Chapter 1

Brief Summary of Finite Difference Methods

This chapter provides a brief summary of FD methods, with a special emphasis on the aspects that will become important in the subsequent chapters.

1.1 Finite Difference formulas

Finite differences (FD) approximate derivatives by combining nearby function values using a set of *weights*. Several different algorithms for determining such weights are mentioned in Sections 1.1.1 - 1.1.5. In the very simplest case, illustrated in Figure 1.1, we use the mathematical definition of a derivative

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \tag{1.1}$$

to arrive at a two-node FD formula.

Taylor expansion of (1.1) shows that

$$\frac{f(x+h) - f(x)}{h} = f'(x) + \frac{h}{2!} f''(x) + \frac{h^2}{3!} f'''(x) + \ldots = f'(x) + O(h^1),$$

i.e. the approximation $f'(x) \approx \frac{f\,(x+h) - f\,(x)}{h}$ is accurate to *first order*. The FD *weights* at the *nodes* x and $x+h$ are in this case $[-1 \;\; 1] \,/\, h$. The FD *stencil* can graphically be illustrated as

$$\begin{array}{cc} \bigcirc & \leftarrow \text{ entry for } f', \text{ value } \{1\} \\ \blacksquare \quad \blacksquare & \leftarrow \text{ entries for } f, \text{ values } \{-\frac{1}{h}, \; \frac{1}{h}\} \\ \uparrow \quad \uparrow & \\ x \quad x+h & \leftarrow \text{ spatial locations} \end{array} \quad . \tag{1.2}$$

The open circle indicates a (typically) unknown derivative value, and the filled squares (typically) known function values. While the *compactness* of this approximation is convenient (it uses only two adjacent function values), its low order of accuracy (first order; exact only for polynomials up to degree one) makes it almost entirely useless for practical computing. Before considering the application of FD formulas to tasks such as approximating ODEs and PDEs (ordinary and partial differential equations), we consider next some different procedures for creating higher order FD approximations.

1.1.1 Some direct approaches for generating FD stencils

The three approaches described next are flexible and conceptually quite straightforward, but also rather inefficient in terms of their operation count. Even so, it should be noted that the linear systems approach (Section 1.1.1.3) will become prominent later in the RBF and RBF-FD contexts. For simplicity of notation,

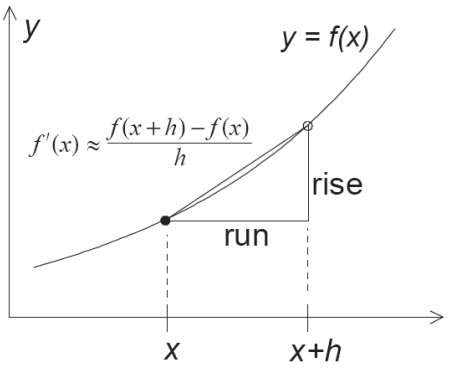


Figure 1.1: Illustration of the approximation $f'(x) \approx \dfrac{\text{rise}}{\text{run}} = \frac{f\,(x+h)-f\,(x)}{h}$, increasingly accurate as $h \to 0$.

we do not describe the approaches in their most general form, but choose the specific example of finding the weight vector $[-\frac{1}{2}\ \ 0\ \ \frac{1}{2}]/h$ in the second order approximation to the first derivative

$$f'(x) \approx \frac{-\frac{1}{2}f\,(x-h) + \frac{1}{2}f\,(x+h)}{h}\,. \tag{1.3}$$

1.1.1.1 Derivative of Lagrange's interpolation polynomial

The value for x does not influence the weights in a formula such as (1.3), so we can assume that the stencil is centered at $x = 0$. The *Lagrange interpolation polynomial* $p(x)$, taking the desired values at the nodes $x = -h, 0, h$, becomes

$$p(x) = \tfrac{(x-0)(x-h)}{(-h-0)(-h-h)}f(-h) + \tfrac{(x+h)(x-h)}{(0+h)(0-h)}f(0) + \tfrac{(x+h)(x-0)}{(h+h)(h-0)}f(+h).$$

Differentiating this polynomial with respect to x and then setting $x = 0$ gives $p'(0) = (-\frac{1}{2}f(-h) + 0\ f(0) + \frac{1}{2}f(+h))/h$, in agreement with (1.3).

1.1.1.2 Taylor expansions

Expressing $f(-h),\ f(0),\ f(h)$ by Taylor expansion around $x = 0$ gives

$$\left\{\begin{array}{ll} f(-h) & = f(0) - \frac{h}{1!}f'(0) + \frac{h^2}{2!}f''(0) - + \ldots \\ f(0) & = f(0) \\ f(h) & = f(0) + \frac{h}{1!}f'(0) + \frac{h^2}{2!}f''(0) \quad + \ldots \end{array}\right. . \tag{1.4}$$

We want to find weights $w_{-1},\ w_0,\ w_1$ such that

$$w_{-1}f(-h) + w_0 f(0) + w_1 f(h) = 0\ f(0) + 1\ f'(0) + 0\ f''(0) + \ldots$$

Using the expansions from (1.4) and equating coefficients for $f(0),\ f'(0),\ f''(0)$ gives rise to a linear system to solve for the unknown coefficients

$$\begin{bmatrix} 1 & 1 & 1 \\ -\frac{h}{1!} & 0 & \frac{h}{1!} \\ \frac{h^2}{2!} & 0 & \frac{h^2}{2!} \end{bmatrix} \begin{bmatrix} w_{-1} \\ w_0 \\ w_1 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \tag{1.5}$$

with the solution $w_{-1} = -\frac{1}{2h},\ w_0 = 0,\ w_1 = \frac{1}{2h}$.

1.1.1.3 Use of monomial test functions

Continuing with the same example, we want the formula $f'(0) \approx w_{-1}f(-h) + w_0 f(0) + w_1 f(h)$ to be exact for as high degree polynomials as possible. Enforcing it in turn for the monomials $f = 1,\ f = x$ and $f = x^2$ gives

$$\begin{array}{lcllll} f=1 & \Rightarrow & w_{-1} & +w_0 & +w_1 & =0 \\ f=x & \Rightarrow & w_{-1}(-h) & & +w_1(h) & =1 \\ f=x^2 & \Rightarrow & w_{-1}(-h)^2 & & +w_1(h)^2 & =0 \end{array} ,$$

equivalent to (1.5).

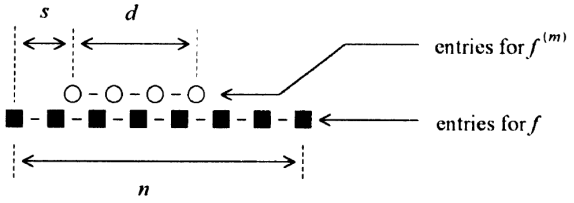
In the more general case of finding the weights $w_1,\ w_2, \ldots,\ w_n$ to use at locations $x_1,\ x_2, \ldots,\ x_n$ for approximating a linear operator L at some location $x = x_c$, we similarly solve the system

$$\begin{bmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_n \\ \vdots & \vdots & \ddots & \vdots \\ x_1^{n-1} & x_2^{n-1} & \cdots & x_n^{n-1} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} = \begin{bmatrix} L\ 1|_{x=x_c} \\ L\ x|_{x=x_c} \\ \vdots \\ L\ x^{n-1}|_{x=x_c} \end{bmatrix} \tag{1.6}$$

The successive lines of this system enforce that the set of weights lead to the correct result for the functions $1,\ x,\ x^2, \ldots,\ x^{n-1}$ and thus, by linearity, for all polynomials up through degree $n-1$. This direct linear systems approach is very flexible and easy to implement. However, it is not computationally fast ($O(n^3)$ operations), and the coefficient matrix can become ill-conditioned. It will however become the primary approach in the context of RBF methods.

1.1.2 Padé-based algorithm for equispaced grids

When the nodes have a uniform spacing h (as has been the case in the examples above), a particularly short symbolic algebra algorithm was presented in 1998 [9]. We generalize the stencil (1.2) to



Here, the numbers s, d, and n describe the stencil shape. In the illustration above, these take the values 3/2, 3, and 7, respectively. The weights, one at each node point, relate nodal values of the m^{th} derivative of f with the nodal values of the function f. In Mathematica (version 7 and higher) the complete code is

```
t = PadeApproximant[x^s(Log[x]/h)^m,{x,1,{n,d}}];
CoefficientList[{Denominator[t],Numerator[t]},x]
```

with similar codes in other symbolic languages. The following are three typical applications of this algorithm:

Example 1. The choices $s = 1$, $d = 0$, $n = 2$, $m = 2$ describe a stencil of the shape ■ ○ ■ ■ for approximating the second derivative (since $m = 2$). The algorithm produces the output

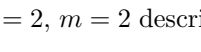
$$\left\{\left\{h^2\right\}, \{1, -2, 1\}\right\} ,$$

corresponding to the explicit 2^{nd} order accurate formula for the second derivative

$$f''(x) \approx \{f(x-h) - 2f(x) + f(x+h)\}\,\frac{1}{h^2} \ . \tag{1.7}$$

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	$\frac{4}{5}$	$-\frac{1}{5}$	$\frac{4}{105}$	$-\frac{1}{280}$	
$\vdots$		$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\vdots$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	
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 1.1: Weights for centered FD approximations of the first derivative on an equispaced grid (omitting the factor $1/h$).

Example 2. The choices $s = 0$, $d = 2$, $n = 2$, $m = 2$ describe a stencil of the shape  , again for approximating the second derivative (since $m = 2$). The algorithm produces the output

$$\left\{\left\{\frac{h^2}{12}, \frac{5h^2}{6}, \frac{h^2}{12}\right\}, \{1, -2, 1\}\right\}$$

corresponding to the compact (implicit) 4^{th} order accurate formula for the second derivative

$$\frac{1}{12}f''(x-h) + \frac{5}{6}f''(x) + \frac{1}{12}f''(x+h) \approx \{f(x-h) - 2f(x) + f(x+h)\}\frac{1}{h^2}\ . \tag{1.8}$$

Example 3. The choices $s = -2$, $d = 2$, $n = 1$, $m = 1$ describe a stencil of the shape  for approximating the first derivative. The output

$$\left\{\left\{\frac{5h}{12}, -\frac{4h}{3}, \frac{23h}{12}\right\}, \{-1, 1\}\right\}$$

is readily rearranged into

$$f(x+h) = f(x) + \frac{h}{12}(23f'(x) - 16f'(x-h) + 5f'(x-2h))\ , \tag{1.9}$$

which we later (in Section 1.2.1.2) will encounter as the third order Adams-Bashforth method for solving ODEs.

In every case, the weights will be the optimal ones with regard to formal order of accuracy. The algorithm is particularly convenient to use when only a small number of stencils are considered, and when one wants to obtain the weights in exact rational form rather than as floating point numbers.

Table 1.1 shows the lowest order centered FD formulas for the first derivative, and Table 1.2 for the second derivative. The existence of infinite order limits (indicated by the bottom line in each of the two tables) will play a key role in Section **??** when we introduce PS methods.

The special case illustrated in Example 3 can be generalized to include all the main classes of linear multistep methods. With $m = 1$ and accuracy order $p \geq 1$, the appropriate settings for s, d, and n become

Adams-Bashforth (AB)	$s = 1 - p$,	$d = p - 1$,	$n = 1$.
Adams-Moulton (AM)	$s = 2 - p$,	$d = p - 1$,	$n = 1$.
Backward Differentiation (BD)	$s = p$,	$d = 0$,	$n = p$.

order						weights						
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		$-\frac{1}{560}$	$\frac{8}{315}$	$-\frac{1}{5}$	$\frac{8}{5}$	$-\frac{205}{72}$	$\frac{8}{5}$	$-\frac{1}{5}$	$\frac{8}{315}$	$-\frac{1}{560}$		
$\vdots$		$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$		
limit	$\cdots$	$-\frac{2}{4^2}$	$\frac{2}{3^2}$	$-\frac{2}{2^2}$	$\frac{2}{1^2}$	$-\frac{\pi^2}{3}$	$\frac{2}{1^2}$	$-\frac{2}{2^2}$	$\frac{2}{3^2}$	$-\frac{2}{4^2}$		$\cdots$

Table 1.2: Weights for centered FD approximations of the second derivative on an equispaced grid (omitting the factor $1/h^2$).

1.1.3 Algorithms for arbitrarily spaced grids

FD approximations based on equispaced grids are very accurate when they are centered (extending equally far to both sides), but tend to lose accuracy when boundaries are approached and they have to become increasingly one-sided. The common remedy is to gradually cluster nodes more densely as the boundary is approached, as will be discussed further in Section **??**. A number of effective algorithms for calculating FD weights are available for such (non-equispaced) cases.

1.1.3.1 FD approximations at select points

We consider here first the case when one merely wants a few stencils. In the case of nodes located at x_i, $i = 1, 2, \ldots, n$, one can obtain the weights for $d^p/dx^p|_{x=z}$, $p = 0, 1, \ldots, m$ (where the location z may or may not coincide with any of the node locations) by means of

```
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,lentgh(x)), containing (as output) in
%     successive rows the weights for derivatives 0,1,...,m.

n = length(x); c = zeros(m+2,n); c(2,1) = 1; x1 = x(ones(1,n),:); A = x1'-x1;
b = cumprod([ones(n,1),A],2); rm = cumsum(ones(m+2,n-1))-1; d = diag(b); d(1:n-1) = d(1:n-1)./d(2:n);
for i = 2:n
   mn = min(i,m+1);
   c(2:mn+1,i) = d(i-1)*(rm(1:mn,1).*c(1:mn,i-1)-(x(i-1)-z)*c(2:mn+1,i-1));
   c(2:mn+1,1:i-1) = ((x(i)-z)*c(2:mn+1,1:i-1)-rm(1:mn,1:i-1).*c(1:mn,1:i-1))./(x(i)-x1(1:mn,1:i-1));
end
c(1,:)  = [];
```

For example, the statement **weights**$(\mathbf{0}, -\mathbf{2} : \mathbf{2}, \mathbf{6})$ returns the output

```
 0         0         1.0000    0         0
 0.0833   -0.6667    0         0.6667   -0.0833
-0.0833    1.3333   -2.5000    1.3333   -0.0833
-0.5000    1.0000    0        -1.0000    0.5000
 1.0000   -4.0000    6.0000   -4.0000    1.0000
 0         0         0         0         0
 0         0    0    0         0         0
```

This output shows the optimal weights to be applied to function values at $x = -2,\ -1,\ 0,\ 1,\ 2$, for approximating the 0^{th} up through the 6^{th} derivative at $x = 0$. Since the approximation point z coincides with one of the data points, the top line tells the obvious fact that the most accurate 'interpolation' is to just use that data value. We recognize lines 2 and 3 from the 4^{th} order approximations in Tables 1.1 and 1.2, respectively. The last two lines are all zero, reflecting the fact that there exist no formulas for the 5^{th} and 6^{th} derivative that extend over only five node points.

The derivation of the algorithm, given in [7, 9], is based on recursions that follow from Lagrange's interpolation formula. If one also wants weights for shorter stencil widths (based on $x_i,\ i = 1, 2, \ldots, \nu;\ \nu = 1, 2, \ldots, n$), these can be picked up 'for free', as otherwise discarded intermediate results after each step in the loop `for i = 2:n`. In that case, the algorithm costs only 4 arithmetic operation per calculated weight. If these shorter stencils are not wanted, the weights algorithm presented in [27] has a somewhat lower operation count, and can be coded particularly efficiently in C++. However, that code is much longer and it runs in Matlab several times slower than the present algorithm.

1.1.3.2 Algorithms for differentiation matrices (DMs)

In case one wants to employ global FD stencils (extending over all the nodes; the case with non-periodic PS methods), one typically needs a sequence of weight sets, providing approximations that are accurate at each of the nodes x_i in turn. Repeated use of the algorithm in Section 1.1.3 would be comparatively slow for producing such *differentiation matrices* (DMs), as it would not utilize the fact that the many separate cases are all based on the same node set. Several specialized algorithms for calculating such DMs have been presented [18, 32]. The algorithm and MATLAB code by Weideman and Reddy is often preferred, and is downloadable from the web [31]. Once a DM has been calculated, the derivative approximations at all the nodes are obtained by a single matrix×vector multiplication

$$\begin{bmatrix} u^{(m)}(x_1) \\ \vdots \\ u^{(m)}(x_n) \end{bmatrix} \approx \begin{bmatrix} \quad \\ \quad DM \quad \\ \quad \end{bmatrix} \begin{bmatrix} u(x_1) \\ \vdots \\ u(x_n) \end{bmatrix}.$$

1.1.4 Errors when applying FD formulas to given functions

The application of FD formulas gives rise to different types of errors:

Truncation errors: These are expressed by the leading error terms that we have quoted above, such as $O(h^2)$, $O(h^4)$, etc. In most applications, higher orders (especially above second order) are preferable. We will introduce PS methods as the limit of increasing order FD methods.

Rounding errors: The numerator in an approximation such as (1.7) can in standard arithmetic be evaluated to an accuracy of 10^{-16}, i.e. if the stencil involves a division by h^2, the rounding error becomes $O(10^{-16}/h^2)$.

Total error: The best accuracy is usually obtained when h is chosen so that the two error types above match. In the present example, $O(h^2)$ matches $O(10^{-16}/h^2)$ when $h \approx 10^{-4}$, producing a total error around 10^{-8}. Higher order FD methods fare better in this type of analysis, but higher derivatives make the situation much worse since, for the p^{th} derivative, the rounding error becomes of the form $O(10^{-16}/h^p)$. FD formulas are for this reason only rarely used for derivatives beyond the third or fourth.

In the case of FD formulas for analytic functions, there are some options available for greatly reducing the rounding errors: If a function is known to be analytic, and it can be computed also for complex arguments, Cauchy's integral formula leads to FD approximations that do not use points along the real axis but instead (for example) around a circle in the complex plane, centered at the location at which we want the derivative approximation. In this case, good accuracy does not require the circle radius to be small, and it turns out that also high order derivatives (say, the 50^{th} or the 100^{th}) become numerically available to high precision [2, 6, 22]. In a different approach, applicable only when a function $f(x)$ is analytic and real-valued for x real, and one only wants the first derivative $f'(x)$ for such an x-value, Cauchy-Riemann's equations tell us that one equivalently can evaluate the derivative in the imaginary direction. Regular FD approximations will then not suffer the usual floating point cancellations as $h \to 0$, and machine precision is readily achieved [28]. This approach has later become known as the 'Complex Step' method. It can also be noted that, for analytic functions that satisfy simple ODEs (thus excluding important classes of functions such as those involving $\Gamma(x)$, $\zeta(x)$, etc.), derivatives of any order can be calculated recursively [1].

1.1.5 Generalizations to more than 1-D

1.1.5.1 Cartesian lattices

On an (x, y)-grid, any mixed derivative, such as $\frac{\partial^3}{\partial x \partial y^2}$, is most easily approximated by a stencil that amounts to approximating in the two directions in sequence. Just as in the case of analytic differentiation, the result will not depend on in which order the partial derivations were carried out. The combined procedure can directly be formulated in terms of a 2-D stencil.

Example 1. Create the second order centered approximation for $\frac{\partial^3}{\partial x \partial y^2}$. Multiplication of the stencils for $\frac{\partial^2}{\partial y^2}$ and $\frac{\partial}{\partial x}$ (from Tables 1.2 and 1.1, resp.) gives

$$\begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix} /h^2 \times \begin{bmatrix} -\frac{1}{2} & 0 & \frac{1}{2} \end{bmatrix} /h = \begin{bmatrix} -\frac{1}{2} & 0 & \frac{1}{2} \\ 1 & 0 & -1 \\ -\frac{1}{2} & 0 & \frac{1}{2} \end{bmatrix} /h^3.$$

Example 2. Create second and fourth order approximation of the 2-D Laplace operator

$$L = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \ .$$

Based on the top two lines in Table 1.2, the 1-D formulas (1.7) for $\frac{\partial^2}{\partial x^2}$ of accuracies $O(h^2)$ and $O(h^4)$ can be written

$$\begin{aligned} &[1 \ -2 \ 1] /h^2 \ , \\ &[-\frac{1}{12} \ \frac{4}{3} \ -\frac{5}{2} \ \frac{4}{3} \ -\frac{1}{12}] /h^2. \end{aligned}$$

Applying these operators in the x- and y-directions and adding the results leads directly to the 2-D counterparts (accurate to orders $O(h^2)$ and $O(h^4)$, respectively):

$$\begin{bmatrix} & 1 & \\ 1 & -4 & 1 \\ & 1 & \end{bmatrix} /h^2 \ \text{ and } \ \begin{bmatrix} & & -\frac{1}{12} & & \\ & & \frac{4}{3} & & \\ -\frac{1}{12} & \frac{4}{3} & -5 & \frac{4}{3} & -\frac{1}{12} \\ & & \frac{4}{3} & & \\ & & -\frac{1}{12} & & \end{bmatrix} /h^2. \tag{1.10}$$

PS methods pursue this stencil construction concept in Example 2 to still higher orders. However, in the context of solving PDEs, this approach might not be optimal, as seen in the examples of Section 1.2.2, and again (from a different perspective) in Section **??**.

1.1.5.2 Hexagonal grids

With a specified distance h between adjacent nodes in 2-D, this grid type provides the densest possible node arrangement. However, boundary conditions may become more difficult to implement, and it also becomes less clear how to best approximate simple derivatives, such as $\frac{\partial}{\partial x}$ and $\frac{\partial}{\partial y}$ (since both of these cannot be aligned with the primary grid directions). Nevertheless, FD approximations can become pleasingly simple. For example, the Laplacian operator $\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$ on a hexagonal node set with the six nodes adjacent to $(0, 0)$ located at $(\pm h, 0)$, $(\pm\frac{1}{2}h, \pm\frac{\sqrt{3}}{2}h)$ can be approximated by

$$\begin{bmatrix} & 1 & 1 & \\ 1 & & -6 & & 1 \\ & 1 & 1 & \end{bmatrix} / (\tfrac{3}{2}h^2), \tag{1.11}$$

accurate to $O(h^2)$. We will study the accuracy of this approximation further in Example 2, Section 1.2.2.

Our main interest in hexagonal grids will arise later in the context of RBF and RBF-FD methods. Both hexagonal and quasi-random (e.g. 'Minimal Energy' distributed) node layouts then become very easy to work with, and it will transpire that they can offer advantages over Cartesian-based ones.

1.2 Application of FD formulas to PDEs

The idea of using FD approximations for numerical solutions of PDEs was first proposed by L.F. Richardson in 1911 [24]. In a revolutionary paper, he notes that "*Step-by-step arithmetical methods of solving ordinary differential equations have long been employed...*" and he then proceeds with generalizing this concept to 2-D. After a flawless equilibrium calculation of the stresses in a cross-section of the first Aswan Dam over the Nile, he looks to the future: "*The extension to three variables is, however, perfectly obvious. One has only to let the third variable be represented by the number of the page of a book of tracing paper.*" Although our computational hardware is now far more powerful than pencil and paper, the basic concept of FD approximations for PDEs remains the same.

1.2.1 Time dependent PDEs

If a PDE is time dependent,

$$\frac{\partial u}{\partial t} = F(u, u_x, u_y, \ldots, \{\textit{maybe higher spatial derivatives}\})$$

the most straightforward FD approach is to place a Cartesian grid over the spatial domain. Given an approximation for u at time t, F is approximated at all interior nodes (with boundary information incorporated as needed), and an ODE solver is invoked to advance in time the resulting coupled ODE system (featuring a separate ODE for each node point). This general approach is known as the Method of Lines (MOL). Its main advantages include:

- Easy to reach high accuracy (and thereby high computational efficiency) in both space and time,
- Easy to interchange ODE solvers (such as explicit, implicit, different orders, etc.)
- Reduced need for user input with regard to time step selection, error control, etc.

Example 1. Create an FD scheme for the 1-D heat equation $\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2}$ with the smallest possible stencil size. This FD approximation would become

$$\frac{u(x,t+k) - u(x,t)}{k} = \frac{u(x-h,t) - 2u(x,t) + u(x+h,t)}{h^2}. \tag{1.12}$$

This stencil can in the (x,t)-plane be illustrated as  . This amounts to a MOL approximation, using the top line in Table 1.2 for the spatial approximation, and then time stepping with the Forward Euler method (also known as AB1, or Adams-Bashforth of first order). The accuracy becomes $O(k) + O(h^2)$, i.e. the scheme is first order accurate in time and second order accurate in space.

The four key concepts with regard to FD approximations to time dependent PDEs are *Convergence, Accuracy*, *Consistency*, and *Stability*. These are connected by the *Lax Equivalence Theorem*: For a well posed linear problem a consistent approximation is convergent if and only if it is stable. Consistency only requires that the FD formula, when k and $h \to 0$, approximates the PDE; first order accuracy in both directions suffice. Hence, stability (ensuring that the numerical approximation does not diverge to infinity within finite time) becomes the key property to test for.

For an explicit time stepping scheme, such as in Example 1 above, the numerical approximation at any fixed time $t > 0$ will diverge to infinity when the step sized k and h are decreased, unless a certain relation between k and h is satisfied, namely $k/h^2 \leq \frac{1}{2}$. There are three main methods to determine such stability conditions. Two of these will be described below. In cases of nonlinearities, variable coefficients, and non-trivial boundary conditions, a third approach - energy methods - is often the only available option. However, its generality usually comes at the price of significant algebraic complexity. A brief summary (focusing on the same case in Example 1) is given in [8], Appendix H. More extensive treatments can be found in [15, 25]. Even when the assumptions of the two approaches below are not fully satisfied, they still often give good indications when applied to suitably simplified equations (e.g. after linearization of nonlinear terms, assumption of spatial periodicity, etc.) The influence of boundary conditions on stability is discussed in [29].

1.2.1.1 von Neumann stability analysis

This approach is strictly applicable only on periodic or infinite domains, and with constant coefficients in the PDE. At time level t, let the numerical solution be a combination of modes $u(x,t) = \sigma^{t/k} e^{i\omega x}$, where σ is the factor by which this mode grows in amplitude for each time step. Substitution into (1.12) gives, after some simplifications, $\sigma = 1 - 4\frac{k}{h^2}\sin^2\left(\frac{\omega h}{2}\right)$. For the solution to stay bounded, we need $|\sigma| \leq 1$ to hold for all ω in the range $|\omega| \leq \pi/h$ (all modes that can be represented on a grid with spacing h). The stability condition thus becomes $k/h^2 \leq \frac{1}{2}$ (as σ then, for all the ω-values, will become confined to the interval $[-1, 1]$).

1.2.1.2 ODE stability domains

Each numerical ODE integration technique has an associated *stability domain* - the region in a complex ξ-plane, with $\xi = \lambda k$, for which the ODE method does not have any growing solutions when it is applied to the constant coefficient ODE

$$y' = \lambda y \,. \tag{1.13}$$

Example 2. Find the stability domain for Forward Euler. This scheme $y(t+k) = y(t) + ky'(t)$ becomes in the case of (1.13) $y(t+k) = (1+\lambda k)y(t)$. With $\xi = \lambda k$, the stability domain thus becomes $|1+\xi| \leq 1$, a circle of radius 1 centered at $\xi = -1$.

Example 3. Find the stability domain for the third order Adams-Bashforth method, as given by (1.9). As an approximation to (1.13), the scheme becomes $y(t+k) = y(t) + \frac{\lambda k}{12}(23y(t) - 16y(t-k) + 5y(t-2k))$. With $\xi = \lambda k$, this linear recursion relation has as its characteristic equation $r^3 = (1 + \frac{23\xi}{12})r^2 - \frac{4\xi}{3}r + \frac{5\xi}{12}$. Next, we solve for ξ to obtain $\xi = \frac{12r^2(r-1)}{23r^2-16r+5}$. The edge of the stability domain is traced out in the complex ξ-plane if we let r move around the edge of the unit circle. The result is seen as the $p = 3$ curve in Figure 1.2 a.

For convenient reference, Figures 1.2-1.5 display the stability domains for a number of well-known ODE solvers. The two schemes in Figure 1.5 are both enhancements to the second order accurate leapfrog (LF) scheme, which we in slightly abbreviated notation write as

$$y_{n+1} = y_{n-1} + 2k\, f(t_n, y_n) \,.$$

In both subplots, the LF stability domain is shown as the line segment joining $+i$ and $-i$. The Hyman method [19] includes a corrector step

$$\begin{cases} \hat{y}_{n+1} &= y_{n-1} + 2k\, f(t_n, y_n) \\ y_{n+1} &= \frac{4}{5}y_n + \frac{1}{5}y_{n-1} + \frac{2}{5}k\left(f(t_{n+1}, \hat{y}_{n+1}) + 2f(t_n, y_n)\right) \end{cases} ,$$

resulting in third order of accuracy and a greatly enlarged stability domain (extending to $-\frac{3}{2}$ and $\pm\frac{3}{2}i$ along respective axes). LF with the Robert filter [26] proceeds by repeating the steps

$$\begin{cases} \hat{y}_{n+1} &= y_{n-1} + 2k\, f(t_n, \hat{y}_n) \\ y_n &= \hat{y}_n + \gamma\left(\hat{y}_{n+1} - 2\hat{y}_n + y_{n-1}\right) \end{cases} ,$$

where γ is a positive parameter. This procedure requires only one function evaluation per time step. However, the accuracy drops to first order (although the actual degradation is small if γ is small). The approach is best suited for cases when eigenvalues stay close to the imaginary axis, rather than extending far into the left halfplane.

Time staggered ODE solvers are discussed in [11, 12, 23, 30]; see also Section **??**. For further discussions of the major classes of ODE solvers, a book specializing on the topic should be consulted, e.g. [16, 17, 20].

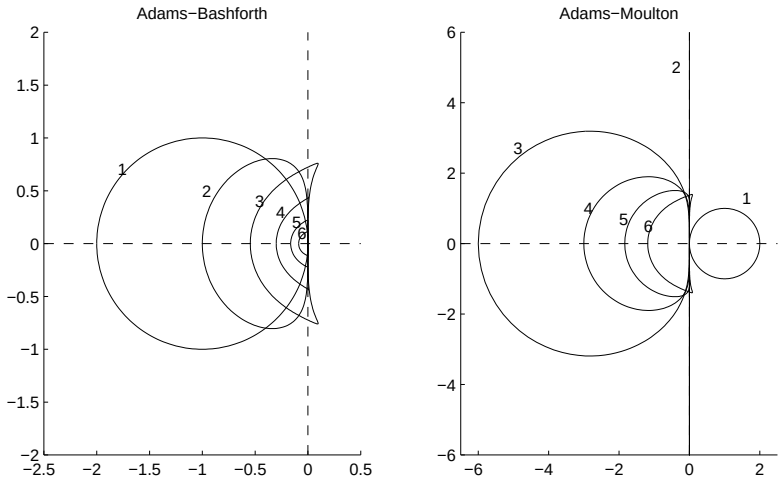


Figure 1.2: Stability domains for (a) Adams-Bashforth (AB) and (b) Adams-Moulton (AM) methods of orders $p = 1, 2, \ldots, 6$. The stability domains in all cases include the regions immediately to the left of the origin, i.e. for AM1, it is the domain $|1 - \xi| \geq 1$, and for AM2 the left halfplane. In all other cases, the regions are bounded. A section along the imaginary axis near the origin is included for AB methods of orders $3, 4, 7, 8, 11, 12, \ldots$ and for AM of orders $1, 2, 5, 6, 9, 10, \ldots$. Note that the scale differs by a factor of three between the two figures.

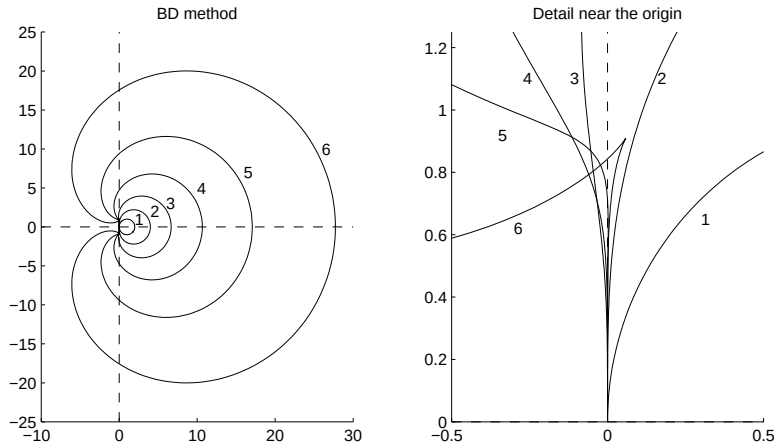


Figure 1.3: Stability domains of backward differentiation (BD) methods of orders 1-6. The domains are in all cases outside of the shown boundary curves. (a) The complete boundary curves, (b) Detailed boundary structure near the origin.

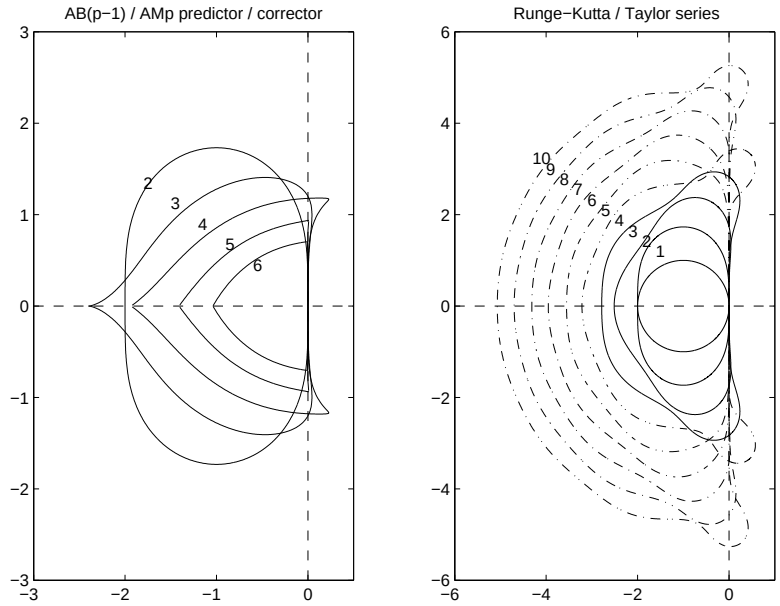


Figure 1.4: Stability domains for (a) AB(p-1)/AMp predictor/corrector methods for $p = 2, 3, \ldots, 6$. A section along the imaginary axis near the origin is included for AB(p-1)/AMp methods of orders $3, 4, 7, 8, 11, 12, \ldots$ and for ABp/AMp methods of orders $1, 2, 5, 6, 9, 10, \ldots$ [13], (b) Solid curves: Runge Kutta (RK) methods or orders (= number of stages) $1, 2, 3, 4$. Solid and dash-dot curves: Taylor series methods of orders $p = 1, 2, \ldots, 10$. Note that the scales differ by a factor of two between the plots.

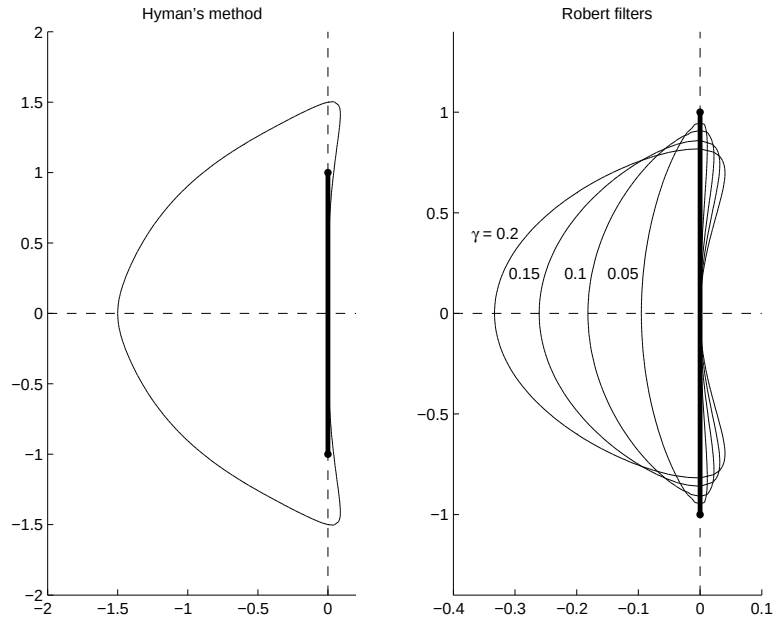


Figure 1.5: Stability domains for (a) Hyman's method, and (b) Leapfrog with Robert filter, for four values of the parameter γ. Note the different scales along the axes in the two subplots.

1.2.1.3 Stability analysis via ODE stability domains

We give here only one illustrative example.

Example 4. Determine the stability condition for (1.12) - same test case as with von Neumann stability above - but using the ODE stability domain approach. The scheme (1.12) amounts to using Forward Euler (also described as AB1) in time together with the spatial approximation, i.e. to solve the ODE system

$$\frac{d}{dt}\begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ \vdots \\ u_n \end{bmatrix} = \begin{bmatrix} -2 & 1 & & & 1 \\ 1 & -2 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \ddots \\ 1 & & & 1 & -2 \end{bmatrix} 1/h^2 \begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ \vdots \\ u_n \end{bmatrix}.$$

The matrix is symmetric, i.e. all its eigenvalues are real. By *Gershgorin's theorem*, they are located on the interval $[-4, 0]$. If the number of nodes n is even, inspection shows that $[1, 1, \ldots, 1]^T$ is an eigenvector with eigenvalue 0 and $[1, -1, 1, -1, \ldots, -1]^T$ with eigenvalue -4, so both interval bounds are sharp (n odd makes an insignificant difference). When including the $1/h^2$ factor, the eigenvalues satisfy $\lambda \in [-4, 0]/h^2$. This range has to fall within the ODE solver's stability domain, which in this case is $\xi = \lambda k \in [-2, 0]$ (according to Example 2 above, or by inspecting the $p = 1$ curve in Figure 1.2 a). We thus need $[-4, 0]/h^2 \in [-2, 0]/k$, which simplifies to $k/h^2 \leq \frac{1}{2}$.

In the later contexts of RBF and RBF-FD methods, the generally irregular node layouts will make von Neumann analysis impossible. In contrast, the ODE stability domain approach, together with a numerical calculation of the differentiation matrix' eigenvalues, will become the outstanding time stepping analysis tool. Comparison of the eigenvalue distributions with the different ODE methods' stability domains will immediately give an excellent guide to both what time stepping method to choose, and then to the step sizes that can be used.

1.2.1.4 The Courant-Friedrichs-Lewy (CFL) condition

The two previous stability tests (von Neumann analysis and matrix eigenvalues together with ODE stability domains) have given necessary and (almost) sufficient stability conditions (some subtleties can arise if the the DMs are non-normal, i.e. can't be diagonalized by a unitary matrix). The CFL condition [4] is an extraordinarily quick test that can show that some schemes are unconditionally unstable and others are unstable for certain k-values, but it can *never* show a scheme to be stable. In heuristic terms, it states that a FD scheme (discrete in both space and time) must be unstable if the stencil shape does not allow information to flow with the speed required by the PDE. We apply next this test to two different schemes for the 1-D one-way wave equation

$$\frac{\partial u}{\partial t} + \frac{\partial u}{\partial x} = 0. \tag{1.14}$$

Example 5. Use the CFL test for $\frac{u(x,t+k)-u(x,t)}{k} + \frac{u(x,t)-u(x-h,t)}{h} = 0$;  . Figure 1.6 illustrates the characteristic speed of the PDE ($v_{PDE} = 1$; the velocity by which the solution travels in the (x, t)-plane), and the maximal speed by which information can travel sideways in the same direction in the numerical solution ($v_{NUM} = h/k$). The condition $v_{NUM} \geq v_{PDE}$ tells us that, for $k/h \leq 1$ (the stencil shape in the illustration), the scheme *might* be stable. For $k/h \geq 1$, v_{NUM} is not fast enough, and the scheme *must* be unstable.

Example 6. Use the CFL test for $\frac{u(x,t+k)-u(x,t)}{k} + \frac{u(x+h,t)-u(x,t)}{h} = 0$;  . With this stencil shape, $v_{NUM} = 0$ for all combinations of k and h, and the scheme must therefore be unconditionally unstable.

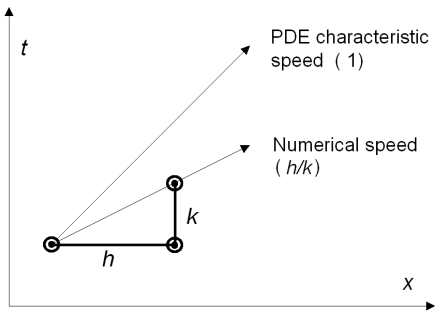


Figure 1.6: Schematic illustration of the CFL analysis for the stencil $\frac{u(x,t+k)-u(x,t)}{k} + \frac{u(x+h,t)-u(x,t)}{h} = 0$.

1.2.2 Elliptic type PDEs

We restrict our discussion here to the case of Poisson's equation in 2-D

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f \ . \tag{1.15}$$

Equations in this 'elliptic' category arise in numerous situations, such as for a *streamfunction* in fluid mechanics or from *field equations* (describing gravitational and electrical fields, featuring *potentials* which satisfy *Laplace's equation*; equation (1.15) with RHS zero). Another source of equations of this type is equilibrium processes. For example, (1.15) arises in the $t \to \infty$ limit of the heat equation $\frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} - f$.

Example 1. Create the following *compact* fourth order accurate approximation for the 2-D Poisson's equation (a 2-D counterpart to (1.8)):

$$\begin{bmatrix} 1 & 4 & 1 \\ 4 & -20 & 4 \\ 1 & 4 & 1 \end{bmatrix} u/(6h^2) = \begin{bmatrix} & 1 & \\ 1 & 8 & 1 \\ & 1 & \end{bmatrix} f \ /12 \ + O(h^4). \tag{1.16}$$

To derive this, we follow Collatz' *Mehrstellenverfahren* [3, 10]. Because of $\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) u = f$, it also holds that

$$\underbrace{\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)^2}_{\frac{\partial^4}{\partial x^4} + 2\frac{\partial^4}{\partial x^2 \partial y^2} + \frac{\partial^4}{\partial y^4}} u = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) \ f$$

Approximation of these two relations to 4^{th} and to 2^{nd} order, respectively, give

$$\begin{bmatrix} & & -\frac{1}{12} & & \\ & & \frac{4}{3} & & \\ -\frac{1}{12} & \frac{4}{3} & -5 & \frac{4}{3} & -\frac{1}{12} \\ & & \frac{4}{3} & & \\ & & -\frac{1}{12} & & \end{bmatrix} u/h^2 = [f] + O(h^4) \ , \tag{1.17}$$

and

$$\begin{bmatrix} & & 1 & & \\ & 2 & -8 & 2 & \\ 1 & -8 & 20 & -8 & 1 \\ & 2 & -8 & 2 & \\ & & 1 & & \end{bmatrix} u/h^4 = \begin{bmatrix} & 1 & \\ 1 & -4 & 1 \\ & 1 & \end{bmatrix} f \ /h^2 + O(h^2) \ , \tag{1.18}$$

respectively. Adding $\frac{1}{12}h^2$ times (1.18) to (1.17) eliminates the 'outliers', and produces (1.16).

The formula (1.16) achieves its fourth order only thanks to the stencil for f in the right hand side (RHS). As an approximation to the Laplace operator, the LHS of (1.16) is only accurate to second order, as seen by Taylor expanding it around the center point:

$$\begin{aligned}\begin{bmatrix} 1 & 4 & 1 \\ 4 & -20 & 4 \\ 1 & 4 & 1 \end{bmatrix} u/(6h^2) = &\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) u \\ &+ \frac{1}{12}h^2 \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)^2 u \\ &+ \frac{1}{360}h^4 \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)\left(\frac{\partial^4}{\partial x^4} + 4\frac{\partial^4}{\partial x^2\partial y^2} + \frac{\partial^4}{\partial y^4}\right) u \\ &+ \frac{1}{60480}h^6 \left(\frac{\partial^4}{\partial x^4} + 4\frac{\partial^4}{\partial x^2\partial y^2} + \frac{\partial^4}{\partial y^4}\right)\left(3\frac{\partial^4}{\partial x^4} + 16\frac{\partial^4}{\partial x^2\partial y^2} + 3\frac{\partial^4}{\partial y^4}\right) u \\ &+ O(h^8)\end{aligned}$$

For solutions to Laplace's equation, the first three RHS terms vanish, and the approximation becomes sixth order accurate. The two key advantages of (1.16) over (1.17) are

- The compact stencil is easier to use near boundaries, and
- The diagonal dominance of coefficient matrix improves numerical stability and speeds up iterative solution methods.

Example 2. Analyze the accuracy of the hexagonal grid Laplace operator approximation (1.11). Series expansion in the same style as in the previous example gives

$$\begin{aligned}\begin{bmatrix} & 1 & 1 & \\ 1 & & -6 & & 1 \\ & 1 & 1 & \end{bmatrix} / \left(\tfrac{3}{2}h^2\right) = &\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right) u \\ &+ \frac{1}{16}h^2 \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)^2 u \\ &+ \frac{1}{5760}h^4 \left(11\frac{\partial^6}{\partial x^6} + 15\frac{\partial^6}{\partial x^4\partial y^2} + 45\frac{\partial^6}{\partial x^2\partial y^4} + 9\frac{\partial^6}{\partial y^6}\right) u \\ &+ O(h^6)\end{aligned}$$

This expansion confirms that the approximation is only second order accurate for the Laplace operator, but shows that it supports a compact fourth order approximation for (1.15). However, the accuracy improves no further for solutions to Laplace's equation, since the operator in the h^4-term does not factorize. It thus falls short in this regard of the Cartesian grid compact 9-point operator analyzed in Example 1.

Extending from 2-D to 3-D does not introduce any significant differences. For example, the 3-D counterpart to (1.16) becomes

$$\begin{bmatrix} & & [0 & 1 & 0] \\ & [1 & 2 & 1] & \\ [0 & 1 & 0] & & \\ - & - & - & - & - \\ & & [1 & 2 & 1] \\ & [2 & -24 & 2] & \\ [1 & 2 & 1] & & \\ - & - & - & - & - \\ & & [0 & 1 & 0] \\ & [1 & 2 & 1] & \\ [0 & 1 & 0] & & \end{bmatrix} u/(6h^2) = \begin{bmatrix} & & [0 & 0 & 0] \\ & [0 & 1 & 0] & \\ [0 & 0 & 0] & & \\ - & - & - & - & - \\ & & [0 & 1 & 0] \\ & [1 & 6 & 1] & \\ [0 & 1 & 0] & & \\ - & - & - & - & - \\ & & [0 & 0 & 0] \\ & [0 & 1 & 0] & \\ [0 & 0 & 0] & & \end{bmatrix} f\ /12\ +\ O(h^4), \tag{1.19}$$

again combining fourth order accuracy with diagonal dominance [33]. In this case, the 19 point stencil in the LHS is an $O(h^2)$ accurate approximation to the Laplacian operator, which reaches $O(h^4)$ for solutions to Laplace's equation - as it does for the Poisson's equation when used with the shown RHS stencil.

The last several examples have shown that FD approximations can provide higher orders of accuracy for PDEs than the orders by which they approximate individual derivative operators. This issue will come up again in the context of RBF-FD methods. Numerous generalizations of the compact formulas mentioned above have been described in the literature, including to variable coefficients, inclusion of lower order terms, extensions to the coupled streamfunction-vorticity system for the steady 2-D Navier-Stokes equations, etc. [5, 14, 21].

Bibliography

[1] D. Barton, I. M. Willers, and R. V. M. Zahar, *An implementation of the Taylor series method for ordinary differential equations*, Comput. J. **14** (1971), 243–248.

[2] F. Bornemann, *Accuracy and stability of computing high-order derivatives of analytic functions by Cauchy integrals*, Found. Comput. Math. **11** (2011), 1–63.

[3] L. Collatz, *The Numerical Treatment of Differential Equations*, Springer Verlag, Berlin, 1960.

[4] R. Courant, K. Friedrichs, and H. Lewy, *On the partial differential equations of mathematical physics, (in German)*, Math. Ann. **100** (1928), 32–74.

[5] E. Erturk and C. Gökçöl, *Fourth order compact formulation of Navier-Stokes equations and driven cavity flow at high Reynolds numbers*, Int. J. Num. Meth. Fluids **50** (2006), 412–436.

[6] B. Fornberg, *Numerical differentiation of analytic functions*, ACM Tans. Math. Softw. **7** (1981), 512–526.

[7] ———, *Generation of finite difference formulas on argitrarily spaced grids*, Math. Comput. **184** (1988), 699–706.

[8] ———, *A Practical Guide to Pseudospectral Methods*, Cambridge University Press, Cambridge, 1996.

[9] ———, *Calculations of weights in finite difference formulas*, SIAM Rev. **40** (1998), 685–691.

[10] L. Fox, *Some improvements in the use of relaxation methods for the solution of ordinary and partial differential equations*, Proc. Roy. Soc. A **190** (1947), 31–59.

[11] M. Ghrist and B. Fornberg, *Two results concerning the stability of staggered multistep methods*, SIAM J. Numer. Anal. **50** (2012), 1849–1860.

[12] M. Ghrist, B. Fornberg, and T. A. Driscoll, *Staggered time integrators for wave equations*, SIAM J. Numer. Anal. **38** (2000), 718–741.

[13] M. L. Ghrist, J. A. Reeger, and B. Fornberg, *Stability ordinates of Adams predictor-corrector methods*, (2013).

[14] M. M. Gupta, *High accuracy solutions of incompressible Navier-Stokes equations*, J. Comput. Phys. **93** (1991), 343–359.

[15] B. Gustafsson, H. O. Kreiss, and J. Oliger, *Time Dependent Problems and Difference Methods*, Wiley, New York, 1995.

[16] E. Hairer, S. P. Nørsett, and G. Wanner, *Solving Ordinary Differential Equations i - Non-stiff Problems*, Springer Verlag, Berlin, 1987.

[17] E. Hairer and G. Wanner, *Solving ordinary differential equations ii - Stiff and differential-algebraic problems*, Springer Verlag, Berlin, 1991.

[18] W. Huang and D. M. Sloan, *The pseudospectral method for solving differential eigenvalue problems*, J. Comput. Phys. **111** (1994), 399–409.

[19] J. M. Hyman, *Numerical methods for nonlinear differential equations*, Nonlinear Problems: Present and Future (A. R. Bishop, D. K. Campbell, and B. Nicolaenko, eds.), North-Holland Publishing Company, 1982, pp. 91–107.

[20] J. D. Lambert, *Numerical Methods for Ordinary Differential Systems: The Initial Value Problem*, Wiley, New York, 1991.

[21] M. Li, T. Tang, and B. Fornberg, *A compact fourth-order finite difference scheme for the steady incompressible Navier-Stokes equations*, Int. J. for Num. Meth. in Fluids **20** (1995), 1137–1151.

[22] J. N. Lyness and C. B. Moler, *Numerical differentiation of analytic functions*, SIAM J. Numer. Anal. **4** (1967), 202–210.

[23] D. Murai and T. Koto, *Stability and convergence of staggered Runge-Kutta schemes for semilinear wave equations*, J. Comput. Appl. Math. **235** (2011), 4251–4264.

[24] L. F. Richardson, *The approximate arithmetical solution by finite differences of physical problems involving differential equations, with an application to the stresses in a masonry dam*, Phil. Trans. Royal Soc. London **210** (1911), 307–357.

[25] R. D. Richtmyer and K. W. Morton, *Difference Methods for Initial-Value Problems, 2nd ed.*, Wiley, New York, 1967.

[26] A. J. Robert, *The integration of a low order spectral form of the primitive meteorological equations*, J. Meteor. Soc. of Japan **44** (1966), 237–245.

[27] B. Sadiq and D. Viswanath, *Finite difference weights, spectral differentiation, and superconvergence*, Math. Comput. (2013), In Press.

[28] W. Squire and G. Trapp, *Using complex variables to estimate derivatives of real functions*, SIAM Rev. **40** (1998), 110–112.

[29] L. N. Trefethen, *Group velocity interpretation of the stability theory of Gustafsson, Kreiss and Sundström*, J. Comput. Phys. **49** (1983), 199–217.

[30] J. G. Verwer, *On time staggering for wave equations*, J. Sci. Comput. **33** (2007), 139–154.

[31] J. A. C. Weideman and S. C. Reddy, *A MATLAB differentiation matrix suite*, ACM Trans. Math. Software **26** (2000), 465–519.

[32] B. D. Welfert, *Generation of pseudospectral differentiation matrices I*, SIAM J. Numer. Anal. **34** (1997), 1640–1657.

[33] S. Zhai, X. Feng, and Y. He, *A family of fourth-order and sixth-order compact difference schemes for the three-dimensional Poisson equation*, J. Sci. Comput. **54** (2013), 97–120.