used up to now, the model evidence is obtained by marginalizing,

$$\begin{aligned} p(\mathcal{D}|\mathcal{M}_i) &= \int p(\mathcal{D},\mathbf{w}|\mathcal{M}_i)d\mathbf{w} \\ &= \int p(\mathcal{D}|\mathbf{w},\mathcal{M}_i)p(\mathbf{w}|\mathcal{M}_i)d\mathbf{w}. \end{aligned}$$

We can try and understand the meaning by making a simple approximation. Assuming a single parameter w, then the posterior distribution for a given model (omitted in the notation) is given by

$$p(w|\mathcal{D}) \propto p(\mathcal{D}|w)p(w).$$

Assuming that the posterior distribution is sharply peaked around w_{MAP} with width $\Delta w_{\mathrm{posterior}}$we approximate the integral by the maximum value of the integrand times the width of the integrand. Also assume that the prior is flat with width

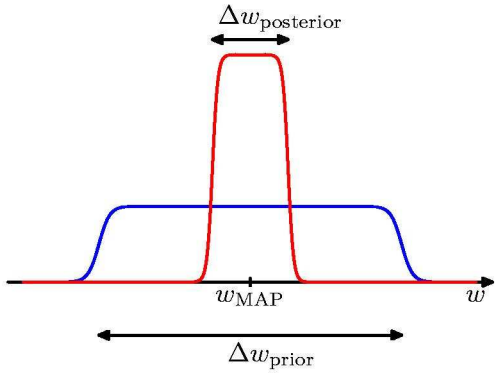


FIGURE 23.5.1. Approximation of the model evidence.

Δw_{prior} so that $p(w) = \frac{1}{\Delta w_{\text{prior}}}$, we find that

$$\begin{aligned} p(\mathcal{D}) &= \int p(\mathcal{D}|w)p(w)dw \\ &\approx p(\mathcal{D}|w_{\text{MAP}})\frac{\Delta w_{\text{posterior}}}{\Delta w_{\text{prior}}}. \end{aligned}$$

Taking logs

$$\ln p(\mathcal{D}) = \ln p(\mathcal{D}|w_{\text{MAP}}) + \ln\left(\frac{\Delta w_{\text{posterior}}}{\Delta w_{\text{prior}}}\right).$$

This approximation is illustrated in Figure 23.5.1.

The first term represents the fit to the data given by the most probable parameter values. The second term penalizes the model according to its complexity. Since $\Delta w_{\text{posterior}} < \Delta w_{\text{prior}}$, it is negative and increases in magnitude as the ratio $\frac{\Delta w_{\text{posterior}}}{\Delta w_{\text{prior}}}$ gets smaller. Thus if the parameters are finely tuned to the data in the posterior distribution, the penalty term is large.

If we have M parameters, and assuming that all parameters have the same ratio $\frac{\Delta w_{\text{posterior}}}{\Delta w_{\text{prior}}}$, we get

$$\ln p(\mathcal{D}) = \ln p(\mathcal{D}|w_{\text{MAP}}) + M\ln\left(\frac{\Delta w_{\text{posterior}}}{\Delta w_{\text{prior}}}\right).$$

Thus the size of the penalty increases linearly with the number of adaptive parameters in the model. If we increase the number of parameters in the model the first term should decrease since we are better able to fit the data. The penalty term on the other hand increases. We need to find the trade-off between these two competing terms.

23.6. Summary.

(1) Want to build a probabilistic model

$$p(t|\mathbf{w}, \beta) = \mathcal{N}(t|\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}), \beta^{-1})$$

by estimating the parameters $\mathbf{w}$ and β from data $\left\{ \begin{array}{cc} \mathbf{x}_n & t_n \end{array} \right\}$, $n = 1, \ldots, N$.

(2) Calculating the parameters using least squares is the same as maximizing the log-likelihood

$$\ln p(\mathbf{t}|\mathbf{w}, \beta).$$

(3) Maximizing the posterior (assuming β to be known), $p(\mathbf{w}|\mathbf{t}) \propto p(\mathbf{t}|\mathbf{w})p(\mathbf{w})$ requires the prior distribution $p(\mathbf{w})$. Assuming $p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}I)$, introduces a hyper parameter α. Maximizing the posterior is the same as least squares with a regularization term.

(4) Since $\mathbf{w}$ is now a full-fledged random variable, one can adopt a full Bayesian treatment and marginalizes over it.

(5) Similarly the hyper-parameters α and β can also be considered to be random variables in which case one can get rid of them by marginalizing over them. Alternatively one can go for a point-wise estimate of the hyper-parameters using the data. A point-wise estimate is obtained from maximizing

$$p(t|\alpha, \beta) = \int p(\mathbf{t}|\mathbf{w}, \beta)p(\mathbf{w}|\alpha)d\mathbf{w}.$$

This is knows as the evidence approximation.

CHAPTER 24

LINEAR MODELS FOR CLASSIFICATION

24.1. Introduction.

We are again observing a generative process. This process generates data of the form $\mathbf{x}_n$ and our task is to assign a given sample $\mathbf{x}_n$ to one of K classes $\mathcal{C}_k$, $k = 1, \ldots, K$, where K is fixed and generally assumed to be known. In the *classification* problem, the training data comes with class labels, i.e. every observation $\mathbf{x}_n$ belongs to a given class $\mathcal{C}_k$. If the data does not come with class labels, we are faced with the problem of identifying the different classes, in which case we face a *clustering* problem. In this case it is still generally assumed that we at least know the number of classes.

Regardless of the training data our task is, given an input $\mathbf{x}$, assign it to a class. As always, the answer is in terms of a probability. Thus, given $\mathbf{x}$ we want to calculate $P(\mathcal{C}_k|\mathbf{x})$ where $\mathcal{C}_k$ is one of the K classes. This probability distribution can be obtained following a generative approach, or a discriminative approach, as indicated in the Introduction. Suppose we know to which class each observation $\mathbf{x}_n$ belongs. For the generative approach a generative model of each class, $p(\mathbf{x}|\mathcal{C}_k)$, is developed from the observations known to belong to that class. Introducing a class prior $P(\mathcal{C}_k)$, the posterior is then given by

$$P(\mathcal{C}_k|\mathbf{x}) \propto p(\mathbf{x}|\mathcal{C}_k)P(\mathcal{C}_k).$$

The alternative, discriminative approach dispenses with the class-conditional distribution $p(\mathbf{x}|\mathcal{C}_k)$, and directly estimates the posterior $P(\mathcal{C}_k|\mathbf{x})$ from the data. This has the advantage that the data is used to best effect to discriminate between the classes, and not 'waste' some to estimate class-conditional models that may not be needed anyway. Each approach has its own strengths and weaknesses.

Before we address this problem we discuss a closely related problem that is of interest in its own right.

24.2. Linear Discriminant Analysis

We start the discussion of this section by developing linear methods separating different classes. It culminated with a linear discriminant method that is widely used to reduce the *dimension* of the problem.

24.2.1. Two Classes. To set the scene, we consider only two classes. The idea is to find a linear function of the input vector $\mathbf{x}$ that maps $\mathbf{x}$ directly onto one of the two classes. This approach is not entirely satisfactory. For one thing, we have no measure of the confidence in the assignment. However, this approach forms the basis of truly useful methods and is therefore worth studying.

A simple example of a linear discriminant function is a linear function of the input vector

$$y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$$

where $\mathbf{w}$ is a weight vector, and w_0 a bias. $\mathbf{x}$ is assigned to class $\mathcal{C}_1$ if $y(\mathbf{x}) \geq 0$ and to $\mathcal{C}_2$ otherwise. The decision boundary is therefore give by $y(\mathbf{x}) = 0$, which is a $D-1$ dimensional hyperplane in the D dimensional input space. One immediate interpretation of w_0 is that of a threshold, since $y(\mathbf{x}) \geq 0$ implies that $\mathbf{w}^T\mathbf{x} \geq -w_0$.

Suppose $\mathbf{x}_A$ and $\mathbf{x}_B$ both lie on the decision boundary then $\mathbf{w}^T(\mathbf{x}_A - \mathbf{x}_B) = 0$ so that $\mathbf{w}$ is orthogonal to the decision boundary.

Let $\mathbf{x}_0 = \alpha\frac{\mathbf{w}}{\|\mathbf{w}\|}$ be the point on the decision boundary closest to the origin, then α is the distance of the decision boundary from the origin. Since $0 = y(\mathbf{x}_0) = \mathbf{w}^T\left(\alpha\frac{\mathbf{w}}{\|\mathbf{w}\|}\right) + w_0$, it follows that $\alpha = -\frac{w_0}{\|\mathbf{w}\|}$. The bias w_0 determines the location of the decision surface, whereas $\mathbf{w}$ determines its direction. More specifically, the decision boundary is a line perpendicular to $\mathbf{w}$.

From Figure 24.2.1 we note that $y(\mathbf{x})$ gives a signed measure of the perpendicular distance r of $\mathbf{x}$ from the decision surface. First write

$$\mathbf{x} = \mathbf{x}_\perp + r\frac{\mathbf{w}}{\|\mathbf{w}\|}.$$

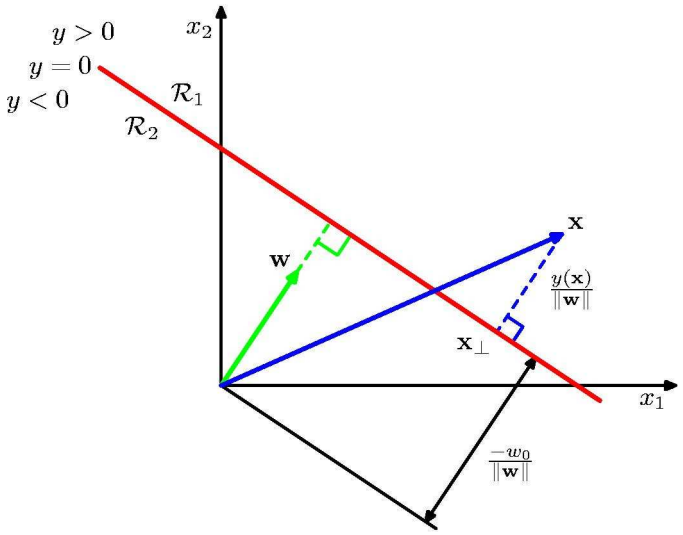


FIGURE 24.2.1. Geometry of the linear discriminant function.

Since $y(\mathbf{x}_\perp) = 0$, i.e. $\mathbf{w}^T\mathbf{x}_\perp + w_0 = 0$ it follows that

$$\begin{aligned} y(\mathbf{x}) &= y(\mathbf{x}_\perp + r\frac{\mathbf{w}}{\|\mathbf{w}\|}) \\ &= \mathbf{w}^T\mathbf{x}_\perp + r\frac{\mathbf{w}^T\mathbf{w}}{\|\mathbf{w}\|} + w_0 \\ &= r\,\|\mathbf{w}\|\,. \end{aligned}$$

so that

$$r = \frac{y(\mathbf{x})}{\|\mathbf{w}\|}.$$

This is quite useful because it softens the hard classification somewhat as it gives us a measure of how close the input vector is to the decision boundary. Note that the linear classifier can also be written as

$$y(\mathbf{x}) = \widetilde{\mathbf{w}}^T\widetilde{\mathbf{x}}.$$

24.2.2. Multiple classes. It is not feasible to build an $K-$class system from a number of two-class systems. Better to use a single K-class system based on K linear functions of the form

$$y_k(\mathbf{x}) = \mathbf{w}_k^T\mathbf{x} + w_{k0}.$$

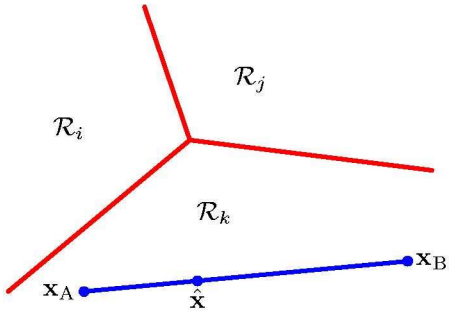


FIGURE 24.2.2. Decision regions for a multi-class linear discriminant function.

A point $\mathbf{x}$ is then assigned to class $\mathcal{C}_k$ if $y_k(\mathbf{x}) > y_j(\mathbf{x})$ for all $j \neq k$. The decision boundary between $\mathcal{C}_k$ and $\mathcal{C}_j$ is therefore given by $y_k(\mathbf{x}) = y_j(\mathbf{x})$ and corresponds to a hyperplane defined by

$$(\mathbf{w}_k - \mathbf{w}_j)^T \mathbf{x} + (w_{k0} - w_{j0}) = 0.$$

This has the same form as the decision surface for the two-class problem.

It is important to note that the decision regions of such a discriminant function are always singly connected and convex. Let $\mathbf{x}_A$ and $\mathbf{x}_B$ both lie inside decision region $\mathcal{R}_k$ as shown in Figure 24.2.2.

Any point $\widehat{\mathbf{x}}$ on the line connecting $\mathbf{x}_A$ and $\mathbf{x}_B$ satisfies

$$\widehat{\mathbf{x}} = \lambda \mathbf{x}_A + (1 - \lambda)\mathbf{x}_B$$

where $0 \leq \lambda \leq 1$. Then

$$y_k(\widehat{\mathbf{x}}) = \lambda y_k(\mathbf{x}_A) + (1 - \lambda) y_k(\mathbf{x}_B).$$

Since $y_k(\mathbf{x}_A) > y_j(\mathbf{x}_A)\ \forall j \neq k$ and $y_k(\mathbf{x}_B) > y_j(\mathbf{x}_B)\ \forall j \neq k$ it follows that $y_k(\widehat{\mathbf{x}}) > y_j(\widehat{\mathbf{x}})\ \forall j \neq k$. This means that $\widehat{\mathbf{x}}$ also lies inside $\mathcal{R}_k$. Thus $\mathcal{R}_k$ is singly connected and convex.

We still need to know how to learn the parameters $\mathbf{w}$.

24.2.3. Least squares for classification. Each class $\mathcal{C}_k$ is described by its own linear model

$$y_k(\mathbf{x}) = \mathbf{w}_k^T \mathbf{x} + w_{k0}, \quad k = 0, \ldots, K$$

that can be combined into a single equation as,

$$\mathbf{y}(\mathbf{x}) = \widetilde{W}^T \widetilde{\mathbf{x}}$$

where

$$\widetilde{W} = \begin{bmatrix} w_{00} & \cdots & w_{K0} \\ \vdots & \ddots & \vdots \\ w_{0D} & \cdots & w_{KD} \end{bmatrix}$$

and $\widetilde{\mathbf{x}} = \begin{bmatrix} 1 & \mathbf{x}^T \end{bmatrix}^T$. An unseen $\mathbf{x}$ is assigned to the class for which the output $y_k = \widetilde{\mathbf{w}}_k^T \widetilde{\mathbf{x}}$ is the largest and we determine the coefficients $\widetilde{W}$ using a least squares technique. We use a 1-of-K coding scheme for the target vector which is far from Gaussian, consequently the least-squares approach is inefficient. It is, among other things, sensitive to outliers. Nevertheless, it is straightforward to obtain the $\mathbf{w}_k$s through least squares, amounting to an exercise in linear algebra.

We assume that we have a fully observable system, i.e. we observe a number of features $\mathbf{x}_n$, $n = 1, \ldots, N$ together with its class label, given by $\mathbf{t}_n$ where $\mathbf{t}_n$ is a K-dimensional vector such that its j-th component is 1if $\mathbf{x}_n$ belongs to class j, and the rest of the components equal zero. Accordingly we observe $\{\mathbf{x}_n, \mathbf{t}_n\}$ $n = 1, \ldots, N$. The trick is to write the equations in matrix form. First we write

$$\begin{bmatrix} y_1(\mathbf{x}_n) \\ \vdots \\ y_K(\mathbf{x}_n) \end{bmatrix} = \begin{bmatrix} \widetilde{\mathbf{x}}_n^T \widetilde{\mathbf{w}}_1 \\ \vdots \\ \widetilde{\mathbf{x}}_n^T \widetilde{\mathbf{w}}_K \end{bmatrix}, \quad n = 1, \ldots, N.$$

Collecting all these equations into a single equation, we get

$$\begin{bmatrix} y_1(\mathbf{x}_1) & \cdots & y_K(\mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ y_1(\mathbf{x}_N) & \cdots & y_K(\mathbf{x}_N) \end{bmatrix} = \begin{bmatrix} \widetilde{\mathbf{x}}_1^T \widetilde{\mathbf{w}}_1 & \cdots & \widetilde{\mathbf{x}}_1^T \widetilde{\mathbf{w}}_K \\ \vdots & \ddots & \vdots \\ \widetilde{\mathbf{x}}_N^T \widetilde{\mathbf{w}}_1 & \cdots & \widetilde{\mathbf{x}}_N^T \widetilde{\mathbf{w}}_K \end{bmatrix} = \begin{bmatrix} \widetilde{\mathbf{x}}_1^T \\ \vdots \\ \widetilde{\mathbf{x}}_N^T \end{bmatrix} \begin{bmatrix} \widetilde{\mathbf{w}}_1 & \cdots & \widetilde{\mathbf{w}}_K \end{bmatrix}.$$

This is rewritten as

$$Y = \widetilde{X} \widetilde{W}$$

where the meanings of $\widetilde{W}$ and $\widetilde{X}$ derive from the previous expression. The n-th column of Y is matched to $\mathbf{t}_n$. If we now define a matrix $T = \begin{bmatrix} \mathbf{t}_1 & \cdots & \mathbf{t}_N \end{bmatrix}$ we need to find $\widetilde{W}$ that minimizes the Frobenius norm $\|Y - T\|_F$, i.e. we need to find

the least squares solution. Let us now stack the columns of T and $\widetilde{W}$ to obtain a system of the form

$$\mathbf{t} = A\widetilde{\mathbf{w}}$$

where $\mathbf{t}$ and $\mathbf{w}$ consist of the columns of Y and $\widetilde{W}$ respectively, and A is a block-diagonal matrix. The least squares solution is then given by $\widetilde{\mathbf{w}} = \left(A^T A\right)^{-1} A^T \mathbf{t}$. If we rearrange we get

$$\widetilde{W} = \left(\widetilde{X}^T \widetilde{X}\right)^{-1} \widetilde{X}^T T^T.$$

Note: As it is written this system is numerically unstable—it is not a good idea to solve the *normal equations* directly. A numerical stable scheme is to first do a QR factorization of A and then solve the normal equations.

24.2.4. Fisher's linear discriminant. As it stands, Fisher's linear discriminant approach does not directly lead to a separation between classes. In fact, that is probably not its primary use. It is however, invaluable as a dimension reduction technique where the reduction aims to give maximum separability between classes. Once the dimension reduction has been achieved, some other classification technique can be employed. First we consider only two classes, projected down to one dimension.

Given a D-dimensional vector $\mathbf{x}$, project it down to one dimension using

$$y = \mathbf{w}^T \mathbf{x}.$$

Note that the interpretation of $\mathbf{w}$ has changed. Before it pointed in a direction orthogonal to the decision boundary. Now it points in the direction of the line onto which we orthogonally project the data points $\mathbf{x}$. Also note that the magnitude of $\mathbf{w}$ is not important, it only scales the data in the transformed space.

Placing a threshold on y and classifying $y \geq -w_0$ as class $\mathcal{C}_1$, and otherwise of class $\mathcal{C}_2$, we again arrive at the linear classifier. The idea is to choose $\mathbf{w}$ in such a way that maximum class separation is achieved in the one-dimensional projected space. One possibility is to maximize the separation between the projected class means. This idea is illustrated in Figure 24.2.3. Calculating the means of the two classes,

$$\mathbf{m}_1 = \frac{1}{N_1} \sum_{n \in \mathcal{C}_1} \mathbf{x}_n, \quad \mathbf{m}_2 = \frac{1}{N_2} \sum_{n \in \mathcal{C}_2} \mathbf{x}_n,$$

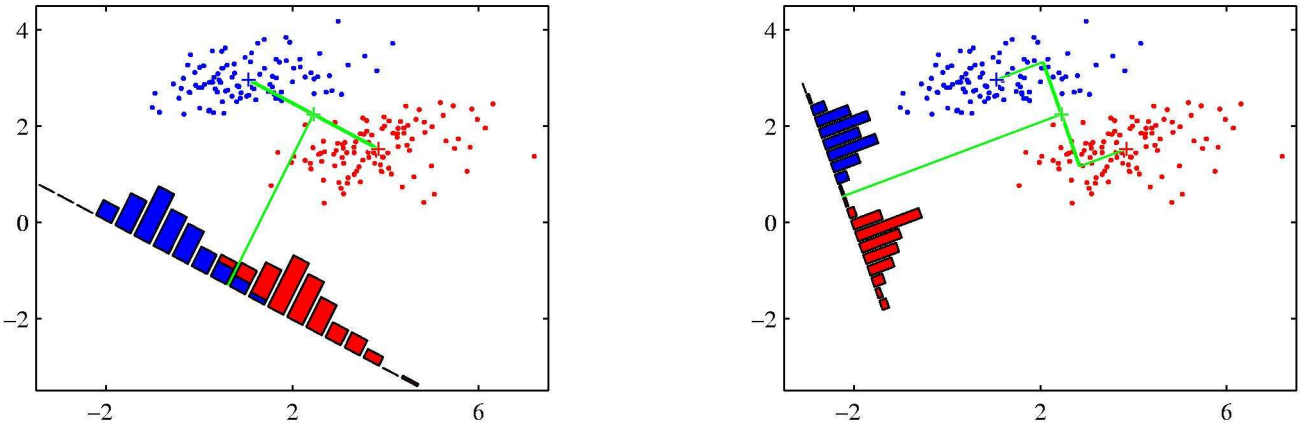


FIGURE 24.2.3. Projection onto the line joining the class means.

we want to maximally separate the projected class means, i.e. we want to choose $\mathbf{w}$ so as to maximize

$$m_2 - m_1 = \mathbf{w}^T(\mathbf{m}_2 - \mathbf{m}_1),$$

where $m_k = \mathbf{w}^T\mathbf{m}_k$. Constraining $\mathbf{w}$ by $\mathbf{w}^T\mathbf{w} = 1$, a Lagrange multiplier λ is introduced so that we minimize

$$L(\mathbf{w}) = \mathbf{w}^T(\mathbf{m}_2 - \mathbf{m}_1) + \lambda(\mathbf{w}^T\mathbf{w} - 1)$$

subject to the constraint $\mathbf{w}^T\mathbf{w} = 1$. From $\frac{\partial L}{\partial w_n} = 0$ follows

$$(m_{2n} - m_{1n}) + 2\lambda w_n = 0,\ n = 1, \ldots, D.$$

subject to $\mathbf{w}^T\mathbf{w} = 1$. In vector notation this looks even better,

$$\mathbf{m}_2 - \mathbf{m}_1 + 2\lambda\mathbf{w} = \mathbf{0},$$

or

$$\mathbf{w} \propto \mathbf{m}_2 - \mathbf{m}_1.$$

This means that $\mathbf{w}$ points in the direction of the vector connecting the two class means, implying that the decision boundary is orthogonal to this direction, see Figure 24.2.3(a).

This is not the best one can do, as seen in Figure 24.2.3(b). Not only do we want to maximize the separation of the projected class means but, simultaneously, we also want to minimize class overlap by minimizing the total projected within-class scatter. The within-class scatter for each of the two classes, is given by

$$s_k^2 = \sum_{n \in \mathcal{C}_k} (y_n - m_k)^2$$

where $y_n = \mathbf{w}^T \mathbf{x}_k$, and the total within-class scatter by $s_1^2 + s_2^2$. The Fisher criterion is to maximize $J(\mathbf{w})$ where

$$\begin{aligned} J(\mathbf{w}) &= \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2}. \\ &= \frac{\mathbf{w}^T S_B \mathbf{w}}{\mathbf{w}^T S_W \mathbf{w}}, \end{aligned}$$

where

$$S_B = (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^T$$

and

$$S_W = \sum_{n \in \mathcal{C}_1} (\mathbf{x}_n - \mathbf{m}_1)(\mathbf{x}_n - \mathbf{m}_1)^T + \sum_{n \in \mathcal{C}_2} (\mathbf{x}_n - \mathbf{m}_2)(\mathbf{x}_n - \mathbf{m}_2)^T.$$

$J(\mathbf{w})$ is maximized when

$$(\mathbf{w}^T S_B \mathbf{w}) S_W \mathbf{w} = (\mathbf{w}^T S_W \mathbf{w}) S_B \mathbf{w}.$$

Note that since we are only interested in the direction of $\mathbf{w}$ not its magnitude, we can drop the scalar factors. Also $S_B \mathbf{w} = (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^T \mathbf{w}$ points in the direction of $\mathbf{m}_2 - \mathbf{m}_1$ and we get that

$$\mathbf{w} \propto S_W^{-1} (\mathbf{m}_2 - \mathbf{m}_1).$$

The attentive reader would have noted that the within-class scatter for each class is not normalized by the number of points in that class. This means that less weight is attached to classes with fewer members.

24.2.5. Fisher's discriminant for multiple classes. Moving on to multiple classes, one can think of Fischer's scheme as a way to do dimensionality reduction,

optimized for classification. The goal is to reduce the dimension to get maximal separation between the classes. There is another common type of (linear) dimensionality reduction where the goal is to preserve the 'geometric' structure of the data space. More of that later, for now we concentrate on classification.

We are given a training set consisting of N data values, each of dimension D, and each one with a class label indicating to which one of K classes the value belongs. We also assume that the dimensionality $D \geq K$. Introduce $D' < D$ linear features $y_k = \mathbf{w}_k^T \mathbf{x}, \;\; k = 1, \ldots, D'$ written in matrix form as

$$\mathbf{y} = W^T \mathbf{x} \tag{24.1}$$

where

$$W = \left[\begin{array}{ccc} \mathbf{w}_1 & \cdots & \mathbf{w}_{D'} \end{array} \right].$$

The D dimensional features $\mathbf{x}$ are therefore projected onto a lower dimensional space, D'. Our task is to find the lower dimensional space with maximal class separation. This of course amounts to finding W. The main idea is to find a measure $J(W)$ of the class separation. Once $J(W)$ is fixed it becomes a matter of finding the optimal W^*,

$$W^* = \arg \max_W J(W).$$

We start by defining total within— and between class scatter. The total within-class scatter is defined by

$$S_w = \sum_{k=1}^{K} P_k \Sigma_k \tag{24.2}$$

where Σ_k is the biased class covariance matrix and P_k is the class probability, $P_k = \frac{N_k}{N}$, where N_k is the number of data values belonging to class k, and N is the total number of data values.

If $\mathbf{m}$ is the mean of the total data set,

$$\mathbf{m} = \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}_n = \sum_{n=1}^{K} P_k \mathbf{m}_k, \tag{24.3}$$

where $\mathbf{m}_k$ is the class mean, the between-class scatter is given by

$$S_b = \sum_{k=1}^{K} P_k(\mathbf{m}_k - \mathbf{m})(\mathbf{m}_k - \mathbf{m})^T. \tag{24.4}$$

We now define similar matrices on the D'-dimensional linear, projected spaces (using $\mathbf{y} = W^T\mathbf{x}$),

$$s_w = \sum_{k=1}^{K} P_k \Sigma_{yk},$$

where Σ_{yk} is the projected class covariance, and

$$s_b = \sum_{k=1}^{K} P_k(\boldsymbol{\mu}_k - \boldsymbol{\mu})(\boldsymbol{\mu}_k - \boldsymbol{\mu})^T$$

where $\boldsymbol{\mu}_k$ is the projected class mean, and $\boldsymbol{\mu}$ is the projected total mean.

Intuitively we again need to simultaneously maximize between-class scatter and minimize within-class scatter. There are a number of suitable objective functions one can use. The one we consider is given by

$$J(W) = \operatorname{tr}\left\{s_w^{-1} s_b\right\}.$$

Written in terms of the original, known quantities, it becomes

$$J(W) = \operatorname{tr}\left\{(W S_w W^T)^{-1}(W S_b W^T)\right\}.$$

EXERCISE 128. Show that $s_w = W S_w W^T$ and $s_b = W S_b W^T$.

Recall that the idea is to maximize $J(W)$ over W, i.e. we need $W^* = \arg\max_W J(W)$ such that W^* defines a projection from our original data space to a new data space with maximal class separability. The data points $\mathbf{x}_n$ are projected to

$$\mathbf{y}_n = W^{*T}\mathbf{x}_n.$$

Also note that this is not a classifier; all it does is to project the data onto a lower dimensional space suitable for classification.

We now proceed to finding the optimal transformation.

24.2.6. Derivation of the optimal transformation. Conceptually, deriving the optimal transform is quite simple. Since the trace of a matrix equals the sum of

its eigenvalues, it is immediately clear that we need to maximize the eigenvalues of $s_w^{-1}s_b$. Written in terms of the given data, it follows that

$$\begin{aligned} J(W) &= \operatorname{tr}\left(s_w^{-1}s_b\right) \\ &= \operatorname{tr}\left(\left(W^T S_w W\right)^{-1}\left(W^T S_b W\right)\right), \end{aligned}$$

as noted above. As usual the optimum is obtained by setting $\frac{\partial J(W)}{\partial W} = 0$. The only aspect that might be somewhat unusaul is taking the derivative of the trace of a matrix. However this is well-documented, see for example Fukanaga, and is given by,

$$\left(S_w^{-1}S_b\right)W = W\left(s_w^{-1}s_b\right). \tag{24.5}$$

This is of the form of a similarity transformation, except for the fact that W is rectangular and does not have an inverse. It is a simple matter to show that the eigenvalues of $s_w^{-1}s_b$ are also eigenvalues of $S_w^{-1}S_b$, and if $\mathbf{b}$ is an eigenvector of $s_w^{-1}s_b$ then $W\mathbf{b}$ is an eigenvector of $S_w^{-1}S_b$. To wit, if B is the eigenvector matrix of $s_w^{-1}s_b$,

$$s_w^{-1}s_b B = B\Xi$$

where Ξ is the diagonal eigenvalue matrix, then

$$S_w^{-1}S_b WB = WB\Xi.$$

Note that the converse is not true. The eigenvalues of $S_w^{-1}S_b$ are not (necessarily) the eigenvalues of $s_w^{-1}s_b$; since $S_w^{-1}S_b$ is of higher dimension it has more eigenvalues than $s_w^{-1}s_b$. More specifically, the D' eigenvalues of $s_w^{-1}s_b$ form a subset of the D eigenvalues of $S_w^{-1}S_b$. Now we get the the heart of the argument: The trace of $s_w^{-1}s_b$ equals the sum of its eigenvalues. If we want to maximize its trace we need to maximize the sum of its eigenvalues. Since its D' eigenvalues are eigenvalues of $S_w^{-1}S_b$ we need to somehow ensure that W automatically selects the D' largest eigenvalues from among the D eigenvalues of $S_w^{-1}S_b$. All we need to do is to choose for WB the eigenvectors of $S_w^{-1}S_b$ belonging to its D' largest eigenvalues.

That's it, except for a small subtlety.

We calculate WB and since we don't know B, we cannot find W by itself. The good news is that it is not necessary to solve for W from the product WB. The difference between the product WB and W is just multiplication by an invertible square

matrix B. This implies another coordinate transformation from the y coordinates to new z coordinates through $\mathbf{z} = B^T\mathbf{y}$. This transformation leaves the fundamental quantity $\mathrm{tr}\,(s_w^{-1}s_b)$ invariant. To see this, suppose that $\mathbf{z} = B^T\mathbf{y}$ transforms s_w and s_b to z_w and z_b respectively. This means that $z_w = B^T s_w B$ and $z_b = B^T s_b B$. A quick calculation shows that

$$\begin{aligned}\mathrm{tr}\left(z_w^{-1}z_b\right) &= \mathrm{tr}\left(\left(B^T s_w B\right)^{-1}\left(B^T s_b B\right)\right)\\ &= \mathrm{tr}\left(B^{-1}s_w^{-1}s_b B\right)\\ &= \mathrm{tr}\left(s_w^{-1}s_b\right).\end{aligned}$$

In this case the eigenvalues of $z_w^{-1}z_b$ and $s_w^{-1}s_b$ are *identical*. The conclusion is, the presence of B does no harm, it does not change the value of the trace of $s_w^{-1}s_b$; we therefore choose the matrix consisting of the D' largest eigenvectors of $S_w^{-1}S_b$ as our transformation matrix.

Finally, a word on the calculation of the eigenvectors of $S_w^{-1}S_b$. Although S_w and S_b are both symmetric, the product $S_w^{-1}S_b$ is not necessarily symmetric. For this, and other reasons, we do not want to calculate the eigenvectors of $S_w^{-1}S_b$ directly. It is better to calculate it indirectly by calculating the matrix that simultaneously diagonalizes S_w and S_b as explained below.

24.2.6.1. *Simultaneous diagonalization of two symmetric matrices.* Suppose we are given two symmetric matrices S_w and S_b and we are looking for a transformation that will transform both of them to diagonal matrices. To be more specific, we are looking for a matrix B such that $B^T S_w B = I$, and $B^T S_b B = \Lambda$, where I is the identity matrix and Λ is a diagonal matrix.

Geometrically the idea is very simple. Let us assume that each covariance matrix represents a Gaussian distribution centered at the origin. If we sample from each distribution the data will be scattered in the shape of two ellipsoids, both centered at the origin. First we whiten the data generated by S_w by transforming it to an identity matrix. We apply the same transformation to S_b, noting that the transformed S_b is again a covariance matrix. At this stage we can think of the data generated by S_w as being scattered in the spherical shape, while the data generated by S_b again scattered in an elliptical shape. All that now remains to be done is to rotate the data

associated with S_b so that its ellipsoid is aligned along the coordinate axes. Noting that a rotation does not change the shape of a sphere, we apply the same rotation to transformed S_w. We are done.

EXERCISE 129. Is it in general possible to simultaneously diagonalize more than two covariance matrices?

Let us now describe the procedure step-by-step:

(1) If S_w is non-singular, whiten S_w by factorizing $S_w = LL^T$ by using for example, Cholesky factorization, or the Singular Value Decomposition (SVD) which is more expensive than Cholesky. If we use the SVD, we write $S_w = U\Sigma U^T = U\Sigma^{1/2}\left(U\Sigma^{1/2}\right)^T =: LL^T$. This means that $L^{-1}S_wL^{-T} = I$, where $L = U\Sigma^{1/2}$.
(2) In the case that S_w is singular we 'remove' the singularity using the SVD, $S_w = U\Sigma U^T$. More specifically, assume that the rank of S_w is r. By retaining only the first r columns of U, written as $\widehat{U}$, and the r nonzero singular values of Σ, written as Σ_+, the reduced form of the SVD is written as $S_w = \widehat{U}\Sigma_+\widehat{U}^T$, so that $\Sigma_+^{-\frac{1}{2}}\widehat{U}^TS_w\widehat{U}\Sigma_+^{-\frac{1}{2}} = I$. The algorithm now proceeds with $L = \widehat{U}\Sigma_+^{\frac{1}{2}}$. It is worth noting that $\widehat{U}$ is no longer square and therefore does not have an inverse. However, since its remaining columns are still orthonormal, $\widehat{U}^T\widehat{U} = I$, but $\widehat{U}\widehat{U}^T \neq I$. $L = \widehat{U}\Sigma_+^{\frac{1}{2}}$is therefore also no longer square, without a regular inverse. It does however, have an inverse from the left since $\left(\Sigma_+^{-\frac{1}{2}}\widehat{U}^T\right)\left(\widehat{U}\Sigma_+^{\frac{1}{2}}\right) = I$, all that is needed.
(3) Perform the same transformation to S_b. Calculate $K = L^{-1}S_bL^{-T}$, where $L^{-1} = \Sigma_+^{-\frac{1}{2}}\widehat{U}^T$ and $L^{-T} = \widehat{U}\Sigma_+^{-\frac{1}{2}}$, if S_w is singular.
(4) Since K is also a covariance matrix, we 'rotate' it so that the associated ellipsoid coincides with the coordinate axes. Accordingly, we diagonalize K using its orthonormal eigenvector matrix Q, i.e. $K = Q\Lambda Q^T$ where $QQ^T = I = Q^TQ$. This means that $Q^TKQ = \Lambda$.
(5) Substituting the expression for K, it follows that

$$Q^TL^{-1}S_bL^{-T}Q = \Lambda.$$

Rotating the transformed S_w by Q leaves it invariant,

$$Q^T L^{-1} S_w L^{-T} Q = Q^T I Q = I.$$

(6) For $H = L^{-T}Q$ we therefore find that $H^T S_w H = I$ and $H^T S_b H = \Lambda$ as required.

Note:

(1) H is just the eigenvector matrix of $S_w^{-1}S_b$ when S_w^{-1} exists with the corresponding eigenvalues given by Λ. To show this, note that

$$\begin{aligned} S_w^{-1}S_b &= \left(H^{-T}H^{-1}\right)^{-1} H^{-T}\Lambda H^{-1} \\ &= H\Lambda H^{-1}, \end{aligned}$$

which is rewritten as

$$S_w^{-1}S_b H = H\Lambda.$$

It is therefore clear that H is the eigenvector matrix of $S_w^{-1}S_b$ and Λ is the corresponding eigenvalue matrix.

(2) The procedure above therefore provides a way of calculating the eigenvalues and eigenvectors of $S_w^{-1}S_b$ without the need of calculating the inverse of S_w. In fact we showed how we can calculate H that simultaneously diagonalizes S_w and S_b even if S_w is singular. Thus we have solved the generalized eigenvalue problem

$$S_b H = H\Lambda S_w.$$

The most important observation is that H is exactly the transformation matrix needed for dimensionality reduction. We have therefore arrived at an expression for W in (24.1),

$$\begin{aligned} W &= L^{-T}Q \\ &= \widehat{U}\Sigma_+^{-\frac{1}{2}}Q, \end{aligned} \tag{24.6}$$

where we use the expression appropriate for squeezing out the 'empty' dimensions corresponding to zero singular values. Of course if S_w in not singular, $\widehat{U}$ and Σ_+ become just the full matrices U and Σ respectively.

Let us see what exactly has been achived so far. Recall that the projection is given by $\mathbf{y} = W^T\mathbf{x}$ where $\mathbf{x} \in \mathbb{R}^D$. From (24.6) follows that

$$W^T = Q^T\Sigma_+^{-\frac{1}{2}}\widehat{U}^T.$$

Since Q is an $r \times r$ matrix where $r \leq D$ is the rank of S_w, $W^T$is an $r \times D$ matrix. Thus apart from squeezing out the 'empty' dimensions of S_w we have not yet achieved any dimensionality reduction. For that we need to investigate the eigenvalues associated with the eigenvector matrix Q. Since we want to project down to D' dimensions, we select the eigenvectors in Q belonging to the D' largest eigenvalues. It is interesting to note that there is an upper limit on the choice of $D' \leq K-1$, where K is the number of classes, since the maximum rank of S_b is $K-1$. To wit, S_b is the sum of K rank-one matrices that satisfy a single constraint (24.3). More specifically, (24.4) tells us that S_b is a linear combination of rank one matrices based on the vectors $\mathbf{m}_k - \mathbf{m}, \quad k = 1\ldots, K$ satisfying the constraint (24.3). The constraint implies that the vectors $\mathbf{m}_k - \mathbf{m}, \quad k = 1\ldots, K$ are linearly dependent (by noting that $\sum_{k=1}^K P_k(\mathbf{m}_k - \mathbf{m}) = 0$). Therefore the rank of S_b is at most $K-1$.

24.3. Probabilistic Generative Models.

We now get to the main goal of this chapter. Recall that we are studying a generative system that generates data $\mathbf{x}_n$, and each data point is associated with a class label. Let us now assume that the training data comes with class labels. Thus for each observation $\mathbf{x}_n$ we are given a corresponding t_n, indicating the class. For example, t_n can be an K-dimensional vector consisting of a 1 in the $k-$th position if $\mathbf{x}_n$ belongs to $\mathcal{C}_k$, and zeros in all the other positions. Using the training data we obtain the class-conditional densities $p(\mathbf{x}|\mathcal{C}_k)$. Given priors $P(\mathcal{C}_k)$, Bayes' theorem then gives the posterior probabilities $P(\mathcal{C}_k|\mathbf{x})$.

Consider the case of only two classes with $P(\mathcal{C}_2|\mathbf{x}) + P(\mathcal{C}_1|\mathbf{x}) = 1$,

$$\begin{aligned} P(\mathcal{C}_1|\mathbf{x}) &= \frac{p(\mathbf{x}|\mathcal{C}_1)P(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)P(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)P(\mathcal{C}_2)} \\ &= \frac{1}{1+\exp(-a(\mathbf{x}))} \\ &= \sigma(a(\mathbf{x})) \end{aligned} \tag{24.1}$$

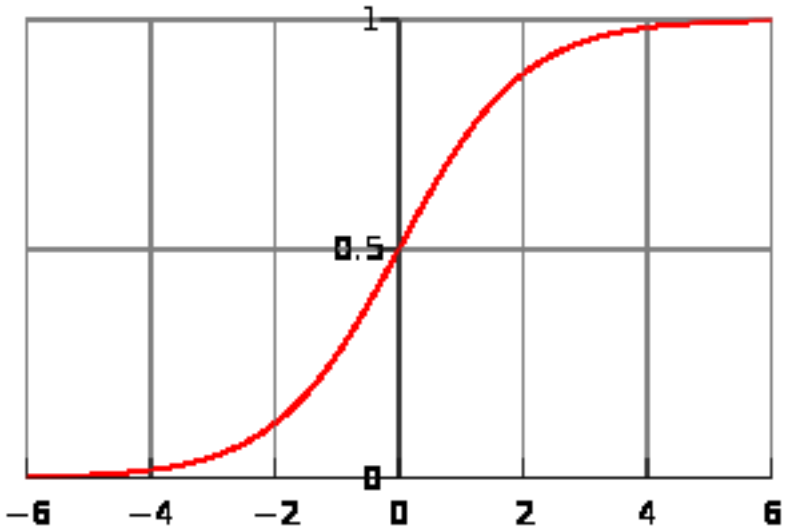


FIGURE 24.3.1. The logistic sigmoid function $\sigma(a)$.

where

$$a(\mathbf{x}) = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)P(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)P(\mathcal{C}_2)},$$

and

$$\sigma(a) = \frac{1}{1+\exp(-a)}$$

is known as the l*ogistic sigmoid* function. Note that the logistic function maps $a(\mathbf{x})$ to a value between 0 and 1. It therefore assigns a probability—the posterior probability—to each input value $\mathbf{x}$, see Figure˜24.3.1.

For more than two classes we have

$$\begin{aligned} P(\mathcal{C}_k|\mathbf{x}) &= \frac{p(\mathbf{x}|\mathcal{C}_k)P(\mathcal{C}_k)}{\sum_{j=1}^{K} p(\mathbf{x}|\mathcal{C}_j)P(\mathcal{C}_j)} \\ &= \frac{\exp a_k(\mathbf{x})}{\sum_{j=1}^{K} \exp a_j(\mathbf{x})} \end{aligned}$$

where

$$a_k(\mathbf{x}) = \ln p(\mathbf{x}|\mathcal{C}_k) + \ln P(\mathcal{C}_k).$$

It is straightforward to verify that $\sum_{j=1}^{K} P(\mathcal{C}_j|\mathbf{x}) = 1$. This normalized sum of exponentials is also known as the softmax function because it is a smoothed version of the 'max' function. If $a_k >> a_j, \;\; j \neq k$, then $P(\mathcal{C}_k|\mathbf{x}) \approx 1$ and $P(\mathcal{C}_j|\mathbf{x}) \approx 0$.

24.3.1. Gaussian class-conditional pdf's. Assume that the class-conditional densities are Gaussian, sharing the *same* covariance matrix (but not the same means) so that the class-conditional densities are given by,

$$p(\mathbf{x}|\mathcal{C}_k) = \frac{1}{|2\pi\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)^T\right\}.$$

For the case of two classes we have

$$P(\mathcal{C}_1|\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x} + w_0) \tag{24.2}$$

where

$$\mathbf{w} = \Sigma^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \tag{24.3}$$

$$w_0 = -\frac{1}{2}\boldsymbol{\mu}_1^T\Sigma^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^T\Sigma^{-1}\boldsymbol{\mu}_2 + \ln\frac{P(\mathcal{C}_1)}{P(\mathcal{C}_2)}. \tag{24.4}$$

Note:

(1) The prior probabilities only enter through the bias term w_0.

(2) The quadratic terms in the Gaussians cancel because of the shared covariance. This is what makes it a *linear* classifier. To see this, we classify $\mathbf{x}$ as belonging to $\mathcal{C}_1$ if $P(\mathcal{C}_1|\mathbf{x}) > P(\mathcal{C}_2|\mathbf{x})$ otherwise to $\mathcal{C}_2$, with the decision boundary given by $P(\mathcal{C}_1|\mathbf{x}) = P(\mathcal{C}_2|\mathbf{x}) = 1 - P(\mathcal{C}_1|\mathbf{x})$. This can be rewritten as $\sigma(\mathbf{w}^T\mathbf{x} + w_0) = 1 - \sigma(\mathbf{w}^T\mathbf{x} + w_0)$, or $\mathbf{w}^T\mathbf{x} + w_0 = \text{const}$.

(3) We have just achieved a remarkable result, one that will be exploited in full in the next section. For the time being, just note the parameter reduction in (24.2). If we have D-dimensional data, then the two classes with different means but a shared covariance, require a total of $2D + \frac{1}{2}D(D+1)$ parameters. In addition one has to assign prior class probabilities. In (24.2) we combine the parameters in such a way that we are effectively left with only $D + 1$ parameters, a significant savings. One can therefore ask whether one can side-step the class-conditional pdf's and directly estimate the $D + 1$ parameters in (24.2). This idea is pursued in the next section.

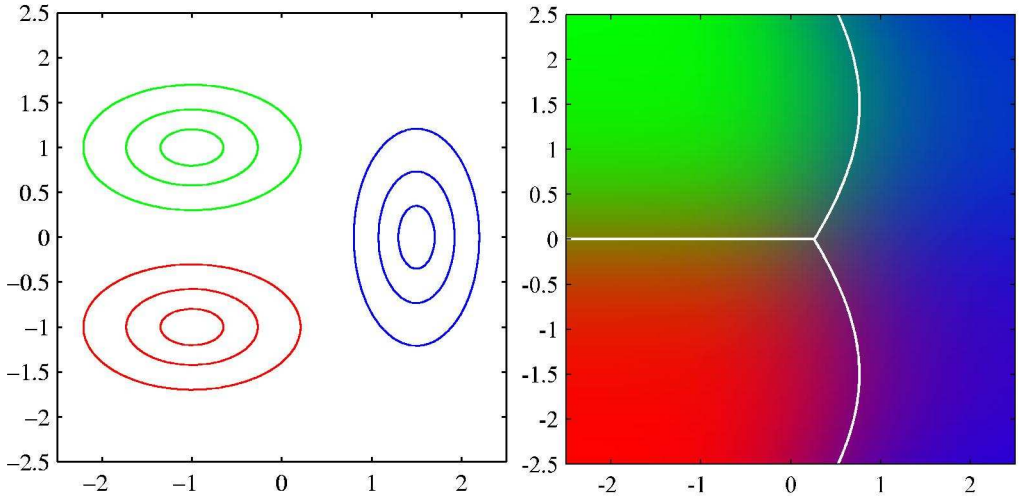


FIGURE 24.3.2. Quadratic discriminant functions.

For K classes we have

$$P(\mathcal{C}_k|\mathbf{x}) = \frac{\exp a_k(\mathbf{x})}{\sum_{j=1}^{K} \exp a_j(\mathbf{x})},$$

where

$$a_k(\mathbf{x}) = \mathbf{w}_k^T \mathbf{x} + w_{k0},$$

with

$$\begin{aligned} \mathbf{w}_k &= \Sigma^{-1}\boldsymbol{\mu}_k \\ w_{k0} &= \frac{1}{2}\boldsymbol{\mu}_k^T\Sigma^{-1}\boldsymbol{\mu}_k + \ln P(\mathcal{C}_k). \end{aligned}$$

Because of the shared covariance, the decision boundary is again linear. For non-shared covariance matrices the decision boundaries become quadratic, see Figure 24.3.2.

24.3.2. Maximum likelihood solution. In order to find the parameters of the class conditional densities as well as the prior densities we need a data set of observations including their class labels. Since the data is completely observed, i.e. we know to which class each data point belongs, one can easily estimate the parameters for each class-conditional density function. Assuming two classes and Gaussian class-conditional density functions, we can easily estimate the mean and covariance of each class. If we also let $P(\mathcal{C}_1) = \pi$ and $P(\mathcal{C}_2) = 1 - \pi$, a maximum

likelihood estimate gives

$$\pi = \frac{N_1}{N},$$

i.e. the prior probability of belonging to class $\mathcal{C}_1$ equals the fraction of the data points belonging to $\mathcal{C}_1$. The mean of the two classes are estimated by

$$\boldsymbol{\mu}_1 = \frac{1}{N_1}\sum_{n=1}^{N} t_n \mathbf{x}_n \text{ and } \boldsymbol{\mu}_2 = \frac{1}{N_2}\sum_{n=1}^{N}(1-t_n)\mathbf{x}_n,$$

where $t_n = 1$ if $\mathbf{x}_n \in \mathcal{C}_1$ and $t_n = 0$ if $\mathbf{x}_n \in \mathcal{C}_2$. The respective covariances are given by

$$\Sigma_1 = \frac{1}{N_1}\sum_{n\in\mathcal{C}_1}(\mathbf{x}_n - \boldsymbol{\mu}_1)(\mathbf{x}_n - \boldsymbol{\mu}_1)^T \text{ and } \Sigma_2 = \frac{1}{N_2}\sum_{n\in\mathcal{C}_2}(\mathbf{x}_n - \boldsymbol{\mu}_2)(\mathbf{x}_n - \boldsymbol{\mu}_2)^T.$$

The only (slight) complication arises if we assume a shared covariance, in which case it is given by

$$\Sigma = P_1\Sigma_1 + P_2\Sigma_2,$$

where P_1 and $P_2 = 1 - P_1$ are the class probabilities, $P_1 = \frac{N_1}{N}$.

The extension to K classes is rather straightforward.

24.3.3. Naive Bayes classifier. The naive Bayes classifier is a simple yet useful classifier. Let us first point out a difficulty with the previous approach. The difficulty is entirely computational and has to do with the number of parameters one has to estimate. Assuming a K-class problem in D dimensions and a different, full covariance matrix for each class (leading to quadratic, not linear decision surfaces), one has to estimate $D+\frac{1}{2}D(D+1)$ parameters for each class, i.e. a total of $\frac{1}{2}KD(D+3)$ parameters for K classes. This can easily become unwieldy when D becomes large. One possibility is to share a full covariance between all the classes as we did above. In this case the number of parameters is reduced to $KD+\frac{1}{2}D(D+1)$. This may not be 'rich' enough to provide good descriptions of each class. Naive Bayes is somewhere in-between, essentially assuming a *diagonal* covariance for each class. This reduces the number of parameters to $2KD$, a substantial saving especially if D is large. More generally the naive Bayes assumption is that the attributes (the components of the random vector $\mathbf{x}$) are conditionally independent, conditioned on the class, i.e. if

$\mathbf{x} = \begin{bmatrix} x_1 & \cdots & x_D \end{bmatrix}^T$ then

$$p(\mathbf{x}|\mathcal{C}) = \prod_{n=1}^{D} p(x_n|\mathcal{C}).$$

EXERCISE 130. Show that the conditional independence assumption leads to a diagonal covariance matrix if one assumes a Gaussian class-conditional pdf.

Using Bayes' theorem we write

$$\begin{aligned} P(\mathcal{C}_k|\mathbf{x}) &= \frac{P(\mathcal{C}_k)p(\mathbf{x}|\mathcal{C}_k)}{p(\mathbf{x})} \\ &= \frac{P(\mathcal{C}_k)\prod_n p(x_n|\mathcal{C}_k)}{\sum_k P(\mathcal{C}_k)\prod_n p(x_n|\mathcal{C}_k)}. \end{aligned}$$

Once the posterior probabilities are known one then proceed to build an appropriate classifier, i.e. a rule that assigns a given observation $\mathbf{x}$ to a specific class. The most naive classifier is given by

$$\mathcal{C} = \arg\max_{\mathcal{C}} \frac{P(\mathcal{C})\prod_n p(x_n|\mathcal{C})}{\sum_k P(\mathcal{C}_k)\prod_n p(x_n|\mathcal{C}_k)}$$

which simplifies to (since the denominator does not depend on the class),

$$\mathcal{C} = \arg\max_{\mathcal{C}} P(\mathcal{C})\prod_n p(x_n|\mathcal{C}).$$

EXERCISE 131. Show, using maximum likelihood, and assuming Gaussian pdf's that

$$\begin{aligned} P(\mathcal{C}_k) &= \frac{N_k}{N} \\ \mu_{nk} &= \frac{1}{N_k}\sum x_{nk} \\ \sigma^2_{nk} &= \frac{1}{N_k}\sum (x_{nk}-\mu_{nk})^2,\ n=1,\ldots,D;\ k=1,\ldots,K \end{aligned}$$

where N_k is the number of samples in class $\mathcal{C}_k$ and the sums go over all the samples x_{nk} that belong to class $\mathcal{C}_k$.

24.4. Probabilistic Discriminative Models.

24.4.1. Logistic Regression. Recall from (24.1) that for the two-class problem the posterior probability is given by a logistic sigmoid function,

$$P(\mathcal{C}_1|\mathbf{x}) = \frac{1}{1+\exp(-a(\mathbf{x}))} = \sigma(a(\mathbf{x})) \tag{24.1}$$

where

$$a(\mathbf{x}) = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)P(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)P(\mathcal{C}_2)}.$$

Recall that, assuming Gaussian densities with a shared covariance, we find that

$$P(\mathcal{C}_1|\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x} + w_0), \tag{24.2}$$

where the parameters are given by (24.3) and (24.4). In this case $a(\mathbf{x})$ is a linear function in $\mathbf{x}$, with the parameters derived from using Gaussian class conditionals. Upon closer inspection, it should be clear that, given $\mathbf{w}$, (24.1) and (24.2) directly map the input variable $\mathbf{x}$ to the posterior class probability $P(\mathcal{C}|\mathbf{x})$. There is no compelling reason for Gaussian class conditionals, or any class conditional for that matter. We could equally well assume a linear expression $a(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$ and infer the parameters directly from the training data. In fact there is no reason to assume a linear relationship for $a(\mathbf{x})$ and we may assume the more general form,

$$P(\mathcal{C}_1|\mathbf{x}, \mathbf{w}) = \sigma(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x})) \tag{24.3}$$

with $P(\mathcal{C}_2|\mathbf{x}, \mathbf{w}) = 1 - P(\mathcal{C}_1|\mathbf{x}, \mathbf{w})$, and where the conditioning on the parameters $\mathbf{w}$ is stated explicitly. The parameters $\mathbf{w}$ of the logistic regression problem can be determined using maximum likelihood. Given a data set $\{\boldsymbol{\phi}_n, t_n\}$, $t_n \in \{0, 1\}$ indicates the class that $\mathbf{x}_n$ belongs to, and $\boldsymbol{\phi}_n = \boldsymbol{\phi}(\mathbf{x}_n)$, $n = 1, \ldots, N$. The joint distribution is given by

$$\begin{aligned} p(X, \mathbf{t}|\mathbf{w}) &= P(\mathbf{t}|X, \mathbf{w})p(X) \\ &= p(X)\prod_n P(t_n|\mathbf{x}_n, \mathbf{w}), \end{aligned} \tag{24.4}$$

where we assume that t_n is conditionally independent of the rest of the data, given $\mathbf{x}_n$. Recall that X consists of data that belongs to the two classes, $t = 1$ or $t = 0$

for classes $\mathcal{C}_1$ and $\mathcal{C}_2$ respectively. This means that t_n satisfies a Bernoulli distribution $P(t_n|\mathbf{x}_n, \mathbf{w}) = P(\mathcal{C}_1|\mathbf{x}_n, \mathbf{w})^{t_n} \left(1 - P(\mathcal{C}_1|\mathbf{x}_n, \mathbf{w})^{1-t_n}\right.$, where $P(\mathcal{C}_1|\mathbf{x}_n, \mathbf{w})$ is given by (24.3). Clearly, $P(t_n = 1|\mathbf{x}_n, \mathbf{w}) = P(\mathcal{C}_1|\mathbf{x}_n, \mathbf{w})$ is the probability that data point $\mathbf{x}_n$ belongs to $\mathcal{C}_1$. The likelihood therefore becomes,

$$p(X, \mathbf{t}|\mathbf{w}) = p(X) \prod_n P(\mathcal{C}_1|\mathbf{x}_n, \mathbf{w})^{t_n} \left(1 - P(\mathcal{C}_1|\mathbf{x}_n, \mathbf{w})\right)^{1-t_n}. \tag{24.5}$$

Making use of (24.3), we find that

$$\begin{aligned} p(X, \mathbf{t}|\mathbf{w}) &= p(X) \prod_n \sigma(\mathbf{w}^T \boldsymbol{\phi}_n)^{t_n} \left(1 - \sigma(\mathbf{w}^T \boldsymbol{\phi}_n)\right)^{1-t_n} \\ &= p(X) \prod_{n=1}^{N} \sigma(\mathbf{w}^T \boldsymbol{\phi}_n)^{t_n} \left(1 - \sigma(\mathbf{w}^T \boldsymbol{\phi}_n)\right)^{1-t_n}. \end{aligned}$$

Using the negative log-likelihood as an error function we get

$$E(\mathbf{w}) = -\ln p(X, \mathbf{t}|\mathbf{w}) = -\sum_{n=1}^{N} \left\{ t_n \ln \sigma(\mathbf{w}^T \boldsymbol{\phi}_n) + (1 - t_n) \ln \left(1 - \sigma(\mathbf{w}^T \boldsymbol{\phi}_n)\right) \right\} - \ln p(X). \tag{24.6}$$

The parameters are obtained by minimizing the error term. The gradient of the error with respect to $\mathbf{w}$ is

$$\boldsymbol{\nabla} E(\mathbf{w}) = \sum_{n=1}^{N} \left(\sigma(\mathbf{w}^T \boldsymbol{\phi}_n) - t_n\right) \boldsymbol{\phi}_n,$$

which is easy to derive by noting that

$$\frac{d\sigma}{da} = \sigma(1 - \sigma).$$

Unlike previous situations where the equations decouple, setting $\boldsymbol{\nabla} E(\mathbf{w}) = 0 = \sum_{n=1}^{N} \left(\sigma(\mathbf{w}^T \boldsymbol{\phi}_n) - t_n\right) \boldsymbol{\phi}_n$ results in a nonlinear system of equations in $\mathbf{w}$ that has to be solved numerically. Since the gradient is readily available, a gradient-based optimization method can be applied directly to (24.6).

It may not be obvious from the discussion above, but it is important to realize that samples of both classes are needed. Both classes 'see' samples belonging to itself, as well as samples belonging to the other class. These are needed in order to find maximal class separation within the parameter space.

24.4.2. Multiclass logistic regression. For K classes we need K models

$$\begin{aligned} P(\mathcal{C}_k|\mathbf{x}, \mathbf{w}_k) &= \frac{\exp\left(\mathbf{w}_k^T \boldsymbol{\phi}(\mathbf{x})\right)}{\sum_{j=1}^K \exp\left(\mathbf{w}_j^T \boldsymbol{\phi}(\mathbf{x})\right)} \\ &= \frac{\exp\left(a_k(\mathbf{x})\right)}{\sum_{j=1}^K \exp\left(a_j(\mathbf{x})\right)},\ k = 1, \ldots, K, \end{aligned}$$

where $a_k(\mathbf{x}) = \mathbf{w}_k^T \boldsymbol{\phi}(\mathbf{x})$. Given data $\mathbf{x}_n$, $\mathbf{t}_n$, we use a 1-of-K coding scheme, i.e. the k-th element of $\mathbf{t}_n$, $t_{kn} = 1$ if $\mathbf{x}_n$ belongs to $\mathcal{C}_k$ with the rest of the elements, $t_{jn} = 0,\ \ j \neq k$. We derive a suitable likelihood function from the joint distribution

$$p(X, T|\mathbf{w}) = p(X) \prod_n P(\mathbf{t}_n|\mathbf{x}_n, \mathbf{w}),$$

again assuming that $\mathbf{t}_n$ is conditionally independent of the rest of the data points, given $\mathbf{x}_n$. We also need an expression for the distribution $P(\mathbf{t}_n|\mathbf{x}_n, \mathbf{w})$, given by the generalized Bernoulli distribution,

$$P(\mathbf{t}_n|\mathbf{x}_n, \mathbf{w}) = \prod_k P(\mathcal{C}_k|\mathbf{x}_n, \mathbf{w})^{t_{nk}}.$$

It therefore follows that

$$p(X, T|\mathbf{w}) = p(X) \prod_n \prod_k P(\mathcal{C}_k|\mathbf{x}_n, \mathbf{w})^{t_{nk}}.$$

The log-likelihood is given by

$$\begin{aligned} \ell(\mathbf{w}) &= \ln p(X) + \sum_n \sum_k t_{kn} \ln P(\mathcal{C}_k|\mathbf{x}_n, \mathbf{w}_k) \\ &= \ln p(X) + \sum_n \sum_k t_{kn} \left[a_k(\mathbf{x}_n) - \ln \left(\sum_{j=1}^K \exp a_j(\mathbf{x}_n) \right) \right], \end{aligned}$$

and we need to find $\mathbf{w}^* = \arg\max_{\mathbf{w}} \ell(\mathbf{w})$, or $\nabla_{\mathbf{w}} \ell(\mathbf{w}^*) = 0$. Since

$$\nabla_{\mathbf{w}_k} \ell(\mathbf{w}) = \sum_n \left[t_{nk} - \frac{\exp \mathbf{w}_k^T \boldsymbol{\phi}(\mathbf{x}_n)}{\sum_{j=1}^K \exp \mathbf{w}_k^T \boldsymbol{\phi}(\mathbf{x}_n)} \right] \boldsymbol{\phi}(\mathbf{x}_n),\ k = 1, \ldots, K,$$

there are no analytical solutions of $\nabla_w \ell(\mathbf{w}) = 0$, and again one has to resort to numerical optimization techniques.

24.4.3. Full Bayesian Approach. In a Bayesian approach we need a prior for the parameters, let us assume the simple form

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \lambda I).$$

The posterior distribution over $\mathbf{w}$ (using Bayes' theorem) is given by

$$p(\mathbf{w}|\mathbf{t}) \propto p(\mathbf{w})P(\mathbf{t}|\mathbf{w})$$

where we recall that

$$P(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^{N} \sigma(\mathbf{w}^T\boldsymbol{\phi}_n)^{t_n} \left(1 - \sigma(\mathbf{w}^T\boldsymbol{\phi}_n)\right)^{1-t_n}$$

so that

$$\begin{aligned} \ln p(\mathbf{w}|\mathbf{t}) &= -\frac{1}{2\lambda}\mathbf{w}^T\mathbf{w} \\ &\quad + \sum_{n=1}^{N} \left\{ t_n \ln \sigma(\mathbf{w}^T\boldsymbol{\phi}_n) + (1-t_n) \ln \left(1 - \sigma(\mathbf{w}^T\boldsymbol{\phi}_n)\right) \right\} + \text{const.} \end{aligned}$$

One can now follow a full Bayesian approach and marginalize over the parameters to find the predictive distribution directly

$$P(\mathcal{C}_1|\mathbf{x}, \mathbf{t}) = \int P(\mathcal{C}_1|\mathbf{x}, \mathbf{w}) p(\mathbf{w}|\mathbf{t}) d\mathbf{w},$$

assuming that the probability of class $\mathcal{C}_1$ becomes independent of the data $\mathbf{t}$ once the parameters $\mathbf{w}$ are known. This becomes cumbersome to evaluate and common sence indicates a point estimate of the parameters using the posterior. Thus one minimizes the negative log posterior,

$$-\ln p(\mathbf{w}|\mathbf{t}) = -\sum_{n=1}^{N} \{t_n \ln y_n + (1-t_n) \ln(1-y_n)\} + \frac{1}{2}\lambda \mathbf{w}^T\mathbf{w} + \text{const.}$$

Comparison with (24.6) shows that the prior introduces a penalty term. In practice this is essential in order to prevent overfitting.

EXERCISE 132. Consider two classes $\mathcal{C}_1$ and $\mathcal{C}_2$ with class conditional densities given by $p(x|\mathcal{C}_1) = \mathcal{N}(x|\mu_1 = -0.3, \sigma_1^2 = 0.8)$ and $p(x|\mathcal{C}_2) = \mathcal{N}(x|\mu_2 = 1.1, \sigma_2^2 = 0.8)$.

The prior probabilities are the same, $P(\mathcal{C}_1) = P(\mathcal{C}_2) = \frac{1}{2}$. Plot both class-conditional densities on the same axes.

Suppose you want to classify two observations, $x_1 = 2.0$ and $x_2 = 0.2$. Calculate the likelihood $p(x|\mathcal{C}_2)$ for both data points. You should observe that both data points have the same likelihood. Can you conclude from this that it is equally likely that the two points belong to class $\mathcal{C}_2$? Plot the two data points on the same axes. What does it tell you?

Calculate the likelihood ratios $\frac{p(x|\mathcal{C}_2)}{p(x|\mathcal{C}_1)}$ for both observed values. Compare this with the posterior probabilities $P(\mathcal{C}_1|x)$ and $P(\mathcal{C}_2|x)$ for the two observations. Is there any doubt about the classification of x_1?

Finally plot the two posterior probabilities $P(\mathcal{C}_1|x)$ and $P(\mathcal{C}_2|x)$ as functions of x, on the same axis as the class-conditional densities. What does it tell you about the classification of observations?

CHAPTER 25

PRINCIPAL COMPONENT ANALYSIS

25.1. Introduction.

We have already studied a dimension reduction problem, namely LDA where the goal is to project the data to a lower dimensional space in such a way that maximal class separation is achieved in the projected space. This is not always the most appropriate projection and in this chapter we study an alternative approach. In brief it can be characterized as identifying a lower dimensional subspace of a given space. In order to explain the general idea, we consider a practical application.

In facial recognition we are interested in facial images, and only facial images. Any image is represented as an $m \times n$ matrix where every entry in the matrix (every pixel) represents a shade of gray (the dimension increases three-fold for color images). It should be clear that an $m \times n$ gray-scale images is a specific point in an $m \times n$-dimensional vector space. This can be very high even for low resolution images. Since we are interested in only facial images it might just be possible that the facial images occupy lower dimensional subspace of the full $m \times n$-dimensional images space.

In general then, in this chapter we are interested in finding lower dimensional subspaces that describe the essential properties of our data. This allows one to project the data onto these lower dimensional subspaces, sometimes leading to significant dimensionality reductions. As before we need to 'learn' this subspace from representative examples.

Since principle components are closely related to the SVD the reader may find it useful to read this chapter together with Section 11.5 on the SVD.

Principle component analysis is a useful technique for identifying *linear* subspaces. Although powerful methods have recently become available to identify *non-linear* subspaces, these remain outside the scope of this discussion.

Add a paragrap
on diffusio
maps?

25.2. Principal Components .

Given N observations $\mathbf{x}_n,\ n = 1, \ldots, N$ of dimension D, we want to find the directions of maximum variation in the data. To start, suppose we project the data onto a one dimensional subspace defined by $\mathbf{u}_1$, where $\mathbf{u}_1^T\mathbf{u}_1 = 1$. Thus $\mathbf{x}_n$ is projected onto the vector $\left(\mathbf{u}_1^T\mathbf{x}_n\right)\mathbf{u}_1$, and in this coordinate system, the projected value is just given by $\mathbf{u}_1^T\mathbf{x}_n$. The idea is to choose $\mathbf{u}_1$ in such a way that the variance of the projected values is a large as possible. If the sample mean is given by

$$\overline{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n$$

then the mean of the projected data is $\left(\mathbf{u}_1^T\overline{\mathbf{x}}\right)$ and the variance of the projected data is

$$\begin{aligned}\frac{1}{N}\sum_{n=1}^{N}\left(\mathbf{u}_1^T\mathbf{x}_n - \mathbf{u}_1^T\overline{\mathbf{x}}\right)^2 &= \frac{1}{N}\sum_{n=1}^{N}\mathbf{u}_1^T(\mathbf{x}_n - \overline{\mathbf{x}})(\mathbf{x}_n - \overline{\mathbf{x}})^T\mathbf{u}_1 \\ &= \mathbf{u}_1^T S\mathbf{u}_1\end{aligned}$$

where S is the sample covariance. Introducing a Lagrange multiplier, we maximize

$$\mathbf{u}_1^T S\mathbf{u}_1 + \lambda(\mathbf{u}_1^T\mathbf{u}_1 - 1),$$

subject to $\mathbf{u}_1^T\mathbf{u}_1 = 1$. This gives the eigenvalue problem

$$S\mathbf{u}_1 = \lambda\mathbf{u}_1,$$

and using the constraint gives

$$\lambda = \mathbf{u}_1^T S\mathbf{u}_1.$$

The projected variance is therefore equal to the eigenvalue λ and attains its maximum if we choose the largest eigenvalue, with $\mathbf{u}_1$ the corresponding eigenvector. This is our first principal vector.

The rest of the principal vectors are obtained by each time projecting onto a vector orthogonal the all the previous principal directions. This means that the principal directions are the eigenvectors of the sample covariance matrix belonging to the largest eigenvalues, and the eigenvalues are the variances in those directions.

EXERCISE 133. Show that the eigenvalues of the sample covariance matrix are all non-negative; a fact that is important for our discussion above.

25.2.1. Dimensionality reduction. Suppose that the N, D-dimensional observations are drawn from a multivariate Gaussian distribution. If $\overline{\mathbf{x}}$ and S are the sample mean– and covariance respectively, then the Gaussian distribution is estimated to be (repeated here for convenience),

$$\mathcal{N}(\mathbf{x}|\overline{\mathbf{x}}, S) = \frac{1}{|2\pi S|^{1/2}} \exp\left(-\frac{1}{2}\left(\mathbf{x}-\overline{\mathbf{x}}\right)^T S^{-1}\left(\mathbf{x}-\overline{\mathbf{x}}\right)\right). \tag{25.1}$$

We clearly assume that the covariance matrix is non-singular. But this is not a given. In fact, in practice the covariance matrix has often zero, or very small eigenvalues and this problem should somehow be addressed. From the discussion above, zero eigenvalues tell us something about the data. It tells us that that particular dimension is void of any data—it is an 'empty' dimension that can be removed. PCA is very good at telling us exactly which dimensions, in a linear sense, can be removed—the ones with zero or negligible small variance as measured by the eigenvalues. Let us therefore attempt a transformation of the data of the form

$$\mathbf{y} = W^T\left(\mathbf{x}-\overline{\mathbf{x}}\right). \tag{25.2}$$

The quadratic form in the Gaussian distribution therefore transforms as,

$$\mathbf{y}^T W^{-1} S^{-1} W^{-T} \mathbf{y},$$

and the covariance matrix associated with the transformed data $\mathbf{y}$ is given by,

$$S_y = W^T S W.$$

If the transformation W is chosen as the eigenvector matrix Q of S then the transformed covariance matrix is given by $S_y = \Lambda$, where Λ is the diagonal eigenvalue matrix. Since the eigenvalues of a covariance matrix is non-negative we assume without loss of generality that the eigenvalues are conveniently ordered from large to small. If the $\text{rank}(S_y) = r$, then we can write

$$\widehat{Q}^T S \widehat{Q} = \Lambda_+, \tag{25.3}$$

where $\widehat{Q}$ consists of the first r eigenvectors of S_y, i.e. the first r columns of Q, and Λ_+ is an $r \times r$ diagonal matrix consisting of the nonzero eigenvalues of S.

If we now transform our original data as

$$\mathbf{y} = \widehat{Q}^T(\mathbf{x} - \overline{\mathbf{x}}),$$

not only do we not loose any information, but we have achieved a dimensionality reduction, from D to r dimensions by squeezing out the empty dimensions. Of course one can also get rid of those dimensions with little variance as measure by the eigenvalues.

25.2.2. The whitening transformation. We need not stop with the dimensionality reduction given by (25.3), indeed it is often convenient to transform the data so that it is spherically distributed. Since this amounts to the identity covariance matrix, the corresponding transformation follows immediately from (25.3),

$$\Lambda_+^{-1/2}\widehat{Q}^T S \widehat{Q}\Lambda_+^{-1/2} = I_r,$$

where I_r is the $r \times r$ identity matrix. Thus the whitening transformation of the data is given by

$$\mathbf{y} = \left(\widehat{Q}\Lambda_+^{-1/2}\right)^T (\mathbf{x} - \overline{\mathbf{x}})\,. \tag{25.4}$$

25.3. Numerical Calculation.

It is not a good idea to calculate the eigenvalues of the sample covariance matrix numerically since this procedure is unstable. More precisely, by calculating the covariance matrix we loose roughly half the available significant digits. This can be disastrous. Instead we therefore form the matrix

$$X = \frac{1}{\sqrt{N}}\begin{bmatrix} \mathbf{x}_1 - \overline{\mathbf{x}} & \cdots & \mathbf{x}_N - \overline{\mathbf{x}} \end{bmatrix}$$

and calculate its SVD, $X = U\Sigma V^T$.

As explained in Section 11.5, the rank r of X is the number of nonzero singular values. The first r columns of U therefore form an orthogonal basis for the column space of X. The singular values are the standard deviations along these principal directions. It therefore measures to what extent a particular principal direction is

'occupied' by the data. A zero singular value implies an empty dimension without data representation. Projecting our data onto the first r columns of U therefore entails no loss in information. It squeezes out the 'empty' dimensions. In practice a system often benefits from also squeezing the directions with little data representation, i.e. the directions associated with small singular values. A simple example may be helpful. Suppose we know that our 2D data is supposed to fall on a straight line. If we calculate the principal components of this data, we certainly expect that the first principle direction should point in the direction of the line (can you see why it is important to remove the mean?). If life were perfect the second singular value should be zero. But of course there are small measurement and possibly other errors. This means that the data does not fall exactly on the line, with the result that the second singular value measures the extent of the noise—the deviation from the straight line. The fact that the second singular value is small, is probably an indication that we are dealing with noise in which case it is perfectly reasonable to project the data back onto the space (line) defined by the first principal component. Note that this amounts to calculating the linear least squares approximation.

Let us now consider the higher dimensional case. The singular values tell us which of the directions we can safely squeeze out. Let us say for our application only the first M singular values are significant. We then project the data onto the M-dimensional subspace defined by the orthogonal basis consisting of the first M columns of U. Let us call this matrix U_M. Given a data value $\mathbf{x}$ its projection onto U_M is given by $\mathbf{z}$ where $\mathbf{x} - \boldsymbol{\mu} = U_M\mathbf{z}$ in a least squares sense. Since U_M has orthogonal columns, the least squares solution gives us the projection,

$$\mathbf{z} = U_M^T(\mathbf{x} - \boldsymbol{\mu}),$$

where $\boldsymbol{\mu}$ is the mean.

25.4. Probabilistic PCA.

PCA, as discussed in the previous sections have little, if any, in terms of a probabilistic description. We now discuss a probabilistic alternative.

Suppose $\mathbf{x}$ is the D-dimensional observed vector and $\mathbf{z}$ a corresponding M-dimensional latent vector with

$$\mathbf{x} = W\mathbf{z} + \boldsymbol{\mu} + \boldsymbol{\epsilon}, \tag{25.1}$$

where the noise term $\boldsymbol{\epsilon}$ is a D-dimensional Gaussian variable with zero and covariance $\sigma^2 I$. Note that (25.1) means that $\mathbf{x} - \boldsymbol{\mu} \in \text{col } W$ and the $\mathbf{z}$ are the coordinates. If we also assume that $\mathbf{z}$ is a Gaussian variable with $p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\mathbf{0}, I)$, then $p(\mathbf{x}|\mathbf{z})$ is also a Gaussian variable given by

$$p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}|W\mathbf{z} + \boldsymbol{\mu}, \sigma^2 I).$$

The marginal distribution $p(\mathbf{x})$ is given by

$$\begin{aligned} p(\mathbf{x}) &= \int p(\mathbf{x}, \mathbf{z}) d\mathbf{z} \\ &= \int p(\mathbf{x}|\mathbf{z}) p(\mathbf{z}) d\mathbf{z}. \end{aligned}$$

Recall that this is also a Gaussian variable

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, C)$$

where the covariance is given by

$$C = WW^T + \sigma^2 I.$$

Note:

(1) This can also be obtained from a direct calculation using (25.1) .
(2) There is considerable redundancy in this formulation. Replacing W with WR, where R is an orthogonal matrix $RR^T = I = R^T R$, leaves the covariance, hence the distribution of $\mathbf{x}$ unchanged.

We need C^{-1}, and a useful formula is

$$C^{-1} = \sigma^{-2} I - \sigma^{-2} W M^{-1} W^T$$

where the $M \times M$ matrix M is given by

$$M = W^T W + \sigma^2 I.$$

The posterior $p(\mathbf{z}|\mathbf{x})$ is also a Gaussian distribution and is given by

$$p(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}|M^{-1}W^T(\mathbf{x}-\boldsymbol{\mu}), \sigma^{-2}M).$$

We have not made any estimates yet, but note that when W, $\boldsymbol{\mu}$, and σ are known,

$$\mathbf{z} = M^{-1}W^T(\mathbf{x}-\boldsymbol{\mu})$$

gives us the projection of $\mathbf{x}$ onto its principal coordinates.

The various unknowns, W, $\boldsymbol{\mu}$, and σ, are estimated using maximum likelihood. Assuming data $X = \{\mathbf{x}_n,\ n = 1, \ldots, N\}$ the data log likelihood is given by

$$\begin{aligned} \ln p(X|W, \boldsymbol{\mu}, \sigma) &= \sum_{n=1}^{N} p(\mathbf{x}_n|W, \boldsymbol{\mu}, \sigma) \\ &= -\frac{1}{2}ND\ln 2\pi - \frac{N}{2}\ln|C| - \frac{1}{2}\sum_{n=1}^{N}(\mathbf{x}_n-\boldsymbol{\mu})^T C^{-1}(\mathbf{x}_n-\boldsymbol{\mu}) \end{aligned}$$

where $C = WW^T + \sigma^2 I$. Maximizing the log likelihood with respect to $\boldsymbol{\mu}$ is easy and we find that

$$\boldsymbol{\mu} = \overline{\mathbf{x}},$$

the sample mean. Maximizing with respect to W and σ is not easy and we quote Bishop,

$$W_{ML} = U_M(L_M - \sigma^2_{ML}I)^{1/2}R$$

where U_M consists of the D eigenvectors of the sample covariance matrix S belonging to the D largest eigenvalues, i.e. these are exactly the same as the principal directions of standard PCA. L_M is a diagonal matrix with the corresponding eigenvalues on its diagonal, and R is an arbitrary orthogonal matrix (for simplicity choose $R = I$). Finally,

$$\sigma^2_{ML} = \frac{1}{D-M}\sum_{i=M+1}^{D}\lambda_i,$$

i.e. σ^2_{ML} is the average variance associated with the discarded dimensions.

The important question is, what do we buy with probabilistic PCA? Let us give one answer. The PCA subspace is still large and contains elements that can be far removed from the features we are interested in. In facial recognition for example, the

PCA 'face space' contains some decidedly non-facial images. Without a probabilistic representation it is hard to decide whether a feature in the PCA space is actually close to the training set. Once we have equipped the feature space with a probabilistic description we have a systematic way of deciding whether a feature in the PCA space is close to the training set, and therefore something we might be interested in.

CHAPTER 26

PARTIALLY OBSERVED DATA AND THE EM ALGORITHM

26.1. Introduction.

In our discussion of classification in Chapter~24 we assumed fully observed data, i.e. each observation came with a class label. We are now ready to start discussing partially observed data. If for example, the data does not come with class labels, the class labels need to be inferred from the observed data. In general, therefore, the task is to infer missing data from the observed data. Throughout our discussion the fundamental assumption is that, should the missing data somehow become available, the training can be done with ease. A general tool for dealing with partially observed data is the Expectation Maximization (EM) algorithm. The basic idea is very simple. Since we can proceed with the training (estimation of the parameter values) for the fully observed data, we estimate the missing values by calculating an expectation based on the current estimate of the parameters. Once we have values for the missing data, we proceed to maximize a likelihood to get an updated estimate for the parameters. These are then used to re-estimate the missing data, etc.

The simplest example of the EM algorithm is K-means clustering, the starting point of our discussion.

26.2. K-Means Clustering.

We have N observations, each observation belonging to one of K clusters, but we don't know which one. Our task is to assign each observation to an appropriate cluster. As soon as we have done that, the usual classification techniques apply.

First we introduce some notation. With each D-dimensional observation $\mathbf{x}_n$ we associate a cluster indicator $\mathbf{t}_n = \begin{bmatrix} t_{n1} & \cdots & t_{nK} \end{bmatrix}^T$ where $t_{nk} = 1$ if $\mathbf{x}_n$ belongs to cluster k, otherwise $t_{nk} = 0$. In the classification problem of Chapter 24 the $\mathbf{t}_n$ are

known, in the clustering problem they are unknown and need to be inferred from the data. Given K, D-dimensional vectors $\boldsymbol{\mu}_k$, which we think of as cluster centers, we define an objective function as

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \left\| \mathbf{x}_n - \boldsymbol{\mu}_k \right\|^2 .$$

Our goal becomes to find the t_{nk} and $\boldsymbol{\mu}_k$ so as to minimize J. This proceeds in two steps. First we choose initial values for the $\boldsymbol{\mu}_k$. Keeping these fixed we minimize J with respect to t_{nk}. In the second step we keep the t_{nk} fixed and minimize J with respect to the $\boldsymbol{\mu}_k$. These steps are repeated until convergence.

The t_{nk} are easily determined. For each data point $\mathbf{x}_n$ assign it to the cluster for which $\left\| \mathbf{x}_n - \boldsymbol{\mu}_k \right\|^2$ is the smallest. This means that each data point is assigned to its nearest cluster, as defined by its estimated cluster center $\boldsymbol{\mu}_k$.

Setting the derivative of J with respect to $\boldsymbol{\mu}_k$ equal to zero, we get

$$\boldsymbol{\mu}_k = \frac{\sum_n t_{nk} \mathbf{x}_n}{\sum_n t_{nk}}.$$

This means that we set $\boldsymbol{\mu}_k$ equal to the sample mean of all the points assigned to cluster k.

Note:

(1) It is common practice to rescale the data before the clustering algorithm is applied, by, for example, whitening the data. This ensures that the data is normalized so that the distance measures have some objective meaning. Note however, the warning given by Duda and Hart that by doing this one runs the risk of rescaling distances when they are actually indicative of between-cluster separation.

(2) The K-means algorithm uses a *hard* assignment of each data point to a specific class. This does not take into account that at least for some points there may be considerable ambiguity as to which class they belong. In such cases it may be better to use a *soft* assignment where data points are assigned to clusters in such a way as to reflect the uncertainty.

(3) One has to initialize the K-means algorithm by assigning initial clusters. This choice is important because it determines the final clustering, i.e. the

algorithm finds a local optimum determined by the initial selection. One possibility is to select K random samples and use these as initial cluster means. This does not work well in practice since the choice of initial clusters may be heavily biased. In our experience a binary split procedure works better. It starts by selecting two initial cluster means at random. The data points are then assigned to these clusters in the normal way. The scatter of the two clusters is then computed and the cluster with the largest scatter is split by choosing two samples from it at random. The algorithm is now iterated a few times, starting with these three initial cluster means. Then the cluster with the largest scatter is split again, and the procedure is repeated until the desired number of clusters is obtained. The algorithm is then iterated until convergence, i.e. until the cluster means remain within a pres.

26.3. Gaussian Mixture Models.

We have seen that Gaussian pdf's have particularly useful analytical properties. In practice however, one often encounters situations where the data cannot be accurately represented by a single Gaussian pdf. This is certainly the case for all multi-modal data sets. One possibility is to represent the data by a mixture of K Gaussians,

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \Sigma_k). \tag{26.1}$$

The parameters to estimate are the class mean and covariance, $\boldsymbol{\mu}_k$ and Σ_k respectively, as well as the 'class probabilities' or mixture coefficients, π_k. All of these need to be estimated from the data, $\mathbf{x}_n,\ n = 1, \ldots, N$.

If we knew the mixture components to which each each data point belong, the learning problem is easy—we simply do sample estimates of the mean and covariances from the data belonging to each mixture component. The fraction of the data points belonging to each mixture then gives the mixture coefficients. Since we don't know it, we need to proceed in a more principled way. Accordingly, we introduce a latent random variable $\mathbf{z} = \begin{bmatrix} z_1 & \cdots & z_K \end{bmatrix}^T$ where $z_k \in \{0, 1\}$ and $\sum_k z_k = 1$. This means

that $\mathbf{z}$ can be in one of K states, according to which element is non-zero. Below we construct the joint distribution $p(\mathbf{x}, \mathbf{z})$ from the marginal $P(\mathbf{z})$ and the conditional distribution $p(\mathbf{x}|\mathbf{z})$, i.e. we use $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}|\mathbf{z})P(\mathbf{z})$. The marginal contribution over $\mathbf{z}$ is given in terms of the mixture coefficients,

$$P(z_k = 1) = \pi_k,$$

where $0 < \pi_k < 1$ and $\sum_k \pi_k = 1$. This means that $P(\mathbf{z})$ is written as a generalized Bernoulli probability distribution,

$$P(\mathbf{z}) = \prod_{k=1}^{K} \pi_k^{z_k}.$$

The conditional distribution $p(\mathbf{x}|z_k = 1)$ is just the k-th mixture component

$$p(\mathbf{x}|z_k = 1) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \Sigma_k),$$

which is rewritten as

$$p(\mathbf{x}|\mathbf{z}) = \prod_{k=1}^{K} \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \Sigma_k)^{z_k}.$$

The joint distribution is given by $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}|\mathbf{z})P(\mathbf{z})$ and the marginal distribution $p(\mathbf{x})$ becomes

$$\begin{aligned} p(\mathbf{x}) &= \sum_{\mathbf{z}} p(\mathbf{x}|\mathbf{z})P(\mathbf{z}) \\ &= \sum_{\mathbf{z}} \prod_{k=1}^{K} \pi_k^{z_k} \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \Sigma_k)^{z_k} \\ &= \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \Sigma_k). \end{aligned} \tag{26.2}$$

Thus the marginal distribution is a mixture model of the form (26.1).

We now compute the conditional probability $P(\mathbf{z}|\mathbf{x})$. Defining $\gamma(z_k) = P(z_k = 1|\mathbf{x})$, it follows from Bayes' theorem that

$$\begin{aligned}\gamma(z_k) &= \frac{P(z_k = 1)p(\mathbf{x}|z_k = 1)}{\sum_{j=1}^{K} P(z_j = 1)p(\mathbf{x}|z_j = 1)} \\ &= \frac{\pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \Sigma_j)}.\end{aligned}$$

Note that this is a soft assignment where $\mathbf{x}$ is not assigned to a specific class, but its probability of belonging to a class is distributed over all the classes. We view π_k as the prior probability of the k-th mixture component, $z_k = 1$ i.e. $P(z_k = 1) = \pi_k$, and $\gamma(z_k)$ as the posterior probability once we have observed $\mathbf{x}$. Alternatively, $\gamma(z_k)$ can be viewed as the *responsibility* that the k-th component takes for 'explaining' the observation $\mathbf{x}$. Said in another way, $\gamma(z_k)$ is the probability of $\mathbf{x}$ belonging to the k-th mixture component.

In summary, provided we know the mixture parameters, π_k, μ_k and Σ_k the responsibilities allow one to make soft assignments of data to the different components. In the next section we'll see, given the responsibilities, how to estimate the mixture parameters.

EXAMPLE 134. Use *ancestral sampling* to generate samples from the joint probability using three mixture components gives Figure 26.3.1. In (a) the samples as drawn from the different components are shown. (This corresponds to fully observed data where each data point is assigned a 'cluster'.) In (b) the data is shown without its mixture components; this is the way it is presented to us. In (c) the responsibility that each component takes for every data point is indicated with different color shades.

Suppose we are given N, D-dimensional data points $\mathbf{x}_n$ collected into an $N \times D$ matrix X. The likelihood is defined as the joint distribution $p(X, \boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma)$. It might be useful to note that we write down the joint distribution of the observed data, and the parameters to be estimated. The latent variables enter through the $\boldsymbol{\pi}$ parameters.

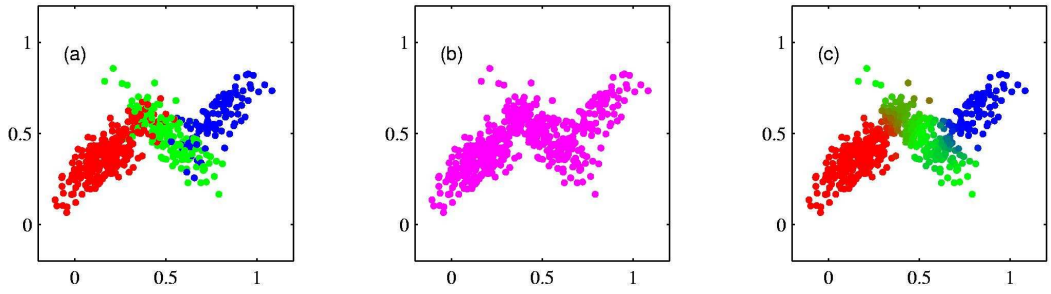


FIGURE 26.3.1. Generating samples from a mixture of 3 Gaussians.

The likelihood is therefore given by

$$\begin{aligned} p(X, \boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma) &= p(X|\boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma) p(\boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma) \\ &= p(\pi, \boldsymbol{\mu}, \Sigma) \prod_{n=1}^{N} p(\mathbf{x}_n|\boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma) \\ &= p(\pi, \boldsymbol{\mu}, \Sigma) \prod_{n=1}^{N} \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \Sigma_k) \end{aligned}$$

where we have used (26.2) in the last step. The log likelihood is therefore given by

$$\ln p(X, \boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma) = \ln p(\boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma) + \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \Sigma_k) \right\}.$$

If we do not want to provide prior probabilities $p(\boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma)$ over the parameters (or maximizing over this more complex expression), we might consider

$$\ln p(X|\boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \Sigma_k) \right\}. \tag{26.3}$$

It is important to note that for mixture models the likelihood can actually become infinite, i.e. it has a singularity, a situation that should be avoided. To see how this can happen, consider a mixture model where all the components have covariances of the form $\sigma_k^2 I$. Suppose one of the mixture components has its mean, say $\boldsymbol{\mu}_j$, exactly equal to one of the data points, i.e. $\boldsymbol{\mu}_j = \mathbf{x}_n$. The contribution of this data point to

the likelihood is of the form

$$\mathcal{N}(\mathbf{x}_n|\mathbf{x}_n, \sigma_j^2 I) = \frac{1}{\sqrt{2\pi\sigma_j^2}}.$$

This goes to infinity as $\sigma_j \to 0$, i.e. this particular component collapses onto a specific data point, sending the likelihood to infinity—we are in the process of maximizing the likelihood after all. This is always a possibility and one has to take steps in order to avoid this situation. (Maybe it was not such a good idea after all to get rid of the priors above.) Also note that this danger does not exist for single Gaussians.

26.4. The Expectation Maximization (EM) Algorithm for Gaussian Mixture Models.

Setting the partial derivative of the log-likelihood (26.3) with respect to $\boldsymbol{\mu}_k$ equal to zero gives

$$\begin{aligned} 0 &= -\sum_{n=1}^{N} \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \Sigma_k)}{\sum_j \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \Sigma_j)} \Sigma_k (\mathbf{x}_n - \boldsymbol{\mu}_k) \\ &= -\sum_{n=1}^{N} \gamma(z_{nk}) \Sigma_k (\mathbf{x}_n - \boldsymbol{\mu}_k), \end{aligned}$$

or

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n, \tag{26.1}$$

where

$$N_k = \sum_{n=1}^{N} \gamma(z_{nk}).$$

Note that all the data points contribute towards the mean of the k-th mixture component, weighted with their responsibilities to the k-th component. The expression for the covariance follows similar lines,

$$\Sigma_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})(\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^T.$$

Again all the data points contribute towards the k-th mixture component, again weighted with their responsibilities.

In order to find the mixture coefficients we need to impose the constraint $\sum_{k=1}^{K} \pi_k = 1$. This is done by introducing a Lagrange multiplier into the log-likelihood,

$$\ln p(X|\boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma) + \lambda \left(\sum_{k=1}^{K} \pi_k - 1 \right).$$

Maximizing with respect to π_k, yields,

$$0 = \sum_{n=1}^{N} \frac{\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \Sigma_k)}{\sum_j \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \Sigma_j)} + \lambda.$$

Multiply both sides with π_k and summing over k gives $\lambda = -N$. Again multiplying both sides with π_k gives

$$0 = \sum_{n=1}^{N} \gamma(z_{nk}) - \pi_k N,$$

or

$$\pi_k = \frac{N_k}{N},$$

which is the average responsibility for the k-th mixture component.

Note these equations are not solvable in closed form since the responsibility depends in a complicated way on the parameters. Their form however, does suggest the following algorithm:

Expectation Maximization for GMMs

1. Initialize the the different coefficients, $\boldsymbol{\pi}$, $\boldsymbol{\mu}$, and Σ. One possibility is to use the K-means algorithm.

2. **E step**. Evaluate the responsibilities using the current estimates,

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \Sigma_j)}.$$

3. **M step**. Re-estimate the parameters using the current responsibilities

$$\begin{aligned}
\boldsymbol{\mu}_k^{\text{new}} &= \frac{1}{N_k}\sum_{n=1}^{N}\gamma(z_{nk})\mathbf{x}_n \\
\Sigma_k^{\text{new}} &= \frac{1}{N_k}\sum_{n=1}^{N}\gamma(z_{nk})(\mathbf{x}_n-\boldsymbol{\mu}_k)(\mathbf{x}_n-\boldsymbol{\mu}_k)^T \\
\pi_k^{\text{new}} &= \frac{N_k}{N}
\end{aligned}$$

where

$$N_k = \sum_{n=1}^{N}\gamma(z_{nk}).$$

4. Evaluate the log likelihood

$$\ln p(X|\boldsymbol{\pi},\boldsymbol{\mu},\Sigma) = \sum_{n=1}^{N}\ln\left\{\sum_{k=1}^{K}\pi_k\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k,\Sigma_k)\right\},$$

check for convergence of either the parameters, or the log likelihood.

5. Keep iterating from step 2 until convergence.

CHAPTER 27

KALMAN FILTERS

27.1. Introduction.

Up to now we have only used i.i.d. data. The next major step is to drop this assumption. The first example that we'll encounter where the observations are no longer independent is the Kalman filter.

27.2. Kalman Filter Equations.

Suppose we know that the dynamics of the system under consideration is given by

$$\mathbf{x}_{n+1} = A_n \mathbf{x}_n + \boldsymbol{\omega}_n \tag{27.1}$$

where $\mathbf{x}_n$ is the feature vector describing the system at time n, A_n is a matrix describing the transition from $\mathbf{x}_n$to $\mathbf{x}_{n+1}$. This means that the process is linear which is of course a strong assumption. It is possible to generalize; in fact, the derivation of the Kalman filter given in this Chapter is eminently suitable for generalization. The final term, $\boldsymbol{\omega}_n$, is a noise term. It may represent inadequate knowledge of the system, and/or noise introduced during the evolution of the system. We assume that the noise is Gaussian with zero mean,

$$p(\boldsymbol{\omega}_n) \;=\; \mathcal{N}(\boldsymbol{\omega}_n|\mathbf{0}, Q_n)$$

and that the $\boldsymbol{\omega}_n$ are independent for different n.

Apart from the dynamics, we also observe the system and the observation equations are given by

$$\mathbf{z}_{n+1} = H_n \mathbf{x}_{n+1} + \boldsymbol{\nu}_{n+1}. \tag{27.2}$$

The system is only indirectly observed and the observations $\mathbf{z}_{n+1}$ are linked to the features $\mathbf{x}_n$ by the matrix H_n. Again a linear relationship is assumed, something that can and should be generalized. The observation noise is given by $\boldsymbol{\nu}_{n+1}$and is again assumed to be zero mean and Gaussian,

$$\boldsymbol{\nu}_{n+1} \;=\; \mathcal{N}(\boldsymbol{\nu}_{n+1}|\mathbf{0}, R_n).$$

We also assume that the $\boldsymbol{\nu}_{n+1}$ are independent for different n. Since we assume linear processes and Gaussian noise, it should be clear that the state variables are also Gaussian, assuming Gaussian initial values. This makes life much easier, in fact, it allows an analytical solution of the Kalman equations.

Our task is to obtain the best possible estimate of the system $\mathbf{x}_n$, given the observations $Z_n = \{\mathbf{z}_j\}\,,\;\; j = 1, \ldots, n$, as well as our knowledge of the noise terms, i.e. given the covariances Q_n and R_n. For this purpose we rewrite the equations in probabilistic form. The dynamic equation becomes,

$$p(\mathbf{x}_{n+1}|\mathbf{x}_n, Z_n) = p(\mathbf{x}_{n+1}|\mathbf{x}_n) = \mathcal{N}(\mathbf{x}_{n+1}|A_n\mathbf{x}_n, Q_n) \tag{27.3}$$

and the observation equation becomes

$$p(\mathbf{z}_{n+1}|\mathbf{x}_{n+1}, Z_n) \;=\; p(\mathbf{z}_{n+1}|\mathbf{x}_{n+1}) \tag{27.4}$$

$$\;=\; \mathcal{N}(\mathbf{z}_{n+1}|H_{n+1}\mathbf{x}_{n+1}, R_n). \tag{27.5}$$

It should be pointed out that it is assumed that $\mathbf{x}_{n+1}$ is conditionally independent of the observations Z_n, given that $\mathbf{z}_n$ is known. Similarly, $\mathbf{z}_{n+1}$ is conditionally independent of Z_n, given that $\mathbf{x}_{n+1}$ is known.

Assuming that we know $p(\mathbf{x}_n|Z_n)$, our goal is to calculate $p(\mathbf{x}_{n+1}|Z_{n+1})$, given the latest observation $\mathbf{z}_{n+1}$. Since we know that $\mathbf{x}_n$ is a Gaussian variable, it is useful to write

$$p(\mathbf{x}_n|Z_n) = \mathcal{N}(\mathbf{x}_n|\widehat{\mathbf{x}}_n, P_n).$$

Note:

- This formulation leads naturally to a recursive implementation. Given $p(\mathbf{x}_0|Z_0)$ initially, we calculate $p(\mathbf{x}_n|Z_n)$ every time we receive a new observation $\mathbf{z}_n$.

The marginal distribution is given by

$$\begin{aligned} p(\mathbf{x}_{n+1}|Z_n) &= \int p(\mathbf{x}_{n+1}, \mathbf{x}_n|Z_n)d\mathbf{x}_n \\ &= \int p(\mathbf{x}_{n+1}|\mathbf{x}_n, Z_n)p(\mathbf{x}_n|Z_n)d\mathbf{x}_n \\ &= \int p(\mathbf{x}_{n+1}|\mathbf{x}_n)p(\mathbf{x}_n|Z_n)d\mathbf{x}_n. \end{aligned}$$

Since we know both Gaussian distributions under the integration, it follows immediately that

$$p(\mathbf{x}_{n+1}|Z_n) = \mathcal{N}(\mathbf{x}_{n+1}|A_n\widehat{\mathbf{x}}_n, Q_n + A_nP_nA_n^T). \tag{27.6}$$

Suppose we need the best estimate of the state variables $\mathbf{x}_{n+1}$ prior to observing $\mathbf{z}_{n+1}$. Usually one has to be careful in specifying exactly what is meant with 'best' estimate. Since we are dealing with Gaussian densities however, 'best' estimate in just about every meaningful sense of the word amounts to the same quantity namely the mean. Rewriting (27.6) as

$$p(\mathbf{x}_{n+1}|Z_n) = \mathcal{N}(\mathbf{x}_{n+1}|\mathbf{x}_{n+1}^-, P_{n+1}^-)$$

with

$$\mathbf{x}_{n+1}^- = A_n\widehat{\mathbf{x}}_n \tag{27.7}$$

and

$$P_{n+1}^- = Q_n + A_nP_nA_n^T$$

the best estimate of $\mathbf{x}_{n+1}$ prior to observing $\mathbf{z}_{n+1}$ is given by (27.7).

The marginal distribution of the observation $\mathbf{z}_{n+1}$ is given by

$$\begin{aligned} p(\mathbf{z}_{n+1}|Z_n) &= \int p(\mathbf{z}_{n+1}, \mathbf{x}_{n+1}|Z_n)d\mathbf{x}_{n+1} \\ &= \int p(\mathbf{z}_{n+1}|\mathbf{x}_{n+1}, Z_n)p(\mathbf{x}_{n+1}|Z_n)d\mathbf{x}_{n+1} \\ &= \int p(\mathbf{z}_{n+1}|\mathbf{x}_{n+1})p(\mathbf{x}_{n+1}|Z_n)d\mathbf{x}_{n+1}. \end{aligned}$$

Since both terms in the integrand are Gaussian, it follows in a straightforward way that the observation marginal is given by

$$p(\mathbf{z}_{n+1}|Z_n) = \mathcal{N}(\mathbf{z}_{n+1}|\mathbf{z}^-_{n+1}, S_{n+1})$$

where

$$\mathbf{z}^-_{n+1} = H_{n+1}\mathbf{x}^-_{n+1} \tag{27.8}$$

and

$$S_{n+1} = R_{n+1} + H_{n+1}P^-_{n+1}H^T_{n+1}. \tag{27.9}$$

Note:

- Since $\mathbf{x}^-_{n+1}$ is the best estimate of $\mathbf{x}_{n+1}$, $\mathbf{z}^-_{n+1}$ given by (27.8) is our best estimate of the next measurement, prior to actually receiving the next measurement.

Since we expect to next measure $\mathbf{z}^-_{n+1}$, the extent to which the actual measurement deviates from this expectation, is a measure of the how much we learn from the measurement at $n+1$. Using Bayes' theorem we now calculate

$$\begin{aligned} p(\mathbf{x}_{n+1}|Z_{n+1}) &= p(\mathbf{x}_{n+1}|\mathbf{z}_{n+1}, Z_n) \\ &= p(\mathbf{z}_{n+1}|\mathbf{x}_{n+1}, Z_n)p(\mathbf{x}_{n+1}|Z_n)/p(\mathbf{z}_{n+1}|Z_n) \\ &= p(\mathbf{z}_{n+1}|\mathbf{x}_{n+1})p(\mathbf{x}_{n+1}|Z_n)/p(\mathbf{z}_{n+1}|Z_n) \end{aligned}$$

where we made use of the various conditionally independencies. It again follows in s straightforward manner that

$$p(\mathbf{x}_{n+1}|Z_{n+1}) = \mathcal{N}(\mathbf{x}_{n+1}|\widehat{\mathbf{x}}_{n+1}, P_{n+1})$$

where

$$\widehat{\mathbf{x}}_{n+1} = P_{n+1}\left(P^-_{n+1}\right)^{-1}\mathbf{x}^-_{n+1} + P_{n+1}H^T_{n+1}R^{-1}_{n+1}\mathbf{z}_{n+1} \tag{27.10}$$

and

$$P_{n+1} = \left[\left(P^-_{n+1}\right)^{-1} + H^T_{n+1}R_{n+1}H_{n+1}\right]^{-1}. \tag{27.11}$$

We are almost done. These equations give us exactly what we want. The best estimate of $\mathbf{x}_{n+1}$, given the observation sequence Z_{n+1}, is given by $\widehat{\mathbf{x}}_{n+1}$with associated covariance P_{n+1}. All that remains to be done is simplify these equation, rewriting them in a more convenient form. For this purpose we define the so-called Kalman gain

$$W_{n+1} = P_{n+1}^{-} H_{n+1}^{T} S_{n+1}^{-1}. \tag{27.12}$$

It takes a bit of manipulation but it is not hard to show that this allows us to write

$$P_{n+1} = P_{n+1}^{-} - W_{n+1} S_{n+1} W_{n+1}^{T} \tag{27.13}$$

and

$$\widehat{\mathbf{x}}_{n+1} = \mathbf{x}_{n+1}^{-} + W_{n+1}\left(\mathbf{z}_{n+1} - \mathbf{z}_{n+1}^{-}\right). \tag{27.14}$$

In summary, given $\widehat{\mathbf{x}}_n$ and P_n:

1. $\mathbf{x}_{n+1}^{-} = A_n \widehat{\mathbf{x}}_n$, and $P_{n+1}^{-} = Q_n + A_n P_n A_n^T$ (using the dynamic information)
2. $\mathbf{z}_{n+1}^{-} = H_{n+1}\mathbf{x}_{n+1}^{-}$ and $S_{n+1} = R_{n+1} + H_{n+1} P_{n+1}^{-} H_{n+1}^T$ (using the measurement equation)
3. $W_{n+1} = P_{n+1}^{-} H_{n+1}^T S_{n+1}^{-1}$ (Kalman gain)
4. Given the new measurement $\mathbf{z}_{n+1}$, calculate

$$\widehat{\mathbf{x}}_{n+1} = \mathbf{x}_{n+1}^{-} + W_{n+1}\left(\mathbf{z}_{n+1} - \mathbf{z}_{n+1}^{-}\right) \text{ and } P_{n+1} = P_{n+1}^{-} - W_{n+1} S_{n+1} W_{n+1}^{T}.$$

27.2.1. Simplifying Using the Kalman Gain. We give the details of deriving (27.13) and (27.14) from (27.10) and (27.11) respectively.

It is the easiest to verify that

$$\begin{aligned} P_{n+1} &= \left[\left(P_{n+1}^{-}\right)^{-1} + H_{n+1}^T R_{n+1} H_{n+1}\right]^{-1} \\ &= P_{n+1}^{-} - W_{n+1} S_{n+1} W_{n+1}^T \end{aligned}$$

by showing that

$$\left[\left(P_{n+1}^{-}\right)^{-1} + H_{n+1}^T R_{n+1} H_{n+1}\right]\left[P_{n+1}^{-} - W_{n+1} S_{n+1} W_{n+1}^T\right] = I.$$

Multiplying out we find that

$$\begin{aligned}\left[\left(P_{n+1}^{-}\right)^{-1}+H_{n+1}^{T}R_{n+1}H_{n+1}\right]\left[P_{n+1}^{-}-W_{n+1}S_{n+1}W_{n+1}^{T}\right] &= \\ I-H_{n+1}^{T}S_{n+1}^{-T}H_{n+1}\left(P_{n+1}^{-}\right)^{T}+H_{n+1}^{T}R_{n+1}^{-1}H_{n+1}P_{n+1}^{-}- & \\ H_{n+1}^{T}R_{n+1}^{-1}H_{n+1}P_{n+1}^{-}H_{n+1}^{T}S_{n+1}^{-T}H_{n+1}\left(P_{n+1}^{-}\right)^{T} &= \\ I-H_{n+1}^{T}S_{n+1}^{-T}H_{n+1}\left(P_{n+1}^{-}\right)^{T}+H_{n+1}^{T}R_{n+1}^{-1}H_{n+1}P_{n+1}^{-}- & \\ H_{n+1}^{T}R_{n+1}^{-1}\left(S_{n+1}-R_{n+1}\right)S_{n+1}^{-T}H_{n+1}\left(P_{n+1}^{-}\right)^{T} &= I,\end{aligned}$$

where we have used (27.9) and the fact that covariance matrices are symmetric.

Starting from (27.10) we find that

$$\begin{aligned}\widehat{\mathbf{x}}_{n+1} &= P_{n+1}\left(P_{n+1}^{-}\right)^{-1}\mathbf{x}_{n+1}^{-}+P_{n+1}H_{n+1}^{T}R_{n+1}^{-1}\mathbf{z}_{n+1} \\ &= \left(P_{n+1}^{-}-W_{n+1}S_{n+1}W_{n+1}^{T}\right)\left(P_{n+1}^{-}\right)^{-1}\mathbf{x}_{n+1}^{-}+\left(P_{n+1}^{-}-W_{n+1}S_{n+1}W_{n+1}^{T}\right)H_{n+1}^{T}R_{n+1}^{-1}\mathbf{z}_{n+1} \\ &= \mathbf{x}_{n+1}^{-}-P_{n+1}^{-}H_{n+1}^{T}S_{n+1}^{-1}S_{n+1}S_{n+1}^{-T}H_{n+1}P_{n+1}^{-}\left(P_{n+1}^{-}\right)^{-1}\mathbf{x}_{n+1}^{-} \\ &\quad +\left(P_{n+1}^{-}-W_{n+1}S_{n+1}W_{n+1}^{T}\right)H_{n+1}^{T}R_{n+1}^{-1}\mathbf{z}_{n+1} \\ &= \mathbf{x}_{n+1}^{-}-P_{n+1}^{-}H_{n+1}^{T}S_{n+1}^{-1}\mathbf{z}_{n+1}^{-}+\left(P_{n+1}^{-}-W_{n+1}S_{n+1}W_{n+1}^{T}\right)H_{n+1}^{T}R_{n+1}^{-1}\mathbf{z}_{n+1} \\ &= \mathbf{x}_{n+1}^{-}-W_{n+1}\mathbf{z}_{n+1}^{-}+W_{n+1}S_{n+1}\left(I-W_{n+1}^{T}H_{n+1}^{T}\right)R_{n+1}^{-1}\mathbf{z}_{n+1}.\end{aligned}$$

If now use (27.9) it follows that

$$\begin{aligned}S_{n+1}\left(I-W_{n+1}^{T}H_{n+1}^{T}\right)R_{n+1}^{-1} &= I+H_{n+1}P_{n+1}^{-}H_{n+1}^{T}R_{n+1}^{-1}- \\ &\quad S_{n+1}S_{n+1}^{-1}H_{n+1}P_{n+1}^{-}H_{n+1}^{T}R_{n+1}^{-1} \\ &= I,\end{aligned}$$

and we are done.

CHAPTER 28

Dynamic Programming.

In practice we are often faced with a situation where it is necessary to compare different *signals* of different length. For instance, if one is interested in identifying a word or a phrase in a speech recognition application, one realizes that different speakers utter the same phrase in different ways, and with different durations. Comparing these signals is exactly the same problem we face in a signature verification problem. Again we have two similar signals, with differences in details and of different durations to compare. The algorithm we are about to describe was therefore extensive used in speech processing applications until it was largely replaced by even more powerful Hidden Markov Models. It should become clear that there is a straightforward generalization to vector-valued signals; for the moment we keep things as simple as possible and concentrate on single signals. The basic idea is to first define a suitable 'cost function' that gives an indication of the difference (total cost) between the signals. One then proceeds to find a map that maps one signal onto the other in such a way that the cost function is minimized. In this sense one therefore finds the best possible match between the two signals, taking into account that they are usually very different. One can then relate the total 'cost' (difference between the two signals) to a confidence value—the lower the cost, the higher the confidence that the two signals are actually different renderings of the same entity.

It might be interesting to note that the algorithms is much more general than just for comparing signals. It is a powerful algorithm for calculating maps minimizing an appropriate cost function. It is particularly useful in situations where one has only one example against which to compare, i.e. when it is not possible to build a model from a number of examples (as an additional advantage we can use it as a pro-type for the Viterbi algorithms that we will encounter a little later). For example, the line-break, paragraph-blocking algorithm used by TEX is based on this algorithm.

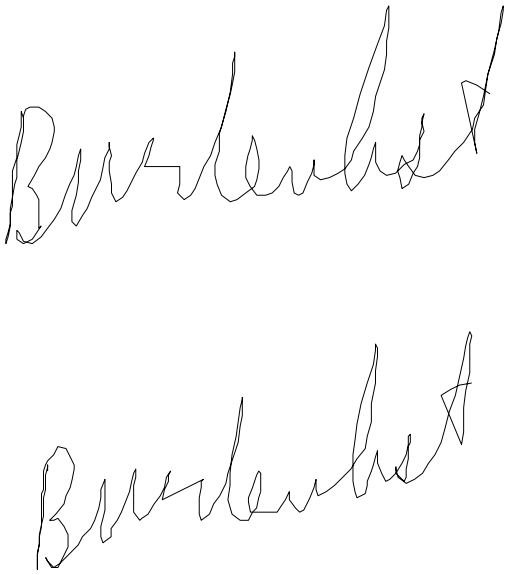


FIGURE 28.0.1. Two different signatures by the same signatory for comparison.

The naïve approach is to estimate the number number of words that will fit into a line, given the page width, and then calculate the inter-word distances so that the line is of exactly the specified length. The problem with this algorithm is that the result does not look good. Sometimes it is better to move a word to the next line for improved overall appearance, etc. Thus the best appearance is achieved by analyzing a whole paragraph and not simply each line separately. The way Knuth [**?**] does this is to define a suitable cost function that is constructed with overall appearance in mind. A mapping is then calculated that fills a whole paragraph in block form.

Let us have a closer look at the problem. In Figure 31.5.1 we show two different signatures by the same person with their y coordinates shown in Figure 31.5.2. It should be clear that the natural way of comparing these signals is to find matching points on the two signatures, such as the peaks indicated in Figure 31.5.2. This clearly requires a nonlinear stretching or 'warping' of the signatures to best fit each other. Mathematically this amounts to a *re-parametrization* of the two signatures.

Given the two discrete signals, $\mathbf{y}_1 := \{y1(t), \quad t = 1, \ldots, L_1\}$ and $\mathbf{y}_2 := \{y_2(s), \quad s = 1, \ldots, L_2\}$, the problem is to find re-parametrizations $p(w)$ and $q(w)$ such that one can identify $y1(p(w))$ with $y_2(q(w)), \quad w = 1, \ldots, L$ in such a way that the difference

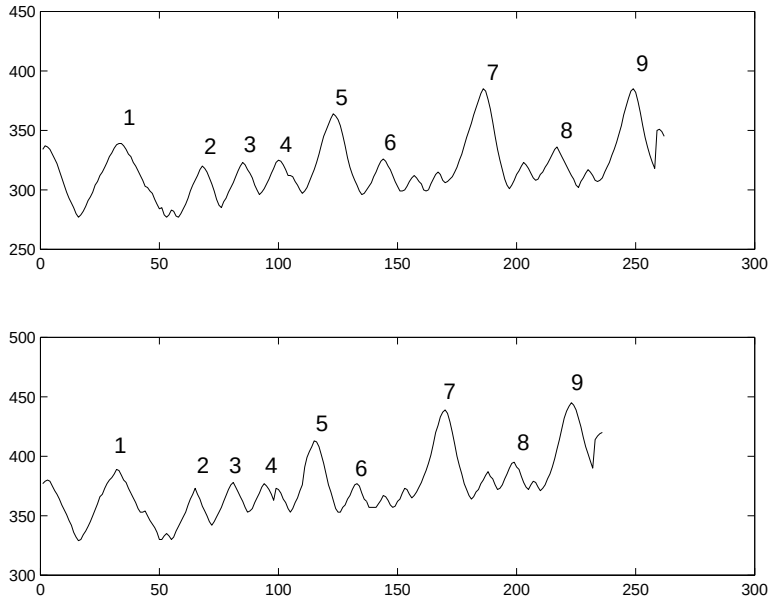


FIGURE 28.0.2. The y coordinates of the two signatures.

between the two resulting function is minimized. In order for $p(w)$ and $q(w)$ to be proper re-normalizations they need to be *non-decreasing* functions of w. Since the value of L depends on y_1 and y_2 it is determined as part of the algorithm.

Let us now become very specific and assume that the two signals we want to compare are given by,

$$\begin{aligned} \mathbf{y}_1 &= [1\ \ 0\ \ 1\ \ 2\ \ 1] \\ \mathbf{y}_2 &= [1\ \ 0\ \ 2\ \ 1]. \end{aligned}$$

Since we want to compare each value of $\mathbf{y}_1$ with each value of $\mathbf{y}_2$ it is convenient to draw a grid as shown in Figure 31.5.3. If we now draw a curve consisting of straight lines connecting the lower left-hand corner with the upper right-hand corner (the reader may wish to have a peak at Figure 31.5.7) below, any such curve defines a mapping between the two signals. It is important to note however that the monotonicity constraint on $p(w)$ and $q(w)$ implies that grid-point (i, j) can only be reached from either $(i-1, j)$, $(i, j-1)$ or $(i-1, j-1)$, as indicated (if there is a tie,

we give preference to the diagonal). It is this curve that we are after, constructed in such a way that the two signals are aligned in the best possible way (in a sense that has to be made precise). Also note that our assumption that the curve begins and ends at the two corners implies that we match the start– and endpoints of the two signals. In some applications it is useful to relax this constraint—it is absolutely straightforward to relax the endpoint matching constraint, as will become clear.

The brute-force way of solving the problem is to investigate all possible paths connecting the lower left-hand corner with the upper right-hand corner. This is a use number and the reader may find it a fun exercise to derive the following formula for the total number of possible paths

$$\mathcal{N} = \sum_{s=0}^{\min(L_1,L_2)} \frac{(L_1 + L_2 - s)!}{(L_1 - s)!(L_2 - s)!s!},$$

constrained only by our monotonicity requirement. This is an enormous number, dominated by the first term which counts only the number of paths not containing a diagonal,

$$(L_1 + L_2)!/L_1!L_2!.$$

Even this is too large to investigate exhaustively by computer for all but the smallest number of samples. We clearly need to do better.

All the most efficient algorithms are based on a very simple observation (which the reader can prove for herself): Each sub-path of an optimal path, is also optimal. This means that the global optimal path is pieced together from local optimal paths—exactly the defining strategy of Dynamic Programming (DP). The key is to define a local distance measure (local cost function) that measures the difference between the two functions. An obvious choice is

$$C_{i,j} = d(y1(i), y2(j)) := |y_1(i) - y_2(j)|, \quad i = 1, \ldots, L_1; \quad j = 1, \ldots, L_2. \tag{28.1}$$

For our example, this is written as

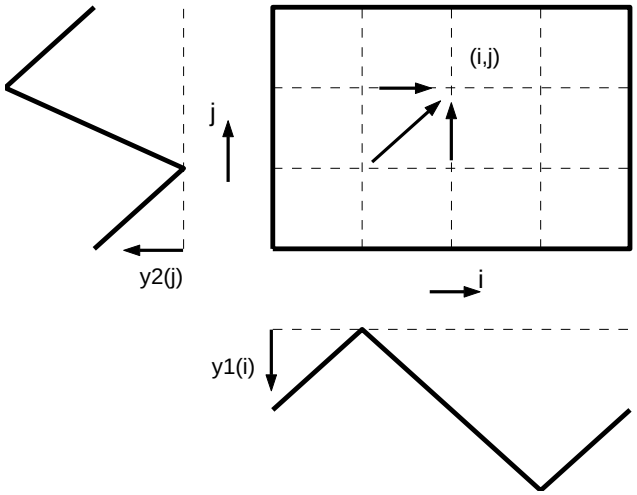


FIGURE 28.0.3. The two signals to be compared.

$$C = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 2 & 1 & 0 & 1 \\ 1 & 0 & 1 & 2 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix},$$

where one should note that the ordering corresponds to the ordering of the grid of Figure 31.5.4. Also note that in Figure 31.5.4 that it is convenient for us to number the grid points consecutively from 1 to 20. For example, grid point 9 has coordinates $(4, 2)$ and its local distance value is $C_{4,2} = 2$. Based on this local distance measure, the total cost function for the two signals $y_1(t)$ becomes

$$C = \sum_{w=1}^{L} |y_1(p(w)) - y_2(q(w))| \,. \tag{28.2}$$

Once the the local costs have been calculated, the rest of the algorithm proceeds without any reference to the original signals—it allows us to calculate the optimal path from the lower left-hand corner to every other point on the grid. This may sound wasteful but with the slight modification explained below, it is still the most efficient algorithm known. The result is shown in Figure 31.5.5. The Figure shows

the optimal path to each grid point with the total cost of each path. For example, the cost of reaching point 19 with coordinates $(4, 4)$, is 2. In order find its optimal path, we start at point and then backtrack until we reach point 1. Note that all we need is to know from where a specific point is reached, i.e. point 19 is reached from point 13, is reached from point 7, etc. Accordingly, we keep track of all the different paths, by defining an array $I(i), i = 1, \ldots, L_1 L_2$ where $I(k)$ denotes the point from which point k is reached. For our example, Figure 31.5.5, we find that $I(20) = 14, I(19) = 13$, etc. Thus the optimal path is obtained from $I(20) = 14,\ I(14) = 8,\ I(8) = 7,\ I(7) = 1$.

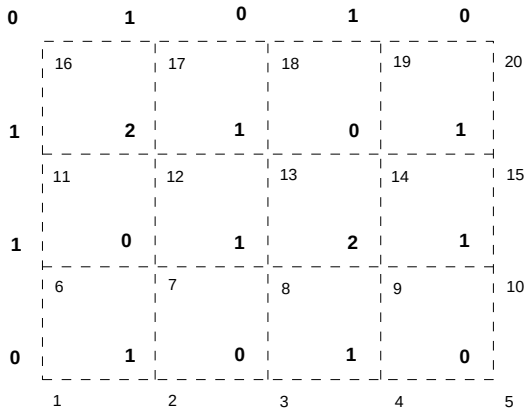


FIGURE 28.0.4. The local cost grid.

Let us proceed with the description of the algorithm. Since the local cost for point 1 is zero (see Figure 31.5.4), the total cost to reach grid point 1 is $TC(1) = C_1 1 = 0$. Noting that point 2 can only be reached from point 1, there is no decision to make and the total cost of reaching point 2 is: $TC(2) = TC(1) + C_{21} = 1$. We also record that it is reached via the optimal (and only) path from point 1, i.e. we set $I(2) = 1$. For points 3, 4, 5 and 6 we do the same, $TC(3) = TC(2) + C_{31} = 1, \quad I(3) = 2, \quad TC(4 = TC(3 + C_{41} = 2, \quad I(4) = 3, \quad TC(5 = TC(4 + C_{51} = 2, \quad I(5) = 4, \quad TC(6 = TC(1 + C_{12} = 1, \quad I(6) = 1$. However, point 7 may be reached from either points 1, 2 or 6. We already know the optimal paths to points 1, 2 and 6. Since the cost of the path to point 1, is less or equal to the cost to points 2 or 6, we reach point 7 via point 1 and set $TC(7) = TC(1) + C_{22} = 0, \quad I(7) = 1$. Now we do the same for the rest of the points on the grid:

$$
\begin{aligned}
TC(8) &= TC(7) + C_{32} = 1, \quad I(8) = 7 \\
TC(9) &= TC(3) + C_{42} = 3, \quad I(9) = 3 \\
TC(10) &= TC(4) + C_{52} = 3, \quad I(10) = 4 \\
TC(11) &= TC(6) + C_{13} = 2, \quad I(11) = 6 \\
TC(12) &= TC(6) + C_{23} = 2, \quad I(12) = 6 \\
TC(13) &= TC(7) + C_{33} = 1, \quad I(13) = 7 \\
TC(14) &= TC(8) + C_{43} = 1, \quad I(14) = 8 \\
TC(15) &= TC(14) + C_{53} = 2, \quad I(15) = 14 \\
TC(16) &= TC(11) + C_{14} = 2, \quad I(16) = 11 \\
TC(17) &= TC(11) + C_{24} = 3, \quad I(17) = 11 \\
TC(18) &= TC(13) + C_{34} = 1, \quad I(18) = 13 \\
TC(19) &= TC(13) + C_{44} = 2, \quad I(19) = 13 \\
TC(20) &= TC(14) + C_{54} = 1, \quad I(20) = 14.
\end{aligned}
$$

With the help of I it is now a simple matter of finding the optimal path as pointed out above, and shown in Figure 31.5.6, with its total cost $TC(20) = 1$.

The computational cost of this algorithm amounts to three tests at each grid point, i.e. it is of $O(L_1 L_2)$. This can be further reduced by realizing that the optimal path should not stray too far from the diagonal, at least not for similar signals. One can therefore restrict the search to a band around the diagonal, further reducing the computational cost. If we restrict the search to a band of width d, the computational cost is of $O(L_1)$ with a possibly large constant, depending on d (assuming $L_1 \geq L_2$).

Figure 31.5.7(a) shows the matching of the y coordinates of different signatures belonging to the same person. It is interesting to note how close the warping function remains to the diagonal, an indication that there is a good correspondence between the two functions. If one now plots y_1 and y_2 against their re-parametrizes indexes, Figure 31.5.7(b) shows how the different peaks are now perfectly aligned. Incidentally, the total cost in this case is 374.1.

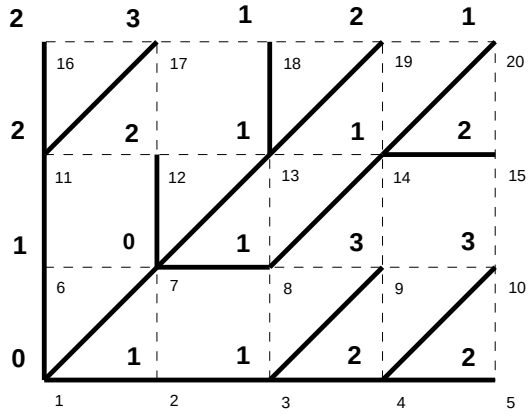


FIGURE 28.0.5. The optimal paths to all the grid points.

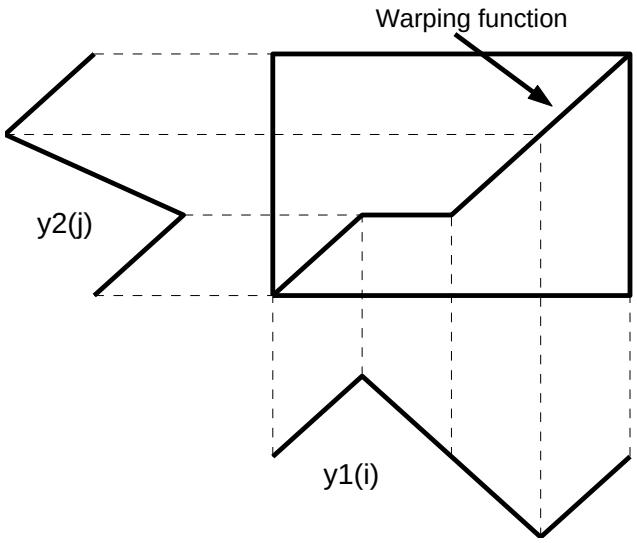


FIGURE 28.0.6. The optimal path.

Let us now do the same thing for two completely different signatures as shown in Figure 31.5.8. One may think of the second signature as a *casual* forgery since the forger clearly had no idea what the original looked like. The point is that the algorithm still finds the best possible match, which in this case is not good at all.

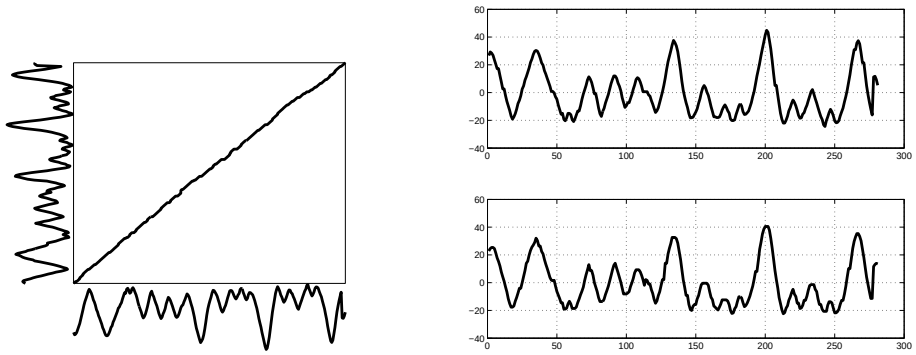


FIGURE 28.0.7. Matching two similar signatures.(a) The warping path. (b) The two signals aligned.

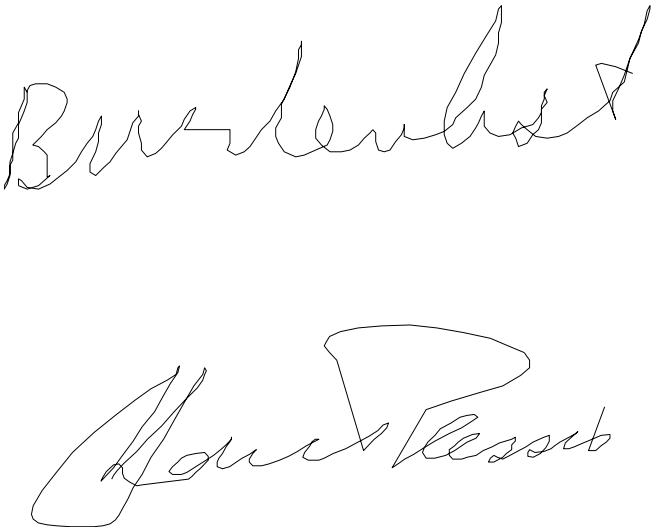
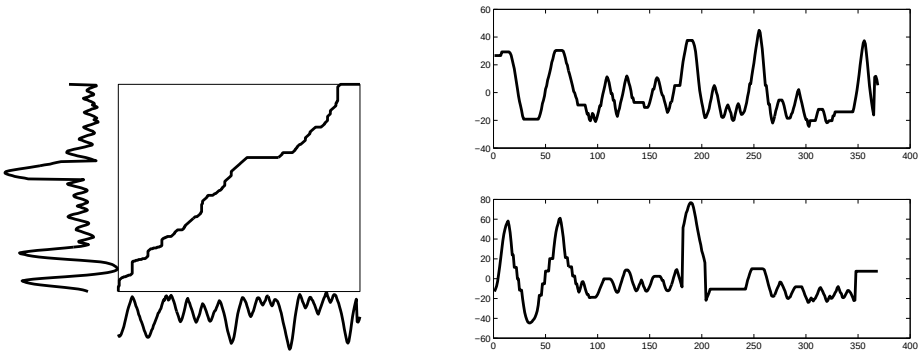


FIGURE 28.0.8. Two unrelated signatures.

The y coordinates, their warping function as well as their best possible alignment are shown in Figure 31.5.9. Although the algorithm still finds the best match, the warping function deviates considerably from the diagonal. The big difference between the two signatures is reflected by the large value of its total cost function, 2 885.



(a) (b)

FIGURE 28.0.9. Matching two unrelated signatures.(a) The warping path. (b) The two signals aligned.

CHAPTER 29

HIDDEN MARKOV MODELS

29.1. Introduction.

Hidden Markov models (HMMs) and Kalman filters are related in the sense that both are models where the usually unobserved internal state governs the external observations that we make. Furthermore the internal state at the next time step is only determined by the current internal state (the so-called 1st-order Markov property). However, they are also very different. Firstly, while the Kalman filter has a continuous state space (i.e. its internal state is described by a set of real-valued numbers), the HMM has a discrete state space (i.e. its state can be depicted by a limited set of integers). The Kalman filter changes its state in a linear manner (i.e. a matrix multiplication will take you from its current state to the next). The HMM changes its state in a probabilistic manner, a transition matrix specifies which states can follow on the current one and with what probability this will happen. This makes its behaviour to be very non-linear. Although these links are intriguing, we will develop the our understanding of the HMM by taking a further step back to pick up the thread from the Naive Bayes approach we previously encountered.

29.2. Basic concepts and notation

Hidden Markov models (HMMs) and Kalman filters are related in the sense that both are models where the usually unobserved internal state governs the external observations that we make. Furthermore the internal state at the next time step is only determined by the current internal state (the so-called 1st-order Markov property). However, they are also very different. Firstly, while the Kalman filter has a continuous state space (i.e. its internal state is described by a set of real-valued numbers), the HMM has a discrete state space (i.e. its state can be depicted by a limited set of integers). The Kalman filter changes its state in a linear manner (i.e.

a matrix multiplication will take it from its current state to the next). The HMM changes its state in a probabilistic manner, a transition matrix specifies which states can follow on the current one and with what probability this will happen. This makes its behavior very non-linear. Although these links are intriguing, we will develop the our understanding of the HMM by taking a further step back to pick up the thread from the Naive Bayes approach we previously encountered.

In the naive Bayes approach we modeled a sequence of T feature vectors (or observations) $\mathbf{x_1^T}$ as $p(\mathbf{x}_1^T|M) = \prod_{t=1}^{T} p(\mathbf{x}_t|M)$[1]. We got from the joint density (pdf) to the product of individual densities by making a very strong assumption, namely that each feature vector $\mathbf{x}_t$ is statistically independent from every other one, while also sharing a common marginal pdf with all of them. This immediately destroys any hope that this model will be able to capture some time-dependent behavior between the various $\mathbf{x}_t$'s, in fact jumbling them into any arbitrary time order will not affect the resulting pdf value at all.

29.2.1. Emitting states. We now want to make less drastic assumptions that will allow us to model time-dependent behavior whilst at the same time keeping computations tractable. HMMs do this by introducing the concept that the model is in 1 of N states at any given time. However, we typically do *not* know in which state we are at such a given time - hence the name *hidden* Markov model. Each state $s = i,\ i = 1, \ldots, N$ on its own behaves very much like the naive Bayes / iid model we previously encountered. It has a pdf describing the feature vectors associated with it, ie $p(\mathbf{x}|s)$. In this way one can think of the naive Bayes model as a 1 state HMM, or alternatively of an HMM as a series of naive Bayes models with probabilistic jumps between them. We will need to refer generically to the state that is active at time t, we will use the notation s_t for this.

29.2.2. Transitions. These states are coupled to each other with transition probabilities $a_{i,j} = P(s_{t+1} = j|s_t = i)$. In other words, it indicates the probability that we will transit to state j at the next time step *if* we knew that we are in state i at the current time step (which we of course unfortunately do not know). Of course,

[1]Here we use the '$|M$' to indicate that the calculation is being done using a specific set of model parameters. When it is clear from the context that we are referring to a specific model we may omit this for the sake of simplicity.

once one has made a transition to a new state, a new pdf apply, thereby giving the HMM the ability to model temporal patterns. This basically allows us to change the model behavior as a function of time. These probabilities are collected in a transition matrix A, its row indexes indicating the source state and the columns indicating the destination state of the transition. Since all the probabilities departing from a given state must sum to one, each row in this matrix will sum to one.

29.2.3. Non-emitting/null states. In addition to the above we also need to be able to specify the states in which our model can start (the initial states) and those in which it can terminate. We do this by adding two special states without any densities, namely state 0 and state $N + 1$[2]. These states are called null or non-emitting states to distinguish them from the normal/emitting states we encountered above [3]. State 0 will have no links entering it, and state $N + 1$ will have no links leaving from it. We require that the process always start in state 0 and terminate in state $N + 1$. As can be seen from figure 29.2.3 this allows for arbitrary initial and terminal states. We extend the s_t notation to allow for these states by also including an s_0 and s_{T+1}. Note, however, that the transition to the final state does not occupy an extra time step since it has no pdf. Therefore s_{T+1} actually is active at the same time as s_T.

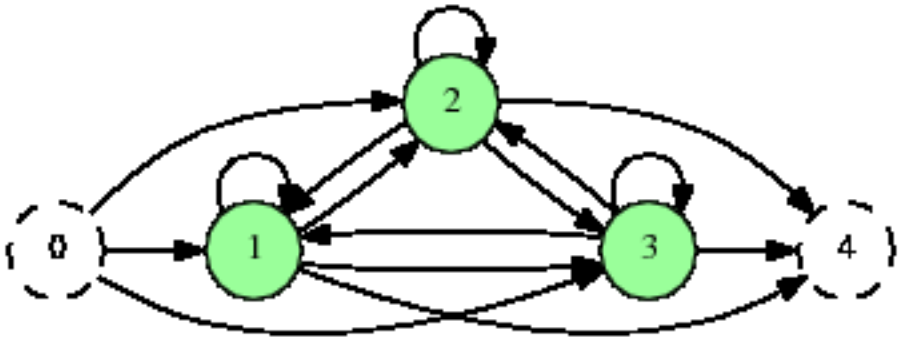


FIGURE 29.2.1. A simple fully connected HMM

[2]It is also possible and useful to use such null states in other places in the model. Since it complicates algorithms we refrain from using this here.

[3]Our notation deviates here from the common convention to use a vector π to indicate initial states, and possibly another vector $\mathbf{F}$ to indicate final/terminating states.

29.2.4. Fundamental HMM assumptions. The above model makes two important assumptions about the relationship between the feature vectors:

(1) The observation independence assumption states that:

$$p(\mathbf{x}_t|\mathbf{x}_1^{t-1}, s_0^t) = p(\mathbf{x}_t|s_t). \tag{29.1}$$

This means that the likelihood of the tth feature vector depends only on the current state and is therefore otherwise unaffected by previous states and feature vectors. This assumption is not affected by the order of the HMM.

(2) The first-order[4] Markov assumption:

$$P(s_t|s_0^{t-1}, \mathbf{x}_1^{t-1}) = P(s_t|s_{t-1}). \tag{29.2}$$

This implies that, apart from the immediately preceding state, no other previously observed states or features affect the probability of occurrence of the next state.

In addition to the above the reader will note that our notation $a_{i,j}$ for transition probabilities does not allow them to be time dependent. We assume the transition probability between two states to be constant irrespective of the time when the transition actually takes place.

29.2.5. A few basic HMM topologies. Topology refers to which states are connected to each other. Many configurations are popular, it usually makes sense to carefully match the topology to the characteristics of the observations that are being modeled. The most generic version is the fully connected topology shown in Figure 29.2.3. It is often used in applications where observations repeats over time (for example a text-independent speaker recognition system).

When the observations has a well-defined sequential nature, such as encountered in for instance word recognition, a left-to-right form such as shown in Figure 29.2.5 may be more appropriate.

As mentioned above, many other topologies are useful, this will be discussed at a later stage.

[4]This can also be generalized to (more powerful) higher Markov orders.

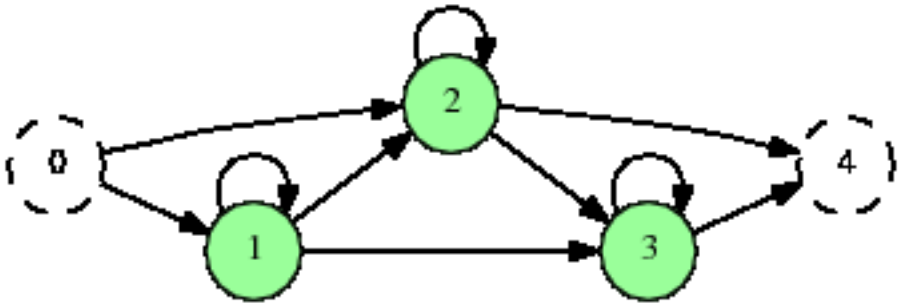


FIGURE 29.2.2. A simple left-to-right HMM

29.3. Calculating $p(\mathbf{x}_1^T|M)$

In the following we repeatedly use two results from basic probability theory namely conditional probability (the product rule)

$$p(a, b|c) = p(a|b, c)p(b|c) \tag{29.1}$$

and total probability (the marginal)

$$p(a) = \sum_i p(a, b_i) \tag{29.2}$$

where b_i forms a partition. For simplicity we will (mostly) also omit the reference to the model M.

29.3.1. A direct approach. Marginalizing, we can write the required probability as,

$$p(\mathbf{x}_1^T) = \sum_{\forall s_0^{T+1}} p(\mathbf{x}_1^T, s_0^{T+1}). \tag{29.3}$$

Using the product rule, the Markov assumption, and the observation independence, we write,

$$\begin{aligned} p(\mathbf{x}_1^T, s_0^{T+1}) &= P(s_{T+1}|s_0^T, \mathbf{x}_1^T)p(\mathbf{x}_1^T, s_0^T) \\ &= a_{s_T, s_{T+1}} p(\mathbf{x}_T|\mathbf{x}_1^{T-1}, s_0^T)p(\mathbf{x}_1^{T-1}, s_0^T) \\ &= a_{s_T, s_{T+1}} p(\mathbf{x}_T|s_T)p(\mathbf{x}_1^{T-1}, s_0^T). \end{aligned}$$

The last term on the right hand side is exactly of the same form as the left hand side, we can therefore recursively complete the evaluation to yield,

$$\begin{aligned} p(\mathbf{x}_1^T, s_0^{T+1}) &= a_{s_T,s_{T+1}} p(\mathbf{x}_T|s_T) a_{s_{T-1},s_T} p(\mathbf{x}_{T-1}|s_{T-1}) \dots \\ &\quad \dots a_{s_1,s_2} p(\mathbf{x}_1|s_1) P(s_1|s_0) P(s_0) \\ &= a_{s_T,s_{T+1}} p(\mathbf{x}_T|s_T) a_{s_{T-1},s_T} p(\mathbf{x}_{T-1}|s_{T-1}) \dots \\ &\quad \dots a_{s_1,s_2} p(\mathbf{x}_1|s_1) a_{0,s_1} \end{aligned} \tag{29.4}$$

Note that the final state s_{T+1} is a non-emitting, terminating state previously indicated as $s_{T+1} = N+1$. Since all the values on the right hand side are either known, or can be readily calculated, it would seem that we have succeeded in providing an approach towards calculating (29.3). The snag lies in the $\forall s_0^{T+1}$. In a fully connected model with N states and T time steps the number of possible state sequences is N^T— which very rapidly becomes prohibitively large[5]. We need to find another method that can do this more efficiently.

29.3.2. The forward algorithm. Let us consider the calculation of the so-called forward likelihoods,

$$\begin{aligned} \alpha_t(j) &= p(\mathbf{x}_1^t, s_t = j), \quad t = 2, \dots, T \text{and } j = 1, \dots, N) \\ &= p(\mathbf{x}_t|\mathbf{x}_1^{t-1}, s_t = j) p(\mathbf{x}_1^{t-1}, s_t = j) \\ &= p(\mathbf{x}_t|s_t = j) \sum_{i=1}^{N} p(\mathbf{x}_1^{t-1}, s_{t-1} = i, s_t = j) \\ &= p(\mathbf{x}_t|s_t = j) \sum_{i=1}^{N} P(s_t = j|s_{t-1} = i, \mathbf{x}_1^{t-1}) p(\mathbf{x}_1^{t-1}, s_{t-1} = i). \end{aligned}$$

Recognizing that the last term on the right hand side is $\alpha_{t-1}(i)$, and using the Markov assumption this reduces to a very usable recursive form,

[5] with $N = 500$ and $T = 300$ we already have approximately 10^{800} paths!

$$\alpha_t(j) = p(\mathbf{x}_t|s_t = j) \sum_{i=1}^{N} a_{i,j}\alpha_{t-1}(i),$$

or if we want to state it more generally to include state 0,

(29.5)
$$\alpha_t(j) = \begin{cases} p(\mathbf{x}_t|s_t = j) \sum_{i=1}^{N} \alpha_{t-1}(i)a_{i,j} & \text{with } t = 1, \ldots, T, \;\; j = 1, \ldots, N \\ 0 & \text{with } t = 1, \ldots, T, \;\; j = 0. \end{cases}$$

To start the recursion at $t = 0$ we need $\alpha_0(j) = P(s_0 = j)$,

(29.6)
$$\alpha_0(j) = \begin{cases} 1 & \text{with } j = 0 \\ 0 & \text{with } j > 0 \end{cases}$$

The total likelihood of the data given the model fits very nicely into this framework. Starting by marginalizing out all posible states that lead to termination, we get,

$$\begin{aligned} p(\mathbf{x}_1^T) &= \sum_{j=0}^{N} p(\mathbf{x}_1^T, s_T = j, s_{T+1} = N + 1) \\ &= \sum_{j=0}^{N} P(s_{T+1} = N + 1|s_T = j, \mathbf{x}_1^T)p(s_T = j, \mathbf{x}_1^T) \\ &= \sum_{j=0}^{N} a_{j,N+1}\alpha_T(j). \end{aligned} \tag{29.7}$$

In a fully connected model with N emitting states and T time steps the number of calculations is now $O(N^2T)$, which is very do-able indeed.

29.3.3. Preventing numerical over-/underflow. Inspecting equation 29.5 reveals repeated multiplication of quantities that are either probabilities and therefore in the [0:1] range, or high-dimensional pdf values that, although they could have any positive value, are likely to be very small. In practical terms the iterative calculation of the above α's *will* underflow and we need to take precautions against this. Two approaches are popular:

- Rescale the α's: After all the $\alpha_t(j)$'s at a specific time t have been calculated, divide them by their sum $\sum_j \alpha_t(j)$ while also recording this sum in log format in a separate variable. Accumulating these log-sums over time will yield the exact factor which should be added to the calculated/scaled $\log p(\mathbf{x}_1^T)$ to yield its correct value.
- Work with $\log \alpha_t(j)$ throughout: Working in log format should prevent any underflow problems. However, now we need to be careful with the summation in equation 29.5 which still needs to be done in the linear domain. Directly converting these log values to linear before summation will immediately re-introduce the underflow problems we are trying to prevent. We come up against expressions such as $L = \log(e^{L_1} + e^{L_2} + \ldots + e^{L_M} + \ldots + e^{L_N})$ where none of the individual terms e^{L_n} are expressible in linear form. This is not as daunting as it might seem at first and can be calculated as $L = L_M + \log(e^{L_1 - L_M} + e^{L_2 - L_M} + \ldots + 1 + \ldots + e^{L_N - L_M})$ where $L_M = \max(L_1, L_2, \ldots L_N)$.

29.4. Calculating the most likely state sequence: The Viterbi algorithm

We see in (29.3) that the total likelihood of the data involves a sum of all possible state sequences s_0^{T+1}. Due to the 'hiddenness' of the HMM, we never know with certainty which of those many state sequences actually give rise to our observed feature vectors $\mathbf{x}_1^T$. But at least one of them will provide the biggest contribution to this sum. This is the most likely state sequence. We can find it easily with a small modification to (29.5) and (29.7). If we replace the summations there with maximizations, while also recording which specific previous state resulted in each such a maximum, we have calculated $p(\mathbf{x}_1^T, S^*)$ where S^* denotes the most likely state sequence. This most likely state sequence can be recovered by recursively backtracking from state $N+1$ to its most likely predecessor etc, until we reach state 0. It is left as an exercise for the reader to determine why this procedure results in the optimal state sequence (hint: the above is a dynamic programming algorithm). Once again one has to give consideration to possible underflow problems. However, by using these Viterbi maximizations, the total calculation has reverted to a single long sequence of multiplications, exactly as we found in equation 29.4. It now becomes

very simple to work in the log domain, this sequence of multiplications simply changes to additions.

29.5. Training/estimating HMM parameters

The GMM we encountered in a previous chapter, weighed and combined a collection of observation pdfs. Because we did not know which of the basis-pdfs we should associate a specific feature vector (i.e. there was missing/latent information), we had no closed form solution for estimating its parameters. We had to resort to the EM algorithm to supply an interactively improving estimate. This estimate converged only to a locally optimal set of parameters, there is no guarantee that the solution is the best possible one.

The HMM also combines a collection of observation pdfs. This combining is even more complex now, since the time order of observations also plays an important role. Once again there is missing information, in this case we do not know for sure which state we should associate each feature vector with. Similar to the GMM case we will find that we can estimate the model parameters via the EM algorithm, and once again the solution will only have the guarantee of being locally optimal.

29.5.1. If the (hidden) state sequence is known. Suppose for the moment that for each sequence of training observations/feature vectors $\mathbf{x}_1^T$ we somehow knew the corresponding state sequence s_1^T. From this it is fairly easy to estimate the model parameters.

- **Transition probabilities:** For the transition probabilities we simply have to count how often various states follow on each specific source state.

$$\hat{a}_{i,j} = \frac{\#(s_t = i, s_{t+1} = j)}{\#(s_t = i)}, \tag{29.1}$$

  where we use $\hat{}$ to indicate an estimate, and # to indicate the count of the number of occurrences of its argument.
- **Observation pdfs:** Similarly we would calculate the parameters of the state pdfs $p(\mathbf{x}|s = i)$ by simply collecting all the feature vectors associated with each state i and from them estimate the pdf parameters using techniques we have encountered before. This will depend on the specific

pdf being used, it could be as simple as calculating the mean and variance vectors of a diagonal Gaussian[6].

Since the transition probabilities and state pdfs comprise our full HMM, it therefore is quite simple to estimate the model parameters *if* we know the (hidden/latent) state sequences. But of course we, unfortunately, do not. However, *if* we somehow knew the HMM parameters, we can use the Viterbi algorithm to estimate the optimal state sequence S^* for each $\mathbf{x}_1^T$. Using this estimated state sequence should be a good stand-in for the the true state sequence. So we have a chicken-egg situation here. With known state sequences we can calculate the model parameters and with known model parameters we can calculate optimal state sequences. But unfortunately both of these 'knowns' are actually unknown. This is where the EM algorithm, here in the form of the Viterbi Re-estimation algorithm, comes in.

29.5.2. The EM algorithm — Viterbi Re-estimation.

(1) **Initialization:**
   (a) For every training observation sequence $\mathbf{x}_1^T$, assign in a sensible manner a corresponding state sequence s_1^T and extend it to s_0^{T+1} by adding the initial and termination state. This 'sensible' manner is dependent on the desired topology. For a fully connected model clustering might provide initial labels, for left-to-right models each training sequence can be subdivided in N equal portions[7].
   (b) From this initial state sequence, generate an initial model as discussed in section 29.5.1.

(2) **EM Re-estimation:**
   (a) **Expectation step:** For the current model estimate, apply the Viterbi algorithm on every training sequence $\mathbf{x}_1^T$, to calculate $\log p(\mathbf{x}_1^T, S^*)$. S^* is the expected state sequence for this observation sequence. Accumulate the 'scores' i.e. $f = \sum_{\forall \text{ training } \mathbf{x}_1^T} \log p(\mathbf{x}_1^T, S^*)$, to be used later to test for convergence.

[6]It is quite common for the pdf to be a GMM.

[7]A random state assignment typically ultimately results in an inferior local optimum and is not recommended.

(b) **Maximization step:** Use *all* the S^*'s obtained in the E-step to update the parameters of the HMM as discussed in section 29.5.1.

(3) **Termination:** Compare the total score f obtained in the E-step to that obtained from the previous E-step. If it is within an acceptable tolerance, terminate, otherwise continue with the re-estimation (i.e. step 2).

29.5.3. The EM algorithm — Baum Welch Re-estimation. The reader might have noticed that in the re-estimation of GMM's, each feature vector $\mathbf{x}$ was partially/probabilistically associated with multiple basis pdfs, where-as in the Viterbi algorithm we associated it fully with only one specific state. This is more similar to what we encountered with K-means clustering where a feature vector was also only associated with one specific cluster.

For all three the above training scenarios we actually have a choice whether we want to use the 'hard' approach which associates each feature vector with only one missing label namely the most probable one, or the 'soft' approach which generalizes this by probabilistically/partially associating the feature vector with multiple labels in accordance to our probabilistic knowledge of the situation. Below we briefly outline this algorithm.

The 'soft' version of the Viterbi re-estimation is called Baum Welch re-estimation. To do this we need to use the so-called 'backwards algorithm' to calculate another set of likelihoods:

$$\beta_t(j) = p(\mathbf{x}_{t+1}^T | s_t = j).$$

Note that the time sequence now runs in the opposite direction, x_t is *not* included, and s_t is now a given. Similarly to the α's there is a recursive algorithm to calculate them efficiently,

$$\beta_{T+1}(j) = \begin{cases} 1 & \text{with } j = N+1 \\ 0 & \text{with } j < N+1 \end{cases} \tag{29.2}$$

and

(29.3)
$$\beta_t(j) = \begin{cases} \sum_{k=1}^{N+1} a_{j,k}\beta_{t+1}(k)p(\mathbf{x}_{t+1}|s_{t+1}=k) & \text{with } t=1\ldots T, j=1\ldots N \\ 0 & \text{with } t=1\ldots T, j=N+1 \end{cases}$$

From these and (29.7) we can then calculate the probability that a specific state $s_t = j$ is active,

$$\gamma_t(j) = p(s_t = j|\mathbf{x}_1^T) = \frac{\alpha_t(j)\beta_t(j)}{p(\mathbf{x}_1^T)}. \tag{29.4}$$

We can also calculate the probability that a specific transition is active,

$$\xi_t(j,k) = p(s_t = j, s_{t+1} = k|\mathbf{x}_1^T) = \frac{\alpha_t(j)a_{j,k}p(\mathbf{x}_{t+1}|s_{t+1}=k)\beta_{t+1}(k)}{p(\mathbf{x}_1^T)}. \tag{29.5}$$

These then are the 'soft'/probabilistic counterpart to the 'hard'/optimal state and transition labellings resulting from the Viterbi algorithm. The EM algorithm we encountered in section 29.5.2 still applies, but now we no longer use a hard allocation of feature vectors x_t to states s_t. Instead by using equation 29.4 and 29.5 we are allocating each x_t to all states, but proportionally to the probability that it matched the state. For instance the transition probabilities equation 29.1 now become:

$$\hat{a}_{i,j} = \frac{\sum_{t=0}^{T}\xi_t(i,j)}{\sum_{t=0}^{T}\gamma_t(i)}. \tag{29.6}$$

Similar extensions apply to the estimation of state densities, but is dependent on the specific density being used. The reader is encouraged to for instance find the maximum likelihood estimate applicable if the state densities were multi-dimensional Gaussian. For establishing convergence/termination we can monitor $p(\mathbf{x}_1^T)$.

In terms of modeling accuracy this is the more general approach. In practical terms, however, the accuracy of these algorithms are mostly very similar. We are not going to investigate this further here, consult the literature for more details.

Part 5

MODELING PROJECTS

CHAPTER 30

DETERMINING THE STRUCTURES OF MOLECULES BY X-RAY DIFFRACTION

30.1. Introduction.

Distances between the atoms in a solid (or within a molecule) are on the order of 1 Å ngström (1Å ngström $= 10^{-10}$ m). To probe the positions of individual atoms using electromagnetic waves, requires

- The waves must have a wavelength λ that is no larger than about 10^{-10} m. From Planck's law $E = h\nu = \frac{hc}{\lambda}$ with Planck's constant $h \approx 6.625 \cdot 10^{-34}$ Js and speed of light $c \approx 3.00 \cdot 10^8$ m/s follows the energy of an individual photon $E \approx 1.99 \cdot 10^{-15}$ J $\approx$ 12.4 KeV. This energy is typical of X-rays.
- Most objects are highly transparent to X-rays. To get a reasonably strong scattered signal (and not just all rays just passing through), we need a very large number of molecules held firmly in perfect alignment. Nature does just that in crystals. Besides, there would not be any good practical way to hold a single molecule immobile for any length of time.

Since the scattered signal is very weak compared to what passes through a crystal sample, it is reasonable to ignore multiple scattering i.e. once an X-ray has been scattered by an atom, we assume that it exits the sample without any further interactions. We can think of every single atom as being individually illuminated by a uniform beam of X-rays, and that in response each atom re-sends a fraction of the incoming energy uniformly in all directions with the same frequency and phase as the incoming waves. When these outgoing scattered signals arrive at the detector, for example a photographic film, they interfere with each other and appear strong in some places and weak in others. The task in X-ray crystallography is to deduce

from these macroscopic recordings the positions of the individual atoms within the scattering molecules.

In this modeling project, we consider a very simplified scenario. In particular

- We consider only the scatter from one single molecule (and ignore the additional interference patterns that come from the fact that many molecules are repeated within the crystal),
- We consider a 2-D rather than 3-D molecule, and record the scatter around the periphery of a circle surrounding it (i.e. 1-D 'line recording' rather than 2-D 'photographic plate-type' recording),
- We assume that we can record the phase angle (relative to the illuminating X-ray) all around the circle. This quantity varies slowly around the device—just like the intensity of the scattered rays—but cannot be experimentally recorded in a practical way.

The Nobel Prize in Physics for 1914 was awarded to M. von Laue and for 1915 to W.H. Bragg and W.L. Bragg (father and son; W.L. Bragg at age 25 was the youngest Nobel Prize winner in history) for their work on X-ray diffraction. These initial studies were of great significance in establishing the lattice arrangement of atoms in crystals, as well as the nature of X-rays. The most obvious interference patterns that are seen are due to the repetition of molecules within a crystal causing 'Bragg reflections' according to the directions of crystal planes. However, refinements were soon made which also allowed a study of the atomic arrangements within the individual molecules. The inability to experimentally detect phase angles was largely overcome in the 1950's by H. Hauptman and J. Karle, and rewarded with the 1985 Nobel Prize in Chemistry. Even before this, X-ray crystallography was a critical tool in deciphering many complicated molecules that are fundamental to molecular biology. Numerous Nobel Prizes have been awarded for such work. This includes L. Pauling (Chemistry, 1954), M. Perutz and J. Kendrew (Chemistry, 1962), F. Crick, J. Watson and M. Wilkins (Medicine, 1962—the discovery of the structure of DNA depended critically on X-ray work by R. Franklin), O. Hassle and D. Barton (Chemistry, 1969), W. Lipscomb (Chemistry, 1976), D. Hodgkin (Chemistry, 1982) and A. Klug (Chemistry, 1982).

List OK up t
1985 - check Klu
- also check mo
recently!

30.2. Model Problem.

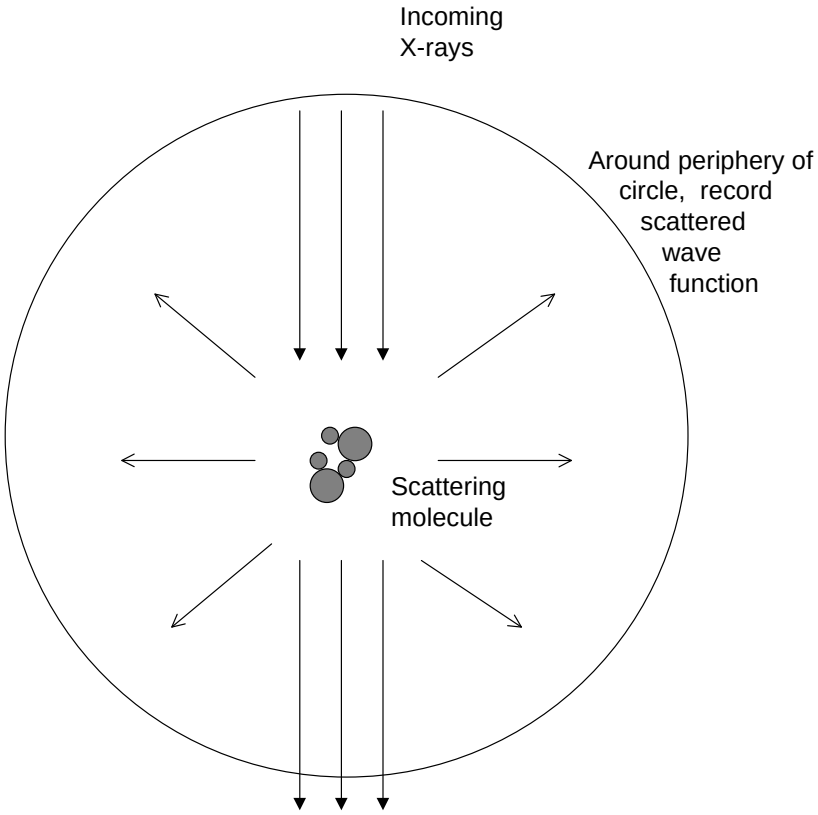


FIGURE 30.2.1. Schematic illustration of crystallography model problem.

Figure 30.2.1 illustrates the 2-D model -problem we will use. A number of atoms are located in a tight cluster (2-D molecule) near the center of an experimental device. Whenever an atom is illuminated, it re-transmits a fraction—determined by the scattering coefficient A—of the signal in equal strength in all directions.The modeling task is to develop codes which

(1) define a test molecule,
(2) simulate the scattering experiment and record the X-ray signal around the periphery of the unit, and
(3) from the scattered signal, recover the position of the atoms in the molecule

We will develop a general code for any number of atoms, and will apply it to the special case of a molecule consisting of four atoms:

List of atoms in model

Positions of atoms (in Å)		Scattering coeff. A
x-coord.	y-coord.	
0	-1	0.001
1	2	0.002
2	1	0.003
-1	1	0.004

We assume that this molecule is illuminated from above (along negative y-direction) with X-rays with *wave number* $\underline{k}_0 = (0, -5 \cdot 10^{10})$, i.e. the wave-function for the incoming rays is

$$\phi_{inc}(\underline{x}, t) = \phi_0 \, e^{i \, (\underline{k}_0 \cdot \underline{x} - \omega t)}. \tag{30.1}$$

30.3. Analytical technique for finding atomic positions.

We refer to Figure 30.3.1 and consider first the case that just one single atom, located at position **a** very near the center of the device, is illuminated with the incoming wave. The wave number vector $\mathbf{k}_0$ points in the direction of wave propagation, and is related to the wavelength through $\lambda \, \|\mathbf{k}_0\| = 2\pi$.

The scattered wave is recorded at the relatively distant location p. The incoming wave at the point **a** is given by () with **a** substituted for **x**. The wave function for the scattered wave at the point **p** then becomes

$$\phi_{sc}(\mathbf{p}, t) = \frac{\phi_0 A}{\|\mathbf{p} - \mathbf{a}\|} \, e^{\, i \, (\mathbf{k}_0 \cdot \mathbf{a} - \omega \, t)} \, e^{\, i \, \|\mathbf{k}_0\| \cdot \|\mathbf{p} - \mathbf{a}\|}. \tag{30.1}$$

Here we assumed that the amplitude of the scattered wave is the same in all directions, and that it decays proportionally to the distance traveled (the is case in 3-D; we use it also in 2-D). With minor approximations, this can be rewritten as

$$\phi_{sc}(\mathbf{p}, t) = \frac{\phi_0}{\|\mathbf{p}\|} \, e^{\, i \, (\|\mathbf{k}_0\| \cdot \underline{\|p\|} - \omega \, t)} \, \left[A \, e^{-i \, \mathbf{k} \cdot \mathbf{a}} \right] \tag{30.2}$$

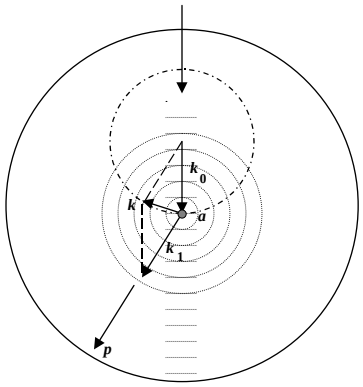


FIGURE 30.3.1. Illustration of an X-ray arriving from above and striking an atom at location $\mathbf{a}$ and being observed from a location $\mathbf{p}$. The figure also illustrates the vectors $\mathbf{k}_0$, $\mathbf{k}_1$ and $\mathbf{k}$ used in the discussion.

where $\mathbf{k} = \mathbf{k}_1 - \mathbf{k}_0$. This can be seen as follows:

Let $\mathbf{k}_1$ be a vector of the same length as $\mathbf{k}_0$ but in the direction of $\mathbf{p}-\mathbf{a}$ (cf. Figure 30.3.1). This vector $\mathbf{k}_1$ describes the wave number for the scattered wave received at $\mathbf{p}$. It satisfies

$$\frac{\mathbf{k}_1}{\|\mathbf{k}_0\|} = \frac{\mathbf{p}-\mathbf{a}}{\|\mathbf{p}-\mathbf{a}\|} \tag{30.3}$$

The difference $\mathbf{p} - \mathbf{a}$ appears twice in each of the equations (30.1) and(30.3). Recalling that $\|\mathbf{a}\|$ is very small in comparison to $\|\mathbf{p}\|$, we approximate $\mathbf{p}-\mathbf{a}$ with $\mathbf{p}$ everywhere, except in the exponent of (30.1) where $\|\mathbf{k}_0\|$ is large, and the phase information is critical. There we use $\|\mathbf{p}-\mathbf{a}\| \approx \|\mathbf{p}\| - \frac{\mathbf{k}_1\cdot\mathbf{a}}{\|\mathbf{k}_0\|in}$. To derive this, we note that

$$\|\mathbf{p}-\mathbf{a}\|^2 = \|\mathbf{p}\|^2 + \|\mathbf{a}\|^2 - 2\,\mathbf{p}\cdot\mathbf{a}.$$

Ignoring the quadratic term $\|\underline{a}\|^2$ gives

$$\begin{aligned}\|\mathbf{p}-\mathbf{a}\| &\approx \|\mathbf{p}\|\,(1-2\,\mathbf{p}\cdot\mathbf{a}\,/\,\|\mathbf{p}\|^2)^{1/2}\\ &\approx \|\mathbf{p}\|\,(1-\mathbf{p}\cdot\mathbf{a}\,/\,\|\mathbf{p}\|^2).\end{aligned}$$

We now substitute $\mathbf{p}\approx\mathbf{k}_1\,\|\underline{p}\|\,/\,\|\underline{k}_0\|$ (from (30.3)) to obtain $\phi_{sc}(\mathbf{p},t)\approx \frac{\phi_0 A}{\|\mathbf{p}\|}\,e^{\,i(\mathbf{k}_0\cdot\mathbf{a}-\omega\,t)}\cdot e^{\,i\,\|\mathbf{k}_0\|\cdot\|\mathbf{p}\|}\cdot e^{\,-\,i\,\mathbf{k}_1\cdot\mathbf{a}}$, which gives (30.2).

The purpose in rearranging (30.1) into (30.2) is that all the quantities associated with the atom with scattering coefficient A and position $\mathbf{a}$ now only appear in the last bracket.

Instead of one atom only, we consider next the whole group of N atoms which form the molecule. These are located at nearby positions $\mathbf{a}_n$ and have scattering coefficients $A_n,\ n=1,2,...,N$. Still assuming that secondary scatterings can be neglected, the wave received at $\mathbf{p}$ now becomes

$$\phi_{sc}(\mathbf{p},t)\approx\frac{\phi_0}{\|\mathbf{p}\|}\,e^{\,i\,(\|\mathbf{k}_0\|\cdot\|\mathbf{p}\|-\omega\,t)}\cdot\left[\sum_{n=1}^{N}A_n\cdot e^{-\,i\mathbf{k}\cdot\mathbf{a}_n}\right] \tag{30.4}$$

By recording ϕ_{sc} at all different positions $\mathbf{p}$ around the periphery, we have in fact recorded

$$\hat{A}(\mathbf{k})=\sum_{n=1}^{N}A_N\cdot e^{-\,i\,\mathbf{k}\cdot\mathbf{a}_n} \tag{30.5}$$

around the dash-dotted circle in Figure 30.3.1 (recalling that $\mathbf{k}=\mathbf{k}_1-\mathbf{k}_0$). Atoms are not quite point-like. It is better to think of A as a continuous function of position, i.e. to replace (30.5) by

$$\hat{A}(\mathbf{k})=\iint A(\mathbf{x})\,e^{-\,i\,\mathbf{k}\cdot\mathbf{x}}\;d\mathbf{x}=\iint A(x_1,x_2)\,e^{\,-\,i(k_1x_1+k_2x_2)}\,dx_1dx_2 \tag{30.6}$$

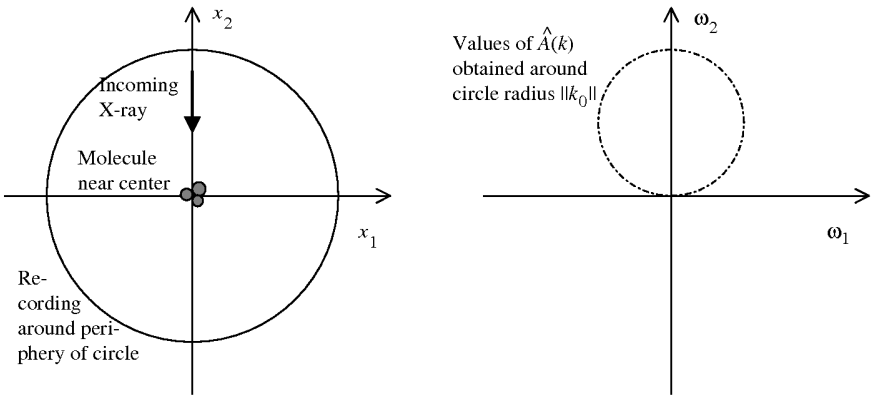


FIGURE 30.3.2. Illustration of the experimental layout in physical space and of the circle in Fourier space around which we get data for $\hat{A}(\omega_1, \omega_2)$.

We have arrived precisely at a standard 2-D Fourier transform. Therefore, it is natural to write $\mathbf{k} = (\omega_1, \omega_2)$ and to plot the physical $\mathbf{x} = (x_1, x_2)$ and Fourier $\mathbf{k} = (\omega_1, \omega_2)$ spaces beside each other (cf. Figure 30.3.2) rather than superimposed as in Figure 30.3.1.

Like in the FT method for tomography, we next appeal to the theorem which states that rotating a function in physical space rotates its Fourier transform with the same angle. Rotating the molecule around the center in the $\mathbf{x} = (x_1, x_2)$-plane rotates the circle in the $\mathbf{k} = (\omega_1, \omega_2)$-plane around the origin in that plane. This allows us to obtain data for $\hat{A}(\omega_1, \omega_2)$ throughout the interior of the domain $\|\mathbf{k}\| \leq 2\,\|\mathbf{k}_0\|$ in the $\mathbf{k} = (\omega_1, \omega_2)$-plane. According to (30.6) it remains then only to do a 2-D Fourier transform to obtain the function $A(x_1, x_2)$ describing the positions and types of the atoms in the molecule.

30.4. Computer implementation.

We implement here the method described in the previous section.

```
 % Main routine


atoms = {[} 0e-10, -1e-10, 1 ;... % Define 2-D molecule set-up:
 1e-10, 2e-10, 2 ;... % Give x,y (positions) and A
 2e-10, 1e-10, 3 ;... % (scattering coefficient) for
 -1e-10, 1e-10, 4 {]}; % each of a set of atoms


kz = {[}0 , -5.e10{]}; % Wavenumber for incoming X-ray
 n = 64; % Set discretization level around
 % periphery of recording device
n2 = 2{*}n; % Need to record at twice as many
 % angles of rotation for the molecule
ang = pi/n;

s = sin(ang); c = cos(ang); % Create a matrix that rotates the
rot = {[}c -s 0 ; s c 0 ; 0 0 1 {]}; % molecule in the array 'atoms'
 % when the matrix product ang*rot
 % is formed.
 expdata = zeros(n,n2); % Lay out array to receive experimental
 % data for all the n rotations of the
 % molecule
 for k = 1:n2 % Perform the experiment for 2*n angles
 expdata(:,k) = experiment (n, kz, atoms); atoms = atoms{*}rot ; % Rotate the molecul
end % Plot real part of complex wave
 % function as seen around the edge
 % of the circle surrounding the
 % sample
figure; pcolor(real(expdata)); shading interp
```

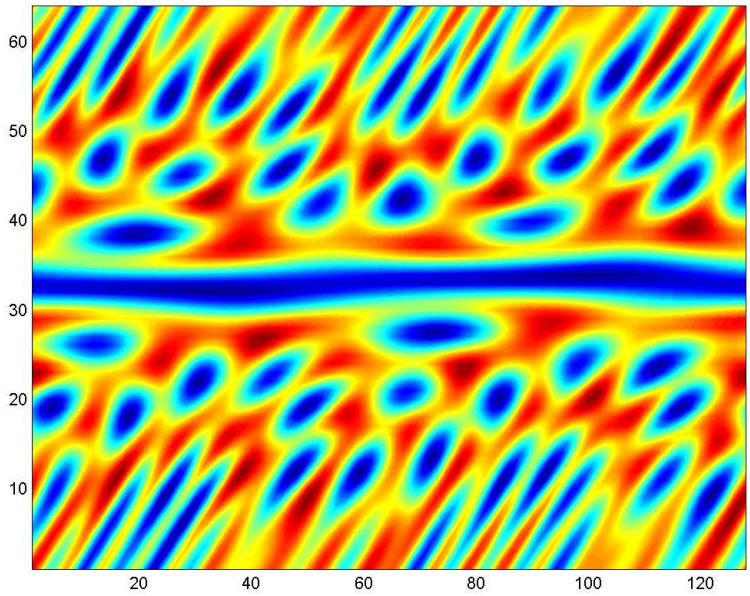


FIGURE 30.4.1. Real part of complex wave function as seen around the edge of the circle surrounding the sample.

Figure 30.4.1 shows graphically the real part of the matrix 'expdata' (of size 64×128 since $n = 64$)—the pattern in the imaginary part is similar. For each of the 128 columns, the molecule has been rotated by an additional angle of $\pi/64$. The data along a column represents the X-ray data $\phi_{sc}(\underline{p}, t)$ collected around the sample, i.e. according to the discussion in the preceding section, the function $\hat{A}(\underline{k})$ around a circle in the (ω_1, ω_2)-plane as shown in Figure .

All the experimental data is now collected. We need need next to re-arrange the data into polar r, θ - coordinates. A nice little geometric curiosity turns now out very helpful. Figure 30.4.2 illustrates this. The code section that utilizes this fact is slightly tricky, and we give it without detailed comments. It calls a routine 'trans' that is given following the main code.

```
pdata = zeros(n+1,n);
```

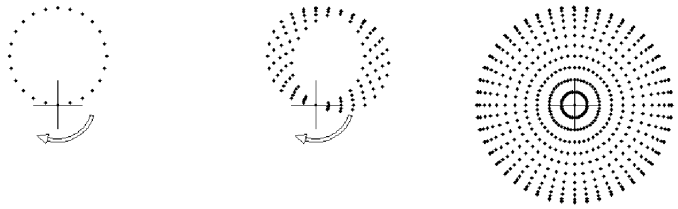


FIGURE 30.4.2. Illustration of how equispaced points on one circle line up if the circle is turned in equal steps around a point on its periphery. To be easier to read, this picture is based on $n = 24$ rather than $n = 64$ as used in the code.

```
% Move the recorded data over into polar form in k_space
% Loop over different rotations
for k=1:n2 % of the molecule.
 for m=1:n % Loop over n entries for each
 {[}i1,i2{]} = trans(m,k,n); % molecule rotation
 pdata(i1,i2) = pdata(i1,i2)+expdata(m,k); end end

pdata( 1,:) = pdata( 1,:){*}2; % Multiply two edge lines in pdata
pdata(n+1,:) = pdata(n+1,:){*}2; % by two since they have been updated
 % only once
pdata(:,n+1)=flipud(pdata(:,1)); % Extend pdata with one more column
 % to aid in the interpolation
 % Plot experimental data in polar-type
 % form. Each of the 64 columns corre-
 % spond to a direction across the
 % origin in Fourier space
figure; pcolor(real(pdata)); shading interp
```

The next task is to interpolate this polar-type data - laid out as shown in the right part of Figure 30.4.2 over into x, y-form. The next code segment does this:

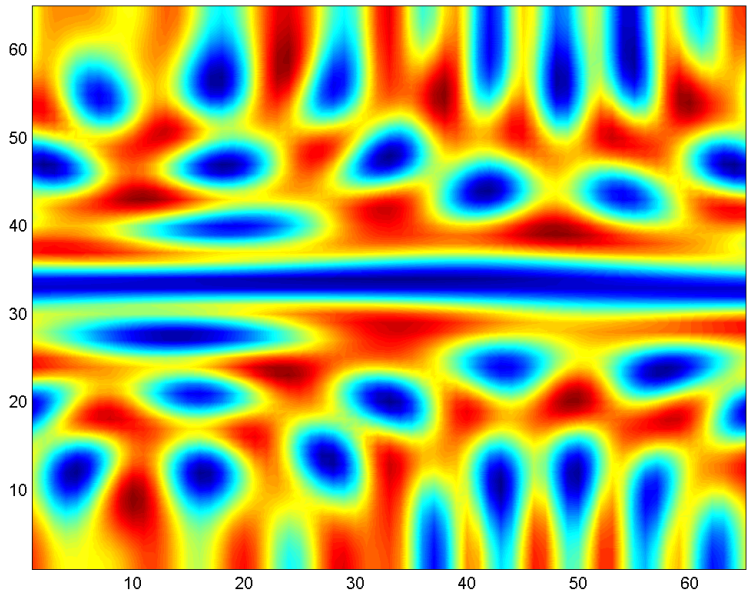


FIGURE 30.4.3. Experimental data turned into a polar-type form. Each of the 64 columns correspond to a direction right across the radial data pattern.

```
xv = linspace (-1,1,n+1); {[}x,y{]} = meshgrid(xv,-xv); {[}th,r{]}
= cart2pol(y,-x);

lg = th<0; % Make adjustment for our polar
r(lg) = -r(lg); % convention, -1<=r<=1 rather than
th(lg) = th(lg)+pi; % traditional 0<=r<=1
 s = interp2(linspace(0,pi,n+1),sin(linspace(-pi/2,pi/2,n+1)),pdata,
... th,r,'cubic'); % Cubic interpolation from polar
 % to Cartesian coordinates
 % Plot real part of the Fourier
 % transform of the molecule
 figure; pcolor(real(s)); shading interp; axis square
```

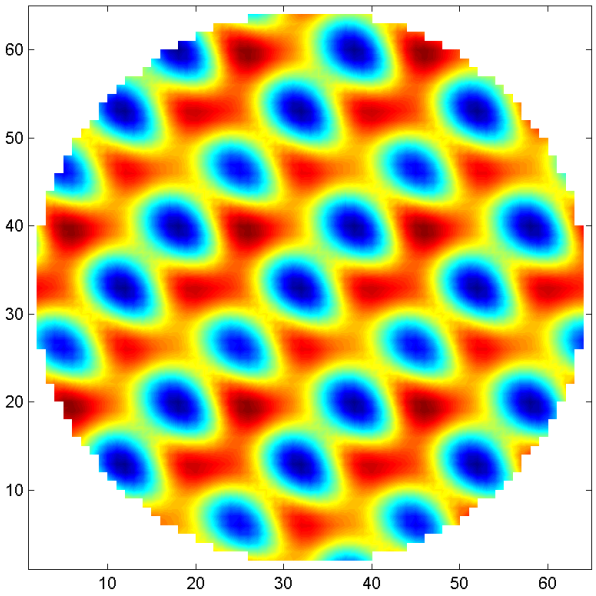


FIGURE 30.4.4. Scattering function $\hat{A}(\omega_1, \omega_2)$ (Fourier space version of image of molecule), as recovered from the experiment.

Figure 30.4.4 shows what $\hat{A}(\omega_1, \omega_2)$ now looks like. Next task is to perform the FFT to bring this function $\hat{A}(\omega_1, \omega_2)$ back to physical space. It will then display the continuous scattering function $A(x_1, x_2)$, i.e. it will show a picture of the molecule that we started with. We are making several approximations in this step. The Fourier data $\hat{A}(\omega_1, \omega_2)$ is known only inside a circle, and we simply set it to zero outside this circle. Our next, rather crude approximation is that we use an FFT which assumes periodic data whereas $\hat{A}(\omega_1, \omega_2)$ is not periodic. However, as Figure 30.4.4 shows, $\hat{A}(\omega_1, \omega_2)$ features a distinct 2-D periodic-looking pattern, so it is reasonable to hope that the inverse Fourier transform will still be able to pick up some main wave numbers. These will then correspond to atomic positions.

```
 s(isnan(s))=0; sr=abs(fft2(s)); sr=fftshift(sr);
sr=sr.' ;
```

```
% Plot the recovered image of the
% molecule (scattering function)
figure; colormap({[]0 0 0{]}); mesh(sr); axis off figure; colormap({[]0
0 0{]}); contour(sr); axis square
```

The two figures that were printed in the last two lines of code (Figures 30.4.5 and 30.4.6) show a near-perfect image of the molecule that we started with - correct atomic positions, and a good recovery also of the relative values of the scattering coefficients (thus allowing us to identify the types of the different atoms as well). In all the coding, we have been very careless with scaling constants, making both amplitudes and length scales qualitative and not quantitative. This can be corrected for - but would add some more detail to the code. In actual modeling, it is common to first try to obtain qualitative results, and then in a second stage fill in precise scalings. To keep the code - and comments - brief, we omit that here.

The main Matlab program given above called two functions, 'experiment' and 'trans'. They are given below :

```
function fip = experiment(n, kz, atoms);

fip = zeros(1,n); % Set up for the loop that will encompass the atoms.


magkz = sqrt(kz(1)2 + kz(2)2);

ang = linspace(0, 2 {*} pi, n + 1);

ang(end) = {[]{]}; x = sin (ang);

y = cos (ang);

s = size (atoms); noofatoms = s(1);
```

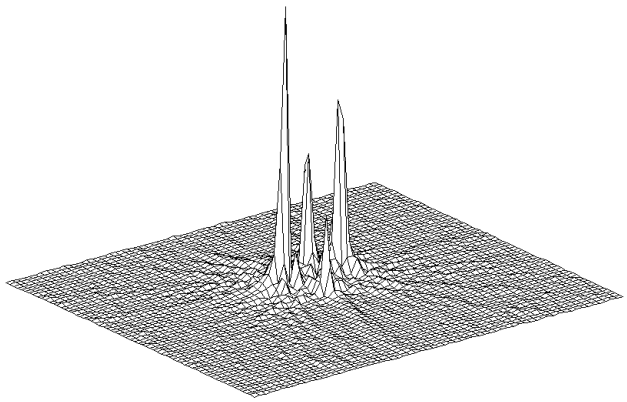


FIGURE 30.4.5. Recovered scattering function - heights and positions of peaks identify types of the the atoms and their positions within the molecule.

```
for k = 1: noofatoms; % Loop over atoms.


atpos = atoms(k , 1 : 2);

sccoeff = atoms(k , 3);

dist = sqrt((x - atpos(1)).2 + (y - atpos (2)).2);

fip = fip + sccoeff./sqrt(dist).{*} exp(i{*}(kz {*} atpos.' + magkz{*}dist));

end
```

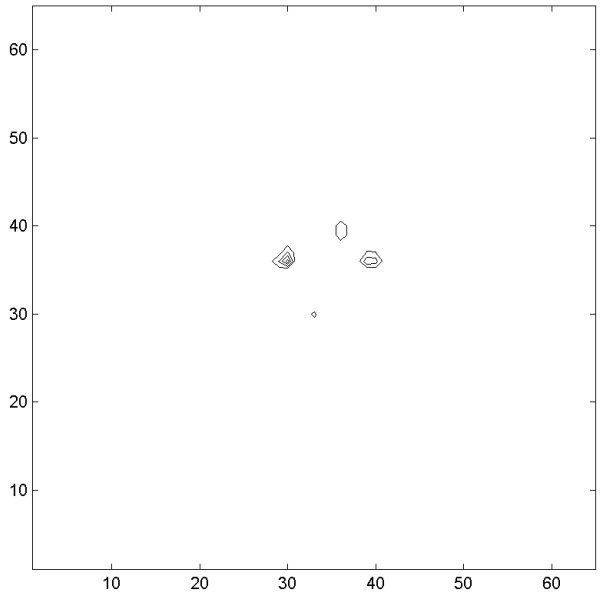


FIGURE 30.4.6. Contour display of the computed scattering function - illustrating the positions of the different atoms in the molecule.

```
fip = fip.' ;

function {[}i,j{]}=trans(i,j,n) % This routine assists in moving the


j=j+n/2+1-i; % scatter data from being available


i=n+2-i; % along successive circles to being


while j<1 % available on a (radially stretched)
```

```
j=j+n; % polar grid.


i=n+2-i;

end

while j>n

j=j-n;

i=n+2-i;

end
```

CHAPTER 31

SIGNATURE VERIFICATION

31.1. Introduction.

One might well ask whether automated signature verification systems are still relevant. After all, there is such a wide choice of personal biometric identification systems available that signatures might appear distinctly old fashioned. The best answer is, probably, by necessity. Commercial banks still process a huge number of checks, despite all attempts to move to a paperless, electronic environment. And of course, checks are endorsed by a signature. Only in exceptional cases are the signatures verified electronically. In this chapter we discuss the elements of a signature verification system.

Signatures on documents such as checks are known as 'static' signatures because the only information one has about the signature is in terms of an static image—the image on the document. There is another possibility and that is of capturing signatures by means of a digitizing tablet. Modern tablets such as the Wacom tablet, have the ability of capturing the signature as a parametrized curve—the x and y coordinates of the signature as a function of time, the pen pressure, the pen direction and the pen tilt. Since the values are sampled at a constant rate, about 125 times a second, it is straightforward to also calculate the pen speed or rhythm used to produce the signature. Thus one can think of a signature as an $m \times n$ array S where m denotes the number of features (x– and y coordinates, pen pressure, etc) and n the number of samples. Note that most of this *dynamic* information is hidden from any would-be forger and therefore hard to reproduce. It should be no surprise that dynamic signature verification systems based on digitizing tablets are much more accurate than their static counterparts.

Banks (still) need to process static signatures, and it has become quite common to capture signatures digitally if you use for example, your credit card. It is certainly

not easy to build a reliable signature verification system, mainly because of the large natural variation between genuine signatures. Signatures have much to offer, however. It is one of the best identification methods if one wants to endorse a transaction. It is very difficult afterwords to argue that the signature was created unintentionally! Signature verification is one of the best of the active verification systems where client cooperation is required. Another advantage is that signature data captured electronically suffers very little contamination—the systems are able to capture the data very accurately. This is one of the weaknesses of fingerprint system and probably the only weakness of iris verification systems. Signatures are also a well-accepted means of personal identification that has been in use for long time. Signature systems therefore tend to integrate easily with what has been in use for a long time.

Producing is signature is a probabilistic process, and therefore fits into the framework described in Part 4. Every time you sign, your signature looks slightly different, yet is still recognizable as your own. Somehow the variations between your signatures have structure that uniquely identify them as your own. It is explained in Chapter 29 how Hidden Markov Models can be used to model these variations, leading to efficient and robust signature verification systems. In this chapter a much simpler strategy is developed. The basic idea is to compare different instances of the same signal produced by a probabilistic process. The basic algorithm, known as Dijkstra's algorithm, employs the dynamic programming (DP) strategy in the sense that it solves the problem by combining the solutions of different sub-problems. Although obsolete as far as signature verification is concerned, it remains of considerable interest in its own right.

31.2. Capturing the Signature.

Commercial tablets such as the Wacom tablet, capture x– and y coordinates, pressure, pen direction and pen tilt as a function of time. Typically these values are recorded at fixed time intervals so that each signature is presented as an $5 \times n$ data array where n denotes the total number of samples. From the x– and y coordinates one can also calculate the rhythm defined as the array with components, $v_j = \sqrt{(x_j - x_{j-1})^2 + (y_j - y_{j-1})^2}, \quad j = 2, \ldots, n$. Since the tablet uses a constant

sampling rate, the rhythm differs from the actual pen speed by a constant. Letting $v_1 = 0$, for example, it may be added to the data array and in general a single signature is represented as an $m \times n$ array, where m is the number of variables we wish to compare.

Figure 31.2.1(a) shows the coordinates of (a part of) a signature as recorded by the tablet. Since these were obtained in sequence, it is natural to connect (interpolate) them to obtained a smooth-looking signature as in Figure 31.2.1(b). This is only possible because we know the order in which the coordinates were recorded.

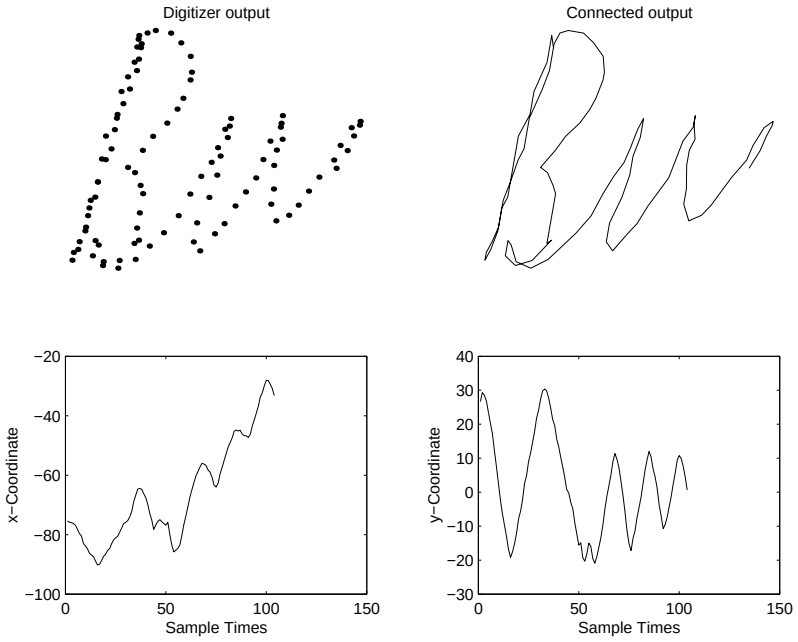


FIGURE 31.2.1. Signature captured by means of a digitizing tablet.

As pointed out in the Introduction, the data obtained from the digitizing tablet is not significantly contaminated by noise and can be used as-is—very little preprocessing is necessary, mainly to normalize the signatures. Because of the quantization offect of the tablet, it might be a good idea to apply a small amount of smoothing.

31.3. Pre-Processing.

One of the immediate difficulties facing a signature verification system is the fact that different signatures are captured at different angles, positions and even at different sizes. This may cause problems if we wish to compare the shape of the signatures. One possibility is to find a norm independent of these variables. Far easier is to transform the signatures to a standard size and orientation. This is illustrated in Figure 31.3.1. Although the signatory is the same person, the signatures differ in size, position and rotation. There are different ways of normalizing signatures, the results of the procedure described below is also shown in Figure 31.3.1.

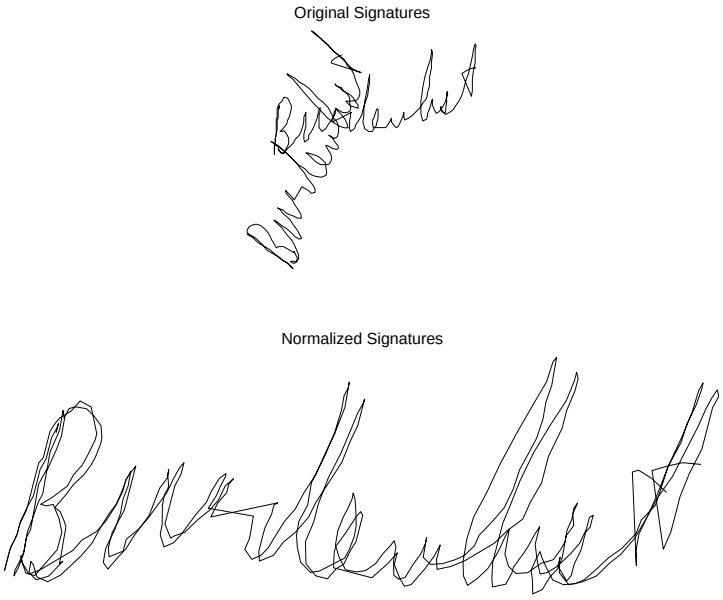


FIGURE 31.3.1. Different signatures of the same signatory.

The shape of a signature is described by its x and y coordinates, written as the $2 \times n$ dimensional array,

$$S = \begin{bmatrix} x_1 & \dots & x_n \\ y_1 & \dots & y_n \end{bmatrix} \tag{31.1}$$

The translational offset is corrected by removing the mean from the signature. Thus if

$$\mathbf{a} = \frac{1}{n}\sum_{j=1}^{n}\begin{bmatrix} x_j \\ y_j \end{bmatrix},$$

the signature is shifted to the origin by subtracting the mean from the original coordinates. The shifted signature is again an array S_0 of the form (31.1) but with zero mean. All that remains to be done is to correct for rotation and size. One elegant way of doing this is based on the SVD, described in Chapter 11. In that chapter we learn how to identify the direction of maximum variation as well as the magnitude of the variation in this direction. All we therefore have to do in order to normalize the signature with respect to rotation, is to rotate the signature so that its direction of maximum variation aligns with the x-axis. Since we have a measure of the magnitude of the variation, we can scale the signature so that this magnitude becomes a prescribed, fixed value. For details the reader is referred to Chapter 11.

This works quite well with most signatures as shown in Figure 31.3.1. Unfortunately for the odd exceptions the signature can be so close to being circular that no direction of maximum variation can be found. In these cases, in our experience the Radon transform discussed in Chapter 1 provides a viable alternative.

31.4. Feature Extraction.

Now we are getting to the heart of the verification problem: Extracting features from the signature that we can compare. Ideally we need features that are stable, i.e. don't change very much between different genuine signatures, and which are hard to forge.

Broadly speaking one can divide features into two classes, the shape based features and the 'hidden' features. Since the former is accessible to any observer, it is relatively easy to forge. It also does not contain any direct information about the individual behind the signature. Since the latter contains information that is not directly available and unique to the individual, it yields a more reliable means of verification.

31.4.1. Rhythm. By rhythm we simply mean the rhythm of the pen tracing out the signature. Few individuals are even aware of the fact that they sign with a distinctive rhythm. Rhythm is not easy to forge, try it! Although we extracted

and studied the rhythms of signatures captured on a digitizing tablet, it is no doubt much harder to forge than the shape of the signature.

Interpolating the data points (x_i, y_i), $i = 1, \ldots, N$, typically using a cubic spline, see Section 12.6, we obtain a parametric curve, $(x(t), y(t))$ representing the signature. The rhythm is then simply calculated as

$$v_i = \sqrt{(x_{i+1} - x_i)^2 + (y_{i+1} - y_i)^2}, \quad i = 1, \ldots, N-1.$$

Note that the rhythm is a scaled version of the actual pen speed.

31.4.2. Curvature. Since the signature is available as a parametric curve, $(x(t), y(t))$, $t \in [T_0, T_N]$, for some parametrization t, the 'velocity components' of the curve are $(\dot{x}, \dot{y})$. The angle α of the tangent with the x-axis may be written as,

$$\tan(\alpha) = \frac{\dot{y}}{\dot{x}}.$$

Note that α is uniquely determined by the signs of $\dot{x}$ and $\dot{y}$, e.g. for $\dot{x} < 0$ and $\dot{y} > 0$ it follows that $\pi/2 < \alpha < \pi$.

It is sometimes useful to be able to change the *parametrization*, $t = t(s)$, of the curve while keeping the curve itself intact. In terms of the new parameter the curve is given by $(u(s), v(s)) := (x(t(s), y(t(s))$. The orientation of the curve is preserved by requiring $dt/ds > 0$. A particularly useful parametrization is the arc-length, defined independently of the particular parametrization by

$$s(t) = \int_{t_0}^{t} ds$$

where $ds = \sqrt{dx^2 + dy^2}$ is an 'element of length' along the curve. The length of the curve between two points A and B is then given by,

This enables us to introduce the 'curvature' of the curve. As we move along a curve, its inclination α with the x-axis will change at a definite rate per unit arc length traversed—the sharper the bend, the higher the rate of change of α. Accordingly *curvature* is defined as

$$\rho = \frac{d\alpha}{ds}.$$

This provides a very convenient local characterization of a curve, independent of its parametrization. It becomes the second feature that we shall use in the comparison of signatures. Note that the curvature does not contain any 'hidden' biometric information, it informs us only about the *shape* of the signature.

A final note a caution: Both speed and curvature need to be calculated numerically. Thus we need to calculate first and second derivatives numerically, a notoriously unstable process. This is a good reason to smooth the signature to get rid of the discretization errors.

31.5. Comparison of Features, Dijkstra's Algorithms.

In practice we are often faced with a situation where it is necessary to compare different *signals* of different length. For instance, if one is interested in identifying a word or a phrase in a speech recognition application, one realizes that different speakers utter the same phrase in different ways, and with different durations. Comparing these signals is exactly the same problem we face in a signature verification problem. Again we have two similar signals, with differences in details and of different durations to compare. The algorithm we are about to describe was therefore extensive used in speech processing applications until it was largely replaced by even more powerful algorithms, the so-called Hidden Markov Models. It should become clear that there is a straightforward generalization to vector-valued signals; for the moment we keep things as simple as possible and concentrate on single signals. The basic idea is to first define a suitable 'cost function' that gives an indication of the difference (total cost) between the signals. One then proceeds to find a map that maps one signal onto the other in such a way that the cost function is minimized. In this sense one therefore finds the best possible match between the two signals, taking into account that they are usually very different. One can then relate the total 'cost' (difference between the two signals) to a confidence value—the lower the cost, the higher the confidence that the two signals are actually different renderings of the same entity.

It might be interesting to note that the algorithms is much more general than just for comparing signals. It is a powerful algorithm for calculating maps minimizing an appropriate cost function. For example, the line-break, paragraph-blocking algorithm

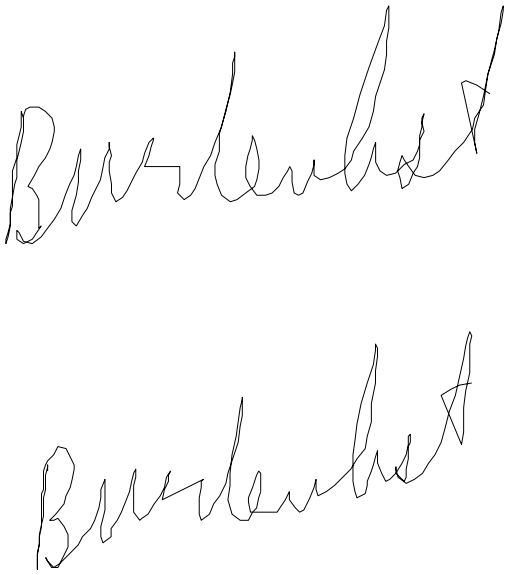


FIGURE 31.5.1. Two different signatures by the same signatory for comparison.

used by TEX is based on this algorithm. The naïve approach is to estimate the number number of words that will fit into a line, given the page width, and then calculate the inter-word distances so that the line is of exactly the specified length. The problem with this algorithm is that the result does not look good. Sometimes it is better to move a word to the next line for improved overall appearance, etc. Thus the best appearance is achieved by analyzing a whole paragraph and not simply each line separately. The way Knuth [**?**] does this is to define a suitable cost function that is constructed with overall appearance in mind. A mapping is then calculated that fills a whole paragraph in block form.

Let us have a closer look at the problem. In Figure 31.5.1 we show two different signatures by the same person with their y coordinates shown in Figure 31.5.2. It should be clear that the natural way of comparing these signals is to find matching points on the two signatures, such as the peaks indicated in Figure 31.5.2. This clearly requires a nonlinear stretching or 'warping' of the signatures to best fit each other. Mathematically this amounts to a *re-parametrization* of the two signatures.

Given the two discrete signals, $\mathbf{y}_1 := \{y1(t), \quad t = 1, \ldots, L_1\}$ and $\mathbf{y}_2 := \{y_2(s), \quad s = 1, \ldots, L_2\}$, the problem is to find re-parametrizations $p(w)$ and $q(w)$ such that one

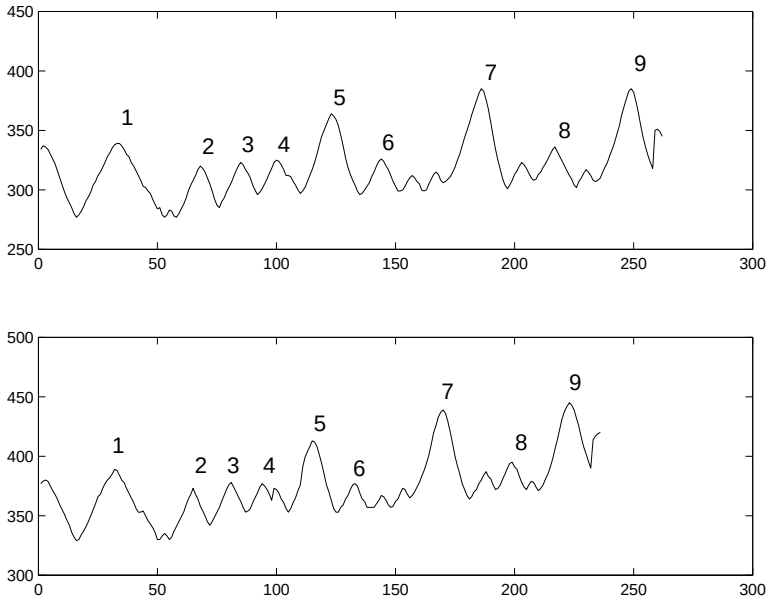


FIGURE 31.5.2. The y coordinates of the two signatures.

can identify $y1(p(w))$ with $y_2(q(w)), \quad w = 1, \ldots, L$ in such a way that the difference between the two resulting function is minimized. In order for $p(w)$ and $q(w)$ to be proper re-normalizations they need to be *non-decreasing* functions of w. Since the value of L depends on y_1 and y_2 it is determined as part of the algorithm.

Let us now become very specific and assume that the two signals we want to compare are given by,

$$\begin{aligned} \mathbf{y}_1 &= [1 \;\; 0 \;\; 1 \;\; 2 \;\; 1] \\ \mathbf{y}_2 &= [1 \;\; 0 \;\; 2 \;\; 1]. \end{aligned}$$

Since we want to compare each value of $\mathbf{y}_1$ with each value of $\mathbf{y}_2$ it is convenient to draw a grid as shown in Figure 31.5.3. If we now draw a curve consisting of straight lines connecting the lower left-hand corner with the upper right-hand corner (the reader may wish to have a peak at Figure 31.5.7) below, any such curve defines a mapping between the two signals. It is important to note however that the monotonicity constraint on $p(w)$ and $q(w)$ implies that grid-point (i, j) can only be

reached from either $(i-1,j)$, $(i,j-1)$ or $(i-1,j-1)$, as indicated (if there is a tie, we give preference to the diagonal). It is this curve that we are after, constructed in such a way that the two signals are aligned in the best possible way (in a sense that has to be made precise). Also note that our assumption that the curve begins and ends at the two corners implies that we match the start– and endpoints of the two signals. In some applications it is useful to relax this constraint—it is absolutely straightforward to relax the endpoint matching constraint, as will become clear.

The brute-force way of solving the problem is to investigate all possible paths connecting the lower left-hand corner with the upper right-hand corner. This is a use number and the reader may find it a fun exercise to derive the following formula for the total number of possible paths

$$\mathcal{N} = \sum_{s=0}^{\min(L_1,L_2)} \frac{(L_1+L_2-s)!}{(L_1-s)!(L_2-s)!s!},$$

constrained only by our monotonicity requirement. This is an enormous number, dominated by the first term which counts only the number of paths not containing a diagonal,

$$(L_1+L_2)!/L_1!L_2!.$$

Even this is too large to investigate exhaustively by computer for all but the smallest number of samples. We clearly need to do better.

All the most efficient algorithms are based on a very simple observation (which the reader can prove for herself): Each sub-path of an optimal path, is also optimal. This means that the global optimal path is pieced together from local optimal paths—exactly the defining strategy of Dynamic Programming (DP). The key is to define a local distance measure (local cost function) that measures the difference between the two functions. An obvious choice is

$$C_{i,j} = d(y1(i), y2(j)) := |y_1(i) - y_2(j)|, \quad i = 1,\ldots,L_1; \quad j = 1,\ldots,L_2. \tag{31.1}$$

For our example, this is written as

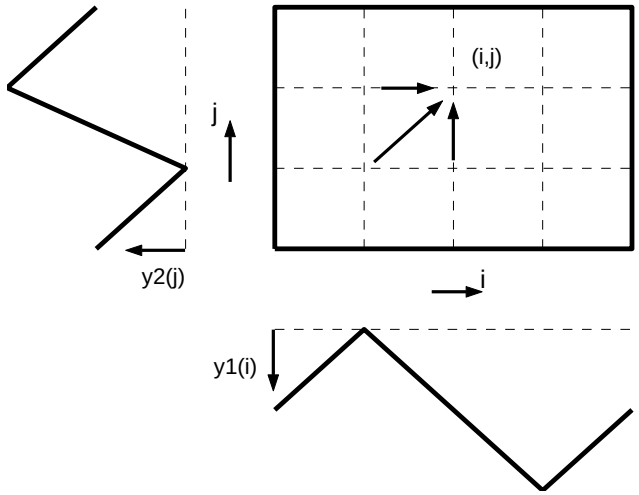


FIGURE 31.5.3. The two signals to be compared.

$$C = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 2 & 1 & 0 & 1 \\ 1 & 0 & 1 & 2 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix},$$

where one should note that the ordering corresponds to the ordering of the grid of Figure 31.5.4. Also note that in Figure 31.5.4 that it is convenient for us to number the grid points consecutively from 1 to 20. For example, grid point 9 has coordinates $(4, 2)$ and its local distance value is $C_{4,2} = 2$. Based on this local distance measure, the total cost function for the two signals $y_1(t)$ becomes

$$C = \sum_{w=1}^{L} |y_1(p(w)) - y_2(q(w))| \,. \tag{31.2}$$

Once the the local costs have been calculated, the rest of the algorithm proceeds without any reference to the original signals—it allows us to calculate the optimal path from the lower left-hand corner to every other point on the grid. This may sound wasteful but with the slight modification explained below, it is still the most efficient algorithm known. The result is shown in Figure 31.5.5. The Figure shows

the optimal path to each grid point with the total cost of each path. For example, the cost of reaching point 19 with coordinates $(4, 4)$, is 2. In order find its optimal path, we start at point and then backtrack until we reach point 1. Note that all we need is to know from where a specific point is reached, i.e. point 19 is reached from point 13, is reached from point 7, etc. Accordingly, we keep track of all the different paths, by defining an array $I(i), i = 1, \ldots, L_1L_2$ where $I(k)$ denotes the point from which point k is reached. For our example, Figure 31.5.5, we find that $I(20) = 14, I(19) = 13$, etc. Thus the optimal path is obtained from $I(20) = 14,\ I(14) = 8,\ I(8) = 7,\ I(7) = 1$.

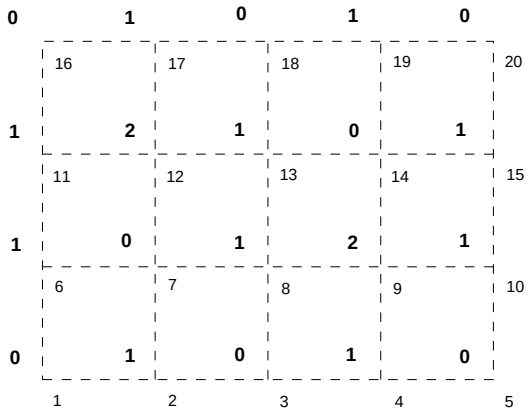


FIGURE 31.5.4. The local cost grid.

Let us proceed with the description of the algorithm. Since the local cost for point 1 is zero (see Figure 31.5.4), the total cost to reach grid point 1 is $TC(1) = C_1 1 = 0$. Noting that point 2 can only be reached from point 1, there is no decision to make and the total cost of reaching point 2 is: $TC(2) = TC(1) + C_{21} = 1$. We also record that it is reached via the optimal (and only) path from point 1, i.e. we set $I(2) = 1$. For points 3, 4, 5 and 6 we do the same, $TC(3) = TC(2) + C_{31} = 1, \quad I(3) = 2, \quad TC(4 = TC(3 + C_{41} = 2, \quad I(4) = 3, \quad TC(5 = TC(4 + C_{51} = 2, \quad I(5) = 4, \quad TC(6 = TC(1 + C_{12} = 1, \quad I(6) = 1$. However, point 7 may be reached from either points 1, 2 or 6. We already know the optimal paths to points 1, 2 and 6. Since the cost of the path to point 1, is less or equal to the cost to points 2 or 6, we reach point 7 via point 1 and set $TC(7) = TC(1) + C_{22} = 0, \quad I(7) = 1$. Now we do the same for the rest of the points on the grid:

$$
\begin{aligned}
TC(8) &= TC(7) + C_{32} = 1, \quad I(8) = 7 \\
TC(9) &= TC(3) + C_{42} = 3, \quad I(9) = 3 \\
TC(10) &= TC(4) + C_{52} = 3, \quad I(10) = 4 \\
TC(11) &= TC(6) + C_{13} = 2, \quad I(11) = 6 \\
TC(12) &= TC(6) + C_{23} = 2, \quad I(12) = 6 \\
TC(13) &= TC(7) + C_{33} = 1, \quad I(13) = 7 \\
TC(14) &= TC(8) + C_{43} = 1, \quad I(14) = 8 \\
TC(15) &= TC(14) + C_{53} = 2, \quad I(15) = 14 \\
TC(16) &= TC(11) + C_{14} = 2, \quad I(16) = 11 \\
TC(17) &= TC(11) + C_{24} = 3, \quad I(17) = 11 \\
TC(18) &= TC(13) + C_{34} = 1, \quad I(18) = 13 \\
TC(19) &= TC(13) + C_{44} = 2, \quad I(19) = 13 \\
TC(20) &= TC(14) + C_{54} = 1, \quad I(20) = 14.
\end{aligned}
$$

With the help of I it is now a simple matter of finding the optimal path as pointed out above, and shown in Figure 31.5.6, with its total cost $TC(20) = 1$.

The computational cost of this algorithm amounts to three tests at each grid point, i.e. it is of $O(L_1 L_2)$. This can be further reduced by realizing that the optimal path should not stray too far from the diagonal, at least not for similar signals. One can therefore restrict the search to a band around the diagonal, further reducing the computational cost. If we restrict the search to a band of width d, the computational cost is of $O(L_1)$ with a possibly large constant, depending on d (assuming $L_1 \geq L_2$).

Figure 31.5.7(a) shows the matching of the y coordinates of different signatures belonging to the same person. It is interesting to note how close the warping function remains to the diagonal, an indication that there is a good correspondence between the two functions. If one now plots y_1 and y_2 against their re-parametrizes indexes, Figure 31.5.7(b) shows how the different peaks are now perfectly aligned. Incidentally, the total cost in this case is 374.1.

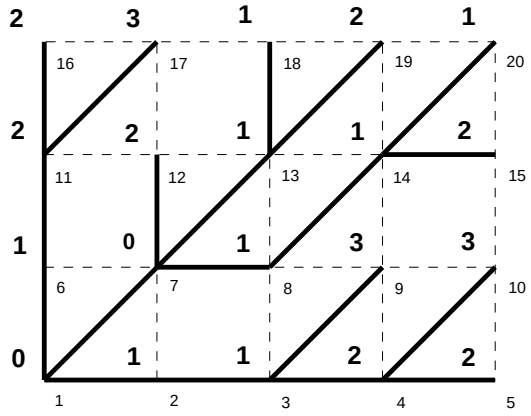


FIGURE 31.5.5. The optimal paths to all the grid points.

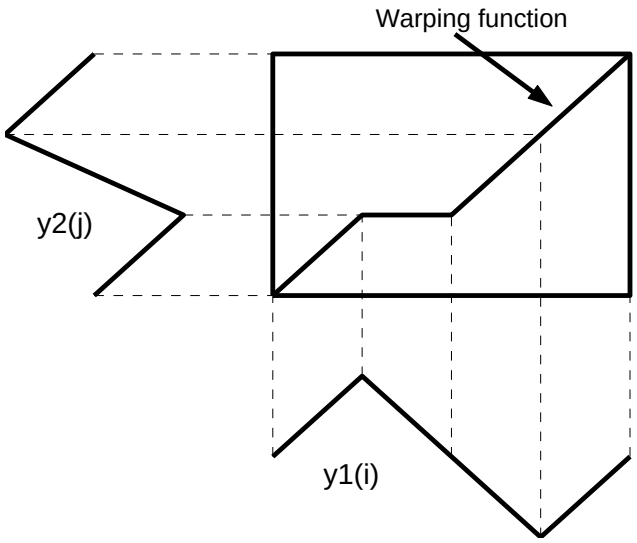


FIGURE 31.5.6. The optimal path.

Let us now do the same thing for two completely different signatures as shown in Figure 31.5.8. One may think of the second signature as a *casual* forgery since the forger clearly had no idea what the original looked like. The point is that the algorithm still finds the best possible match, which in this case is not good at all.

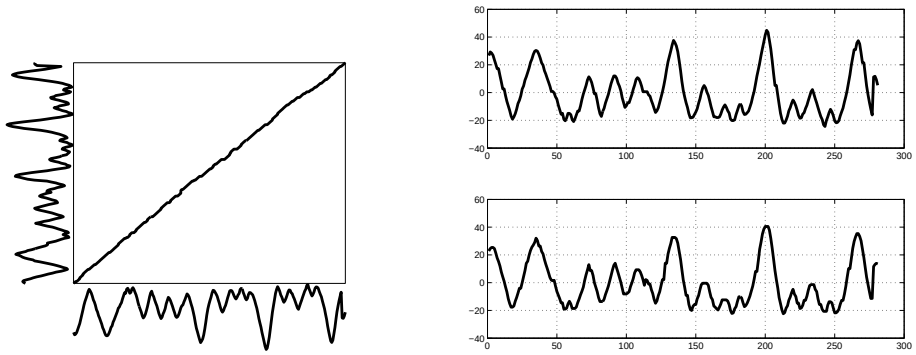


FIGURE 31.5.7. Matching two similar signatures.(a) The warping path. (b) The two signals aligned.

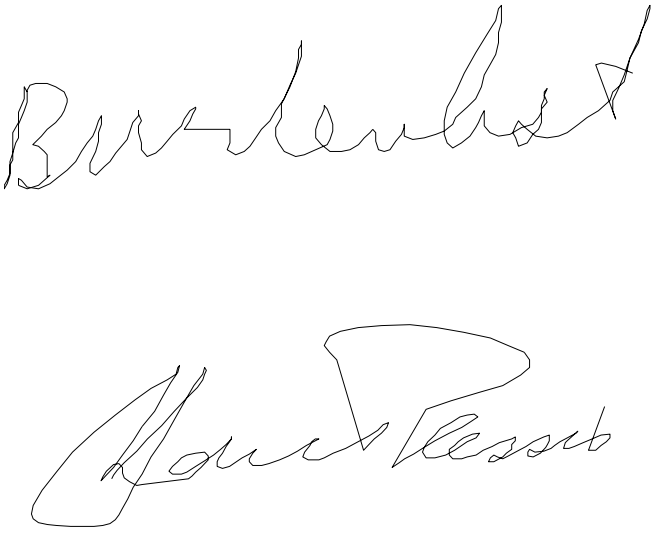
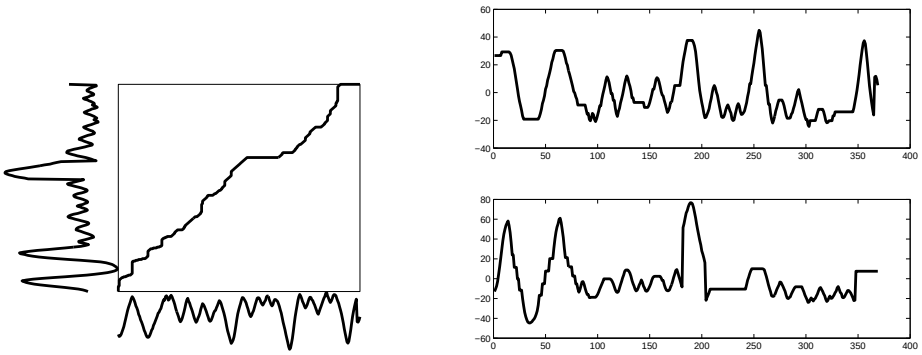


FIGURE 31.5.8. Two unrelated signatures.

The y coordinates, their warping function as well as their best possible alignment are shown in Figure 31.5.9. Although the algorithm still finds the best match, the warping function deviates considerably from the diagonal. The big difference between the two signatures is reflected by the large value of its total cost function, 2 885.



(a) (b)

FIGURE 31.5.9. Matching two unrelated signatures.(a) The warping path. (b) The two signals aligned.

31.6. Example.

We end this chapter with a final example. Figure 31.6.2 shows 10 signatures, the first one in the box is a genuine signatures and the idea is to spot the five forgeries from among the rest of the signatures. The forgeries were obtained by tracing genuine signatures—not the ones shown here. The results are summarized in Figure 31.6.1. Despite a considerable natural variation between the genuine signatures, the system achieved a definite separation between genuine signatures and forgeries. This was possible because we also compared the dynamic 'hidden' variables such as rhythm, pressure, etc., remained consistent between different genuine signatures. The answer is given in Figure 31.6.3 where the genuine signatures are boxed.

As confirmation we give the matching values of the signatures with a genuine signature in the database in Figure 31.6.1

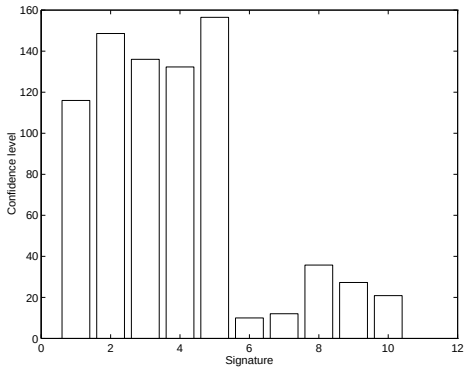


FIGURE 31.6.1. Matching values with a genuine signature as calculated by the system.

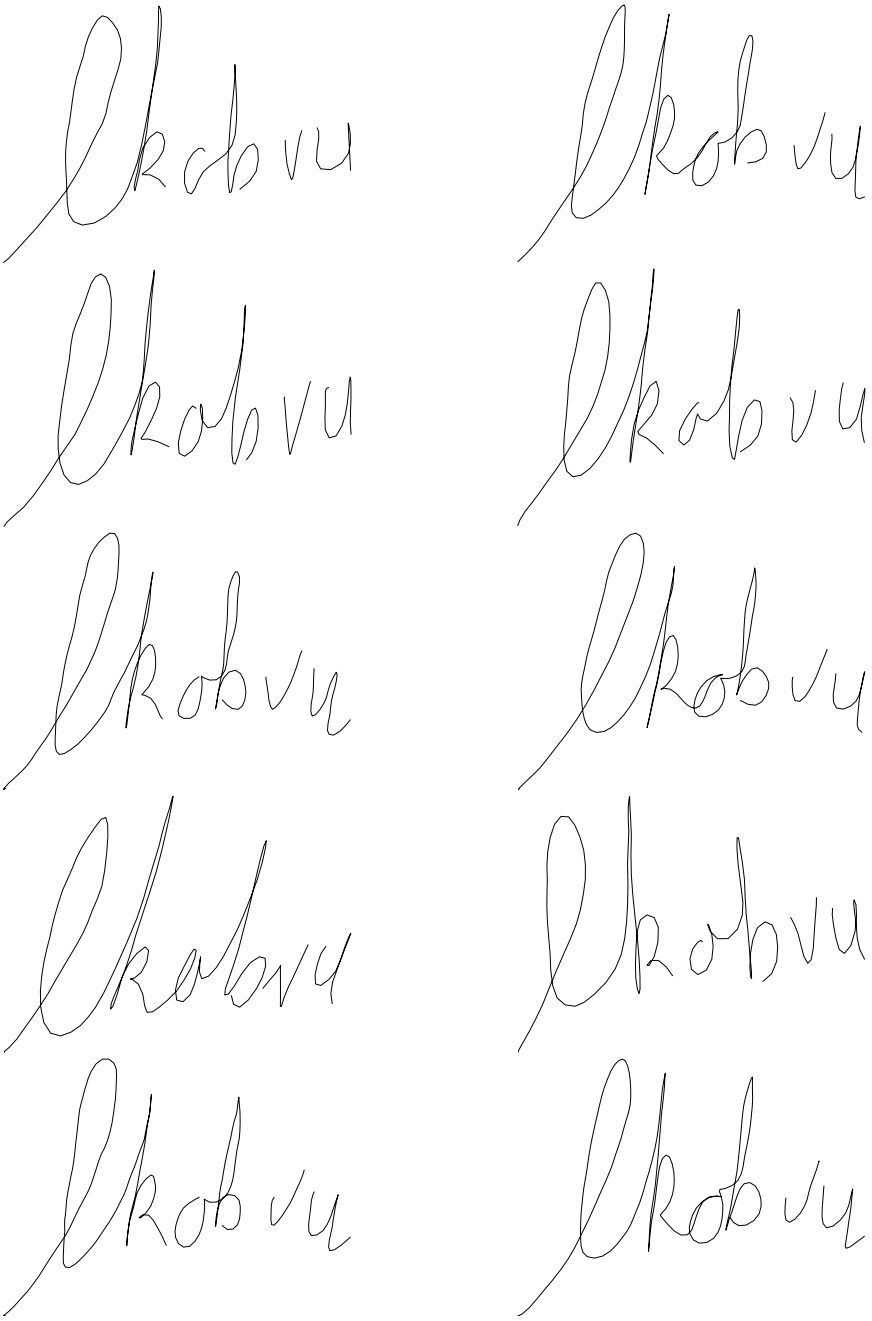


FIGURE 31.6.2. Can you spot the forgeries? (The first signature is genuine.)

FIGURE 31.6.3. The genuine signatures are boxed.

CHAPTER 32

STRUCTURE-FROM-MOTION

32.1. Introduction

At least some of the difficulties mentioned in the chapter on facial recognition are a direct consequence of working with projections of faces onto a 2D imaging medium. Note that in such images all information about the facial structure is indirect: Light interacts with the face, and this interaction is captured on film or other imaging media. Of course, the final image reflects the facial structure, as is evident from the (partial) success in reconstructing 3D objects from their gray-scale images, but the final image remains highly dependent on the position and strength of the light source. Why then not directly work with a 3D image? It is possible to represent a 3D structure with a gray-scale image, where the shades of gray have a direct geometric meaning. This permits us to use 2D techniques but sidestep the illumination problem—we still work with a gray-scale image, but now the different shades of gray describe structure.

Although this makes perfect sense from a theoretical point of view, the most serious practical problem is obtaining the 3D image. Various approaches have been tried. The first is the shape-from-shading method in which a 3D reconstruction is attempted from a 2D gray-scale image. This problem is ill-posed since many images give the same 2D projections, forcing one to impose external restrictions on the reconstruction. Another method imitates the human visual system and attempts a 3D reconstruction from stereo images. Parenthetically, is it known to what extent the human recognition system depends on its stereo vision, for example, under adverse lighting situations? Another closely related method involves the 3D reconstruction from a stream of video images. Mathematically the reconstruction is simple; for stereo images little more than high school geometry is required and for reconstructions from streams of video images SVD suffices (provided that a simple camera

model is used as explained below). The difficulty lies in the image processing—a large number of features must be located very accurately in all the images, in order to compute a *disparity* map, i.e. the relative offset of the individual features from one frame to the next.

Since large parts of the face are practically featureless, it is clear that the problem is a challenging one. Note the ease with which this is accomplished by the human mind—the same laws that determine automated 3D reconstruction with a computer also apply to the human visual system. In particular, the human visual system also requires a disparity map and evidence supports the believe that the human mind point-wise matches the images captured on the retinas of our two eyes. The computational complexity is staggering, see [**?**].

Of course, even 3D reconstruction does not address the problem of facial expression or the presence of a beard, mustache or glasses.

Another application involving 3D reconstructions from video sequences is in the movie industry. The problem is to seamlessly insert computer graphics in a real scene captured on film. The computer graphics need to become part of the scene itself. For that a 3D reconstruction of the scene is necessary. The idea is to do a 3D reconstruction that allows one to insert the appropriate graphics as part of the 3D scenery, with a much more realistic end result.

In this chapter we explore a simple reconstruction, just using the SVD. This is possible if we use an *orthographic* camera model.

32.2. Orthographic camera model.

It is well-known that 3D objects can be reconstructed from stereo pairs of 2D images. For example, the human 3D vision system relies heavily on the presence of *two* eyes. 3D information is also present in a *sequence* of images, such as a video film or indeed, human vision with one eye closed. Imagine that a person moves about and that we capture views of the face from different angles. These views certainly contain 3D information. The problem is to reconstruct a 3D face from these images.

In general the idea is to identify a number of features on the object that can be tracked over the different video frames. These features should also be selected in such a manner that the object can be reconstructed from them. There are different

ways of selecting and tracking features, a particularly useful one is due to Lucas and Kanade [**?**, **?**].

A face is a complicated 3D object requiring a large number of feature points for its reconstruction. In addition, it also contains large areas such as the forehead where features are difficult to identify. Although not an impossible task (see for example [**?**]), it is much simpler to illustrate the main ideas through simple geometric objects.

The problem we have to solve is an inverse one—from the projections on the video film we need to reconstruct the original object. For this we need to know how the projection was created in the first place, i.e. we need to have a model of the camera. A reasonable model is the so-called pinhole camera with its perspective projection, see for example [**?**]. Although it is possible to do a reconstruction using this camera model, the problem becomes nonlinear, requiring advanced techniques such as the Kalman filter or, more precisely, one of its nonlinear variants, see [**?**]. Fortunately Kanade and co-workers [**?**, **?**, **?**, **?**, **?**] developed a useful simplification, requiring only the techniques discussed in this paper.

Imagine objects far from the camera or a focal length approaches infinity. In that case, to a good approximation, the object is projected onto the film parallel to the Z– or optical axis as shown in Figure 32.2.1, also known as an orthographic projection.

We'll see in a moment that this simple camera model allows one to solve the problem using linear methods, in essence, by using a matrix factorization (see [**?**]).

Assume that we observe only one object and, importantly, that the object is a rigid body. Choosing a reference frame (x, y, z) fixed to the object, we describe the object in terms of n feature points

$$\mathbf{p}_s = \begin{bmatrix} x_s \\ y_s \\ z_s \end{bmatrix}, \quad s = 1, \ldots, n. \tag{32.1}$$

The shape matrix, describing the shape of the object in the object coordinate system is therefore written as

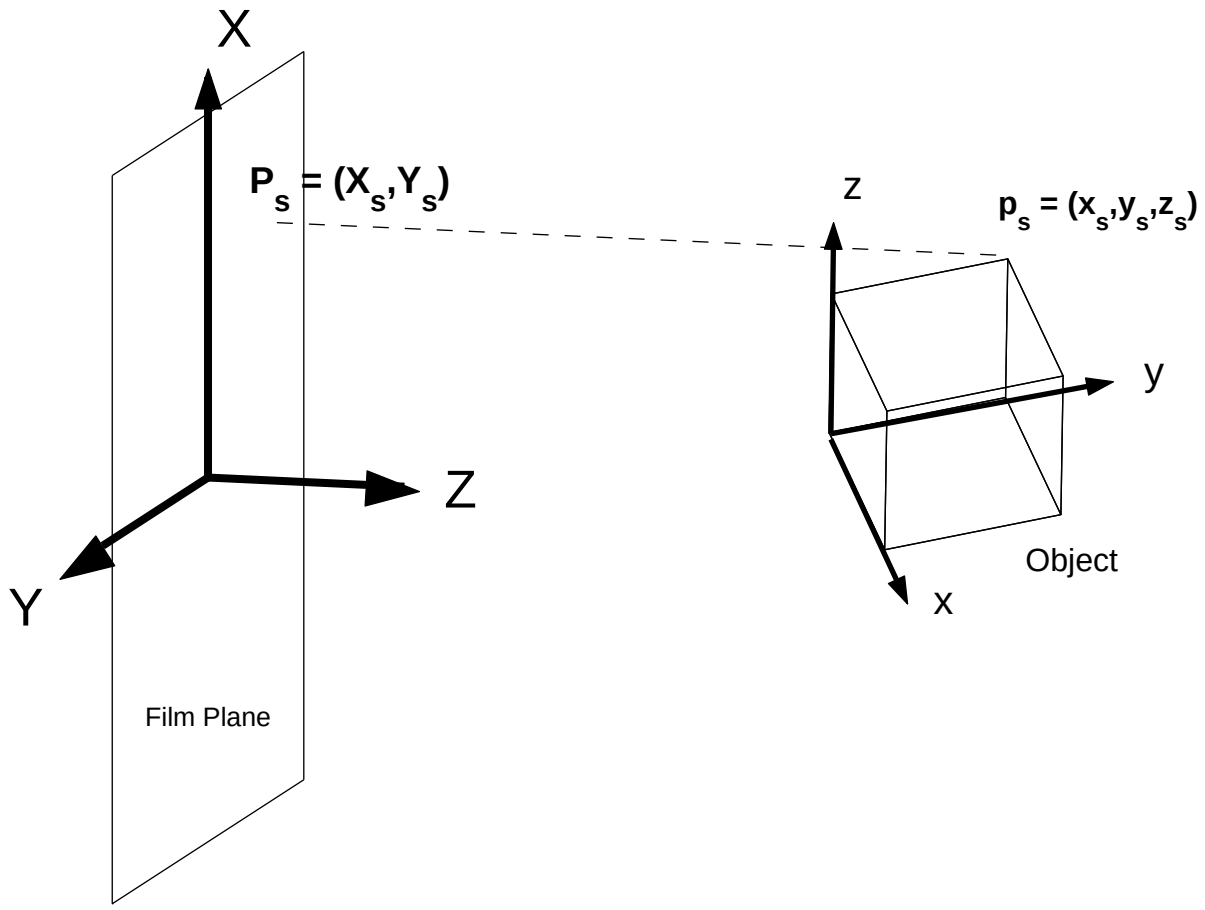


FIGURE 32.2.1. Orthographic projection.

$$S = \left[\begin{array}{cccc} \mathbf{p}_1 & \mathbf{p}_2 & \dots & \mathbf{p}_n \end{array} \right]. \tag{32.2}$$

The feature points are really all we know about the object and for a full reconstruction some kind of interpolation or subdivision is required. In the case of a wire-frame

cube as in Figure 32.2.1, the eight vertexes are natural features, allowing a perfect reconstruction.

We now allow the object to rotate; in Section 32.4 translation is handled similarly, using so-called homogeneous coordinates. The feature points of a rigid body do not change relative to a coordinate system fixed to the object. Now, imagine that the object moves with respect to another coordinate system fixed to the camera. Each feature point is projected onto the film plane of the camera and as the object moves, the features points are projected onto slightly different positions in consecutive frames of the video sequence. The information about the 3D structure of the object is contained in the offsets of the features points in the different frames.

Consider one of the feature points, $\mathbf{p}_s$, of the object, represented in the *object coordinates* fixed to the object as given by (32.1). If the rotation of the object coordinate system with respect to that of the camera at the time of the t-th video frame is given by the rotation matrix,

$$Rt = \begin{bmatrix} \mathbf{i}_t^T \\ \mathbf{j}_t^T \\ \mathbf{k}_t^T \end{bmatrix} \tag{32.3}$$

where

$$\begin{bmatrix} \mathbf{i}_t^T \\ \mathbf{j}_t^T \\ \mathbf{k}_t^T \end{bmatrix} = \begin{bmatrix} i_{xt} & i_{yt} & i_{zt} \\ j_{xt} & j_{yt} & j_{zt} \\ k_{xt} & k_{yt} & k_{zt}, \end{bmatrix}$$

then the feature point $\mathbf{p}_s$ is given in the camera coordinate system by $\mathbf{P_{ts}}^c$ where

$$\mathbf{P}_{ts}^c = R_t \mathbf{p}_s. \tag{32.4}$$

A little later we'll exploit the fact that rotation matrices are orthonormal, i.e. $R_t^T R_t = I$. This means that the rows (and columns) are orthogonal and normalized. In particular, we'll make use of

$$\mathbf{i}_t^T \mathbf{i}_t = 1 = \mathbf{j}_t^T \mathbf{j}_t \tag{32.5}$$

$$\mathbf{i}_t^T \mathbf{j}_t = 0 \tag{32.6}$$

Since our simple camera projects objects onto the film by translates parallel to the Z-axis of the camera coordinate system, a point $\mathbf{P}_{ts}^c = [X_{ts}, Y_{ts}, Z_{ts}]^T$ in camera coordinates projects onto $\mathbf{P}_{ts} = [X_{ts}, Y_{ts}]^T$ onto the film. This can be written as

$$\mathbf{P}_{ts} = O_Z \mathbf{P}_{ts}^c = O_Z R_t \mathbf{p}_s$$

where O_Z is the orthographic projection matrix,

$$O_Z = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}. \tag{32.7}$$

If we do this for all n features we measure the following features in frame t,

$$W_t = \begin{bmatrix} X_{t1} & X_{t2} & \cdots & X_{tn} \\ Y_{t1} & Y_{t2} & \cdots & Y_{tn} \end{bmatrix} = O_Z R_t S, \tag{32.8}$$

with S given by (32.2). If we write

$$M_t = O_z R_t = \begin{bmatrix} i_{xt} & i_{yt} & i_{zt} \\ j_{xt} & j_{yt} & j_{zt}, \end{bmatrix} \tag{32.9}$$

(32.8) becomes

$$W_t = M_t S. \tag{32.10}$$

All that remains to be done is to collect all the observations over f frames into a single matrix W,

$$W = \begin{bmatrix} W_1 \\ W_2 \\ \vdots \\ W_f \end{bmatrix} = \begin{bmatrix} X_{11} & X_{12} & \cdots & X_{1n} \\ Y_{11} & Y_{12} & \cdots & Y_{1n} \\ \vdots & & & \vdots \\ X_{f1} & X_{f2} & \cdots & X_{fn} \\ Y_{f1} & Y_{f2} & \cdots & Y_{fn} \end{bmatrix}. \tag{32.11}$$

From the expression for a single frame (32.10) we obtain

$$W = MS, \tag{32.12}$$

where M is the *motion* matrix assembled from (32.10),

$$M = \begin{bmatrix} M_1 \\ \vdots \\ M_f \end{bmatrix} = \begin{bmatrix} i_{x1} & i_{y1} & i_{z1} \\ j_{x1} & j_{y1} & j_{z1} \\ i_{x2} & i_{y2} & i_{z2} \\ j_{x2} & j_{y2} & j_{z2} \\ & \vdots & \\ i_{xf} & i_{yf} & i_{zf} \\ j_{xf} & j_{yf} & j_{zf}. \end{bmatrix} \tag{32.13}$$

Thus, given the observation matrix W, the mathematical problem is to find M and S, i.e. the problem is reduced to factorizing the observation matrix into motion and shape matrices.

Let us take another look at the structures of the motion and shape matrices, M and S. Note that their maximum rank is 3. This means that the maximum rank of the observation matrix W should also be 3. In practice however, W is obtained by tracking features over a number of frames, introducing all kinds of tracking and measurement errors with the result that the actual rank of the $2f \times n$ matrix W will be higher than 3. We should therefore try to extract the best possible rank 3 approximation of W. We have seen that this is where the SVD is particularly effective.

It is possible for S to have rank 1, 2 or 3, where the latter indicates a full 3D object. Rank 2 corresponds to a planar object, rank 1 to a line. For instance, for a

planar object we can always choose the object coordinate systems in the plane $z = 0$. The motion matrix M can have rank 2 or 3. Rank 3 indicates a full 3D rotation and rank 2 describes a rotation around the Z-axis. A rotation around the Z-axis presents only one 'face' of the object to the film, much like the moon showing only one face to the earth. In case the observation matrix has rank 2, a full 3D object is perceived as a planar object. So, all this information is encoded in the rank of the observation matrix W. To reiterate, due primarily to measurement errors, that rank may not be exact and one needs a strategy to determine the effective rank of W. The key is SVD.

32.3. Reconstructing 3D Images.

In the previous section we showed that we need to factorize the observation matrix W into a $2f \times 3$ motion matrix M and a $3 \times n$ shape matrix S. For this purpose we calculate the full SVD of the observation matrix,

$$W = U\Sigma V^T. \tag{32.1}$$

As pointed out at the end of the previous section, the rank of W is in general higher than 3. Thus we choose the 3 largest singular values and form the rank 3 approximation of W,

$$\widetilde{W} = U_+\Sigma_+V_+^T, \tag{32.2}$$

where

$$\Sigma_+ = \mathrm{diag}\,(\sigma_1, \sigma_2, \sigma_3) \tag{32.3}$$

and U_+ and V_+ consist of the first 3 columns of U and V respectively. The expression (32.2) for $\widetilde{W}$ is therefore the best rank 3 approximation of W in the sense of (11.9).

Thus, we can write

$$\widetilde{W} = \widetilde{M}\widetilde{S}, \tag{32.4}$$

where

$$\widetilde{M} = U_+ \Sigma_+^{\frac{1}{2}} \text{ and } \widetilde{S} = \Sigma_+^{\frac{1}{2}} V_+^T. \tag{32.5}$$

Although $\widetilde{W}$ is written in the correct form, $\widetilde{M}$ and $\widetilde{S}$ are not yet the desired motion and shape matrices. In fact, the factorization is not unique. Indeed, let A be any invertible 3×3 matrix, then

$$\widetilde{W} = \left(\widetilde{M}A\right)\left(A^{-1}\widetilde{S}\right) \tag{32.6}$$

is another factorization. We now exploit the freedom provided by A to get the motion matrix M in the correct form, i.e. we choose A in such a way that

$$M = \widetilde{M}A \tag{32.7}$$

has all the properties of the motion matrix. Now it is time to recall that the motion matrix M consists of orthogonal rows in the sense of (32.6). Setting $Q = AA^T$, the orthonormality relations (32.6) are written with the help of (32.7), (32.10) and (32.7) as,

$$\widetilde{\mathbf{m}}_{2t-1}^T Q \widetilde{\mathbf{m}}_{2t-1} = 1 = \widetilde{\mathbf{m}}_{2t}^T Q \widetilde{\mathbf{m}}_{2t}, \tag{32.8}$$

$$\widetilde{\mathbf{m}}_{2t-1}^T Q \widetilde{\mathbf{m}}_{2t} = 0. \tag{32.9}$$

Here, $\widetilde{\mathbf{m}}_k^T$ is the kth row of $\widetilde{M}$ and $t = 1, \ldots, f$. Since Q is a symmetric 3×3 matrix it has 6 unknown entries that can be determined in a least squares sense from the $3f$ equations (32.8).

Once Q is calculated, another factorization is required to obtain A. This problem is also undetermined. Indeed, A has 9 unknown entries and we only know the 6 elements of the symmetric matrix Q. So, 3 elements of A remain undetermined. The reason for this is that we are free to choose the object's coordinate system—since we do not yet allow translation, we are free to choose its rotational orientation. Thus, through factorization of Q we find A only up to an arbitrary rotation. More precisely, if $Q = \widetilde{A}\widetilde{A}^T$ then $Q = (\widetilde{A}R)(\widetilde{A}R)^T$ is another factorization for any rotation matrix R. Any 3×3 rotation matrix has precisely 3 arbitrary elements, these are the 3

undetermined elements of A. We therefore have a choice and one possibility is to require the object to be oriented as in the first frame.

Since Q is symmetric it can be diagonalized with an orthonormal matrix U_Q (see e.g. [**?**]),

$$Q = U_Q \Lambda_Q U_Q^T,$$

where one again notes that the eigenvalues of $Q = AA^T$ are all non-negative. If $\widetilde{A} = U_Q \Lambda_Q^{1/2}$ then $A = \widetilde{A} R^T$ for any 3×3 rotation matrix R. Choosing A such that the object is oriented as in the first frame, R follows from

$$\mathbf{w}_1 = O_Z R \widetilde{A}^{-1} \widetilde{S}, \tag{32.10}$$

where $\mathbf{w}_1$ denotes the first two rows of the observation matrix (32.8). Looking more carefully at (32.10) we note that it consists of six linear equations (the left hand side is a 2×3 matrix) in six unknowns, $O_Z R$ selects the first two rows of the 3×3 matrix R. Since $\widetilde{A}$ and $\widetilde{S}$ are known, we can solve the resulting 6×6 linear system. For a rotation matrix this then determines the last row uniquely.

The final step is to calculate the motion and shape matrices from,

$$M = \widetilde{M} \widetilde{A} R T \text{ and } S = R \widetilde{A}^{-1} \Sigma + \frac{1}{2} V + T. \tag{32.11}$$

We now summarize the algorithm, assuming that the rigid object only rotates (translation is described in the next section):

(1) Select n features on the object and track their images on the film over f frames.
(2) Assemble the x and y coordinates of the features on the f frames into a single $2f \times n$ observation matrix W.
(3) Compute the SVD of $W = U \Sigma V^T$.
(4) Determine the effective rank of W by inspecting the singular values. For non-degenerate motion rank$(W) = 3$, in which case form the best rank 3 approximation of W namely $\widetilde{W} = U_+ \Sigma_+ V_+^T$.
(5) Construct $\widetilde{M} = U_+ \Sigma_+^{\frac{1}{2}}$.

(6) Calculate the symmetric matrix Q as the least-squares solution of (32.8).
(7) Diagonalize Q, i.e. $Q = U_Q \Lambda_Q U_Q^T$.
(8) Set $\widetilde{A} = U_Q \Lambda_Q^{\frac{1}{2}}$.
(9) Calculate R by solving the linear system of equations implied by (32.10).
(10) Compute $M = \widetilde{M}\widetilde{A}R^T$ and $S = R\widetilde{A}^{-1}\Sigma_+^{\frac{1}{2}}V_+^T$.

32.4. Rotation and Translation.

For simplicity we have assumed up to now that we are only dealing with pure rotations. Now we incorporate translations into the model. The ideas as well as much of the formalism are the same as for pure rotations, if we use the so-called homogeneous coordinates. If during the t-th frame a translation $\mathbf{c}_t$ occurs, the observation is

$$\mathbf{w}_t = O_Z \left(R_t S + \mathbf{c}_t \right). \tag{32.1}$$

In homogeneous coordinates (see [?] for an excellent account of the applications of projective geometry to computer vision) each feature point is written as

$$\mathbf{p}_s = \begin{bmatrix} x_s \\ y_s \\ z_s \\ 1 \end{bmatrix}, \tag{32.2}$$

resulting in a homogeneous shape matrix of the form,

$$S = \begin{bmatrix} x_1 & x_2 & & x_n \\ y_1 & y_2 & \cdots & y_n \\ z_1 & z_2 & & z_n \\ 1 & 1 & & 1 \end{bmatrix}. \tag{32.3}$$

The advantage of the homogeneous coordinates is that the translation can now be incorporated into the motion matrix as,

$$M = \left[\begin{array}{ccc|c} i_{x1} & i_{y1} & i_{z1} & c_{x1} \\ j_{x1} & j_{y1} & j_{z1} & c_{y1} \\ & \vdots & \vdots & \\ i_{xf} & i_{yf} & i_{zf} & c_{xf} \\ j_{xf} & j_{yf} & j_{zf} & c_{yf} \end{array}\right] =: [M_r \;\; \mathbf{c}], \tag{32.4}$$

with the result that the observation matrix is again written as a product $W = MS$. We now proceed along similar lines as with pure rotation, again the SVD of W is calculated. In this case the maximum rank of W in the non-degenerate case is 4. Accordingly we extract the best rank 4 approximation of W and factorize into $\widetilde{M}$ and $\widetilde{S}$. The calculation of A requires a bit more care. For rank$(W) = 4$, A is 4×4 and can be partitioned as

$$A = \left[\begin{array}{cc} A_r & \mathbf{a}_c \end{array}\right], \tag{32.5}$$

where A_r is a 4×3 matrix related to rotation and $\mathbf{a}_c$ is a vector related to translation. More specifically, $\widetilde{M} A_r = M_r$, where M_r is the rotational part of the motion matrix in (32.4). Thus A_r is again obtained from the constraints (32.8), but now with $Q = A_r A_r^T$.

In order to determine $\mathbf{a}_c$ we note that we are free to choose the translational orientation of the object's coordinate system. Since the centroid of the object maps to the centroid of the observations under an orthographic projection, the centroid of the observations is

$$\overline{\mathbf{w}} = \left[\begin{array}{c} \frac{1}{n}\sum_{s=1}^{n} X_{s1} \\ \frac{1}{n}\sum_{s=1}^{n} Y_{s1} \\ \vdots \\ \frac{1}{n}\sum_{s=1}^{n} X_{sf} \\ \frac{1}{n}\sum_{s=1}^{n} Y_{sf} \end{array}\right]. \tag{32.6}$$

If

$$\overline{\mathbf{p}} = \frac{1}{n}\sum_{s=1}^{n} \mathbf{p}_s \tag{32.7}$$

is the centroid of the object, then

$$\overline{\mathbf{w}} = \widetilde{M} \begin{bmatrix} A_r & \mathbf{a}_c \end{bmatrix} \begin{bmatrix} \overline{\mathbf{p}} \\ 1 \end{bmatrix}. \tag{32.8}$$

The simples choice for the origin of the object coordinate system is $\overline{\mathbf{p}} = 0$. Consequently,

$$\overline{\mathbf{w}} = \widetilde{M}\mathbf{a}_c. \tag{32.9}$$

This over-constrained problem for $\mathbf{a}_c$ is again solved with the least squares method.

The algorithm can be extended to handle several objects moving simultaneously, but independently. Tracking the features is considerably harder and requires one to determine to what object each feature belongs. That is a non-trivial task for which more details are given in [**?**].

32.5. Example.

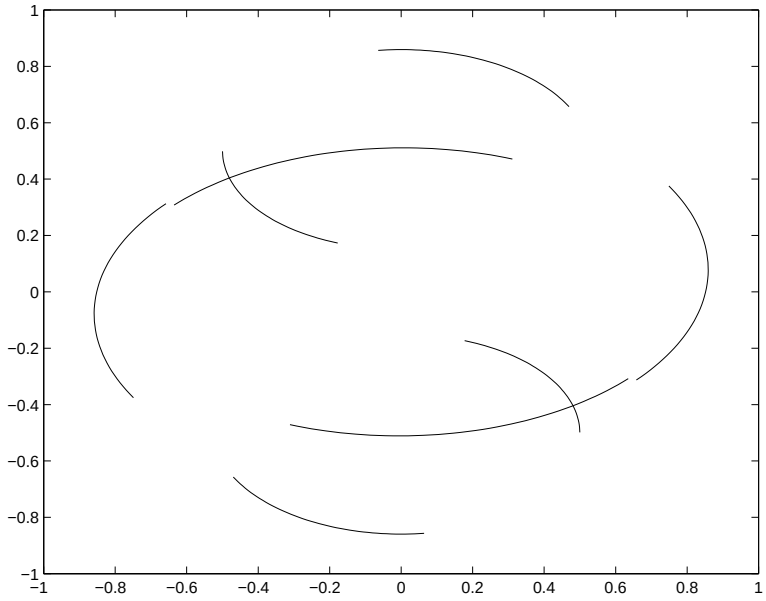


FIGURE 32.5.1. Tracking the corners (features) of a rotating cube.

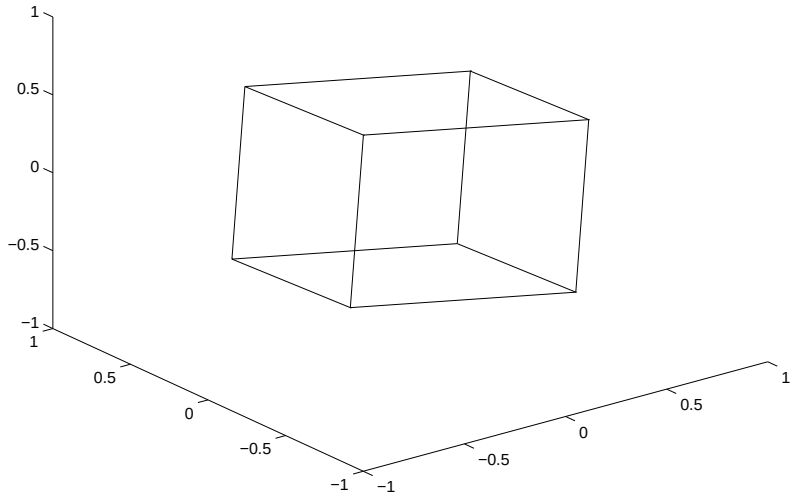


FIGURE 32.5.2. Reconstruction of the cube.

For our first example, we generate a cube and then rotate it mathematically. The eight vertexes of the cube form natural features and during the rotation their images on an imaginary film under an orthographic projection are recorded. Now imagine that the shutter of the camera is kept open while recording the projections of the feature points so that each one of the 8 vertexes traces a specific trajectory on the film plane, as shown in Figure 32.5.1. This is the only information available about the object, and the movement of the vertexes illustrated in Figure 32.5.1 are then assembled in the observation matrix W. For this example W turned out to be 100×8, implying that the 8 vertexes are tracked through 50 frames. It is from this measurement matrix that the motion and shape is reconstructed. It is not possible to illustrate the construction of the motion matrix but the shape matrix, hence the object, is shown in Figure 32.5.2, see also [**?**]. It may be worth pointing out that it is only the features that are reconstructed and that the lines connecting the vertexes in Figure 32.5.2 can only be drawn because we have advances knowledge of the object—it is of course possible to connect the vertexes in a different manner where it might not be so obvious that we are dealing with a cube.

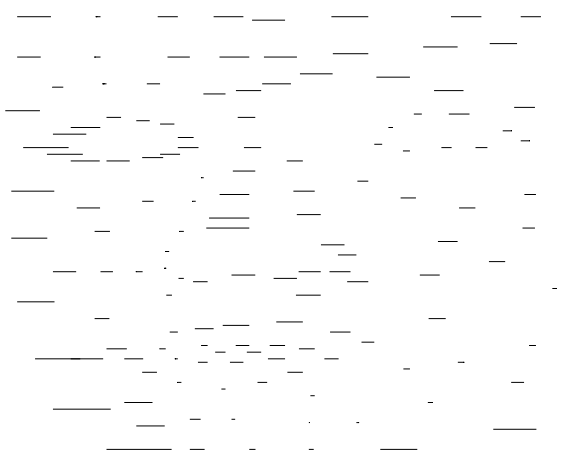
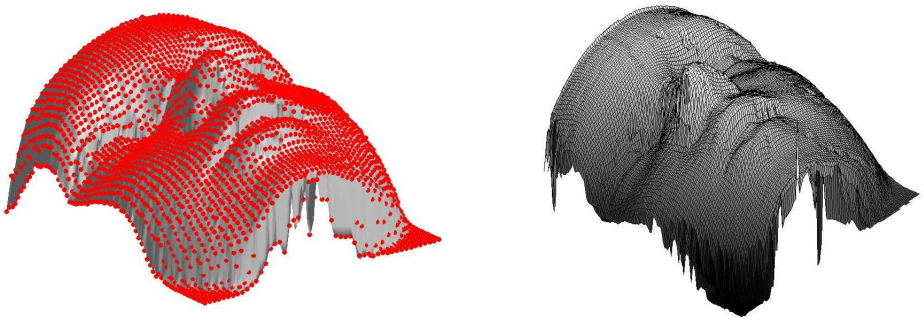


FIGURE 32.5.3. Tracking the facial features.

For our second example we use 3D data obtained from scanning the face of a mannequin. The facial data was then rotated mathematically and the feature points orthographically projected onto the film. In our first experiment the feature points were chosen by hand, shown as red dots in Figure 32.5.4(a). The motion of the feature points on the film is shown in Figure 32.5.3. Note that the 3D information is encoded in the different path lengths traced by the different feature points on the film. The fact that the features follow straight lines is the result of a pure rotation around the vertical axis. The reconstruction is shown in Figure 32.5.4(b). Since a small number of feature points was used, the reconstruction is not particularly good.

In order to convert these ideas to a practical facial identification system, a sufficient number of features should be identified on the face for a complete reconstruction of the face. Tracking these features in a video sequence containing different views of the face would then allow one to do a 3D reconstruction as described above. The 3D image can then be displayed as a gray-scale image at which stage the eigenface procedure can be applied. As pointed out before, the main practical difficulties lie with the image processing—identifying and tracking a sufficient number of features to allow a facial reconstruction.

It is also of interest to note that the features appear as scattered data. We do not in general have control over the position of the features—one has to use whatever



(a) (b)

FIGURE 32.5.4. The original and reconstructed faces.

is available. Therefore, in order to interpolate in between the features—fill in the face—the Radial Basis Functions of Chapter 14 come to our rescue.

We have an ex
ample where w
interpolated th
facial data wit
RBFs. Possibly
good example fo
inclusion.

Bibliography

[1] Etham Alpaydin. *Introduction to Machine Learning (Adaptive Computation and Machine Learning)*. MIT Press, 2010.

[2] George B. Arfken and Hans J. Weber. *Mathematical Methods for Physicists, Sixth Edition: A Comprehensive Guide*. Academic Press, 2005.

[3] David Barber. *Bayesian Reasoning and Machine Learning*. Cambridge University Press, 2012.

[4] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer, 2007.

[5] R. Fletcher. *Practical Methods of Optimization*. Wiley, 2nd edition, 2000.

[6] B. Fornberg, N. Flyer, S. Hovde, and C. Piret. Locality properties of radial basis function expansion coefficients for equispaced interpolation. *IMA Journal of Num. Anal.*, 28:121–142, 2008.

[7] Carl E. Froberg. *Introduction to Numerical Analysis*. Addison Wesley Publishing Company, second edition, 1969.

[8] David E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Professional, 1989.

[9] E. J. Hinch. *Perturbation Methods (Cambridge Texts in Applied Mathematics)*. Cambridge University Press, 1991.

[10] David J. C. MacKay. *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, 2002.

[11] Stephen Marsland. *Machine Learning: An Algorithmic Perspective (Chapman & Hall/CRC Machine Learning & Pattern Recognition)*. Chapman and Hall/CRC, 2009.

[12] Marc Mézard and Andrea Montanari. *Information, Physics, and Computation (Oxford Graduate Texts)*. Oxford University Press, USA, 2009.

[13] Melanie Mitchell. *An Introduction to Genetic Algorithms (Complex Adaptive Systems)*. The MIT Press, 1998.

[14] Jorge Nocedal and Stephen Wright. *Numerical Optimization* . Springer, 2006.

[15] Frank W. J. Olver, Daniel W. Lozier, Ronald F. Boisvert, and Charles W. Clark. *NIST Handbook of Mathematical Functions*. Cambridge University Press, 2010.

[16] Judea Pearl. *Causality: Models, Reasoning and Inference*. Cambridge University Press, 2009.

[17] Lloyd N. Trefethen. *Spectral Methods in MATLAB* . SIAM: Society for Industrial and Applied Mathematics, 2001.

[18] P. J. M. van Laarhoven. *Simulated Annealing: Theory and Applications*. Springer Netherlands, 1987.

Index

V

W