

APPM 2360 Project 2

Eigenvalues in Plankton Blooms

Due: Thursday, April 4th, 2019 by 11:59 p.m.
Submit as a PDF to “Assignments” on Canvas

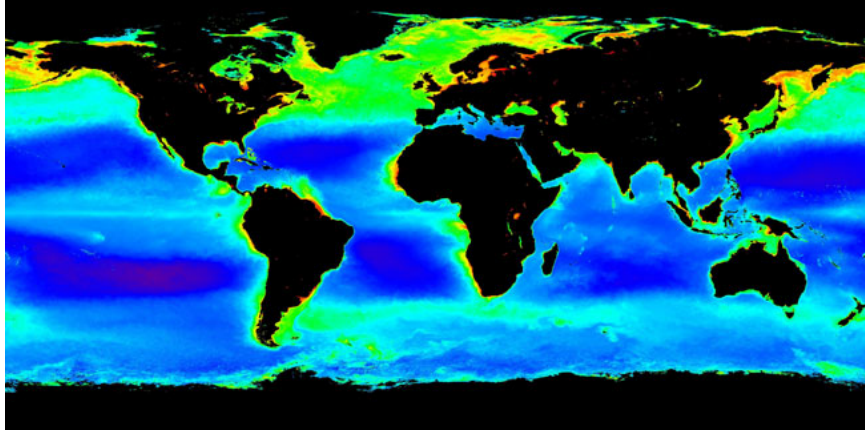


Figure 1: Plankton Density Across the world [3].

Abstract

Finding eigenvalues of large matrices is a problem that comes up frequently in applications. The strategy of finding λ such that the equation $(\mathbf{A} - \lambda\mathbf{I})\mathbf{v} = 0$ is nonsingular becomes too costly as the size of n grows. We must resort to other methods for computation of eigenvalues in large systems. In this project we analyze two methods: *QR Algorithm* and the *Power Method*. QR iterations takes the original matrix $\mathbf{A}$ and finds a sequence of matrices similar to $\mathbf{A}$ that converges to an upper triangular matrix. Once we have a the upper triangular matrix $\mathbf{T}$ that is similar (similar means the two matrices have the same eigenvalues) to $\mathbf{A}$, then we are able to simply read off the eigenvalues from the diagonal of $\mathbf{T}$. Our power method is different in the sense that it is used to compute the extreme eigenvalues i.e., the largest and smallest eigenvalues of a matrix. We use a model describing the size of plankton population as a case study of how the eigenvalue problem can arise in applications. In this project we derive a condition based on the smallest eigenvalue of the system for the population to grow.

1 Introduction

The motivation for this project comes from the growth-diffusion equation of plankton populations. Phytoplankton provide 50% of the world's oxygen and allow some of the atmosphere's carbon dioxide to be absorbed into the ocean. It is thus important for us to know when a plankton bloom will be growing or decaying. This problem was studied in an important paper of Kierstead and Slobodkin [1]. The simplest growth model of population of plankton is the following [1]

$$\frac{dN}{dt} = (K - \lambda)N$$

where K is the growth rate, λN is the rate of population loss by diffusion, and N is the population size. For the simple growth model, growth of a plankton population will occur if $K > \lambda$. However, since the

population is modeled with diffusion, we instead use the diffusion term in the simple growth model arriving at a partial differential equation that describes growth in space and time

$$\frac{\partial c}{\partial t} = D \frac{\partial^2 c}{\partial x^2} + Kc \quad (1)$$

for the concentration $c(x, t)$ of phytoplankton. Here, we sum the diffusion term with the term Kc which represents the linear growth term of the plankton bloom. The concentration is assumed to be zero at the ends of an interval of length l , so we have the boundary conditions $u(0, t) = u(l, t) = 0$.

The diffusion equation is very well understood and easy to work with, so we make the transformation

$$c(x, t) = u(x, t) e^{Kt} \quad (2)$$

and get the diffusion equation

$$\frac{\partial u}{\partial t} = D \frac{\partial^2 u}{\partial x^2} \quad (3)$$

Introduction Questions

1. Show that the substitution of (2) into (1) reduces the model to (3).

2 The Exact Solution

To solve equation (3) when the diffusion coefficient D is constant, we will use separation of variables. We assume a solution of the form $u(x, t) = X(x)T(t)$. Substituting into equation (3) leads to the two equations

$$T' + \lambda DT = 0 \quad (4)$$

$$X'' + \lambda X = 0 \quad (5)$$

where λ are our eigenvalues and the solution to the diffusion equation. The boundary conditions are

$$X(0) = 0 = X(l)$$

This gives us the eigenvalues and eigenfunctions

$$\lambda_n, \quad X_n(x) = \sin(\sqrt{\lambda_n}x), \quad T_n(t) = e^{-D\lambda_n t}$$

Each product $X_n(x)T_n(t)$ is a solution of (3), and satisfies the boundary conditions. The general solution of the diffusion equation (3) is obtained by taking the weighted sum of all these solutions $X_n T_n$:

$$u(x, t) = \sum_{n=1}^{\infty} A_n \sin(\sqrt{\lambda_n}x) e^{-D\lambda_n t}$$

The concentration is, from (2),

$$c(x, t) = \sum_{n=1}^{\infty} A_n \sin(\sqrt{\lambda_n}x) e^{(K-D\lambda_n)t} \quad (6)$$

The constants A_n are determined by the initial conditions $c(x, 0)$. The population will grow if the growth rate $K > \lambda_1$, where λ_1 is the smallest eigenvalue known to be

$$\lambda_1 = \left(\frac{\pi}{l}\right)^2 \quad (7)$$

We will show how discretization of the diffusion equation (3) leads to this result.

Section 2: Questions

1. Solve the ODEs obtained by the separation of variables (4) and (5) subject to the boundary conditions to obtain the eigenvalues and eigenfunctions stated above.
2. Explain why taking a weighted sum of solutions is also a solution to the differential equation. (Hint: can you write (3) as a linear operator)
3. Use (2) to derive the solution for the concentration from the solution of $u(x, t)$.
4. Explain why the plankton population will grow if the growth rate $K > D\lambda_1$.

3 Discretization

In this section, we will show how discretization of the eigenvalue problem leads to our result. We can discretize the operator X'' , to obtain the matrix eigenvalue problem, which approximates the eigenvalue problem (4). Discretize the interval $(0, l)$ into $n + 1$ parts of length $h = l/(n + 1)$. Then the second centered difference approximation of the second derivative is [2]:

$$X''(x) \approx \frac{X(x - h) - 2X(x) + X(x + h)}{h^2}$$

Let $x_k = X(hk)$. Then the eigenvalue problem (4) is approximated by the matrix-vector eigenvalue problem

$$\frac{1}{h^2} \begin{bmatrix} 2 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & -1 & 2 & -1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & \ddots & \ddots \\ & & & & \ddots & 2 & -1 \\ & & & & & -1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \lambda \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} \quad (8)$$

Note that the equation is for the *interior* points, as $X(0) = 0 = X(l)$, corresponding to $x_0 = 0 = x_{n+1}$. Let B_n be the $n \times n$ “ $-1, 2, -1$ ” matrix above. Then the eigenvalue problem can be written as

$$\left(\frac{n+1}{l}\right)^2 B_n \mathbf{x} = \lambda \mathbf{x}$$

We define $A_n \equiv \left(\frac{n+1}{l}\right)^2 B_n$. For simplicity assume that $l = 1$ for the remainder of the project. The eigenvalues of this matrix approximate the eigenvalues for the plankton model. As $n \rightarrow \infty$ the approximations become increasingly accurate.

Section 3: Questions

1. The derivative of a single valued function f at a point x can be defined as either

$$f'(x) = \lim_{h \rightarrow 0} \frac{f(x+h) - f(x)}{h} \approx \frac{f(x+h) - f(x)}{h} \quad (9)$$

or as

$$f'(x) = \lim_{h \rightarrow 0} \frac{f(x) - f(x-h)}{h} \approx \frac{f(x) - f(x-h)}{h}. \quad (10)$$

Where the approximation is increasingly accurate for small h . The first approximation is known as a forward difference approximation and the second is known as a backward difference. Derive the approximation for the second derivative by applying (9) to X' and (10) to your approximation for X' .

2. Derive the system of equations that leads to the matrix equation. To show this explicitly write out the equations for

$$-X'' = \lambda X$$

at the points x_1, x_2, x_3 and recognize the pattern.

3. Write a program that creates the A_n matrix.

4 QR Algorithm

We now use the *QR Algorithm* to find the eigenvalues of several $\mathbf{A}_n$. We extract the smallest eigenvalue and compare it to the actual value $\lambda_1 = \left(\frac{\pi}{l}\right)^2$. Before applying the algorithm we study how the algorithm works. To understand this algorithm we study Gram-Schmidt algorithm and the QR factorization.

Gram-Schmidt algorithm

Given a set of linearly independent vectors $\{\mathbf{v}_i\}_{i=1}^n$ it is possible to create an orthonormal basis $\{\mathbf{u}_i\}_{i=1}^n$. To illustrate this let us analyze low-dimensional cases and build up an algorithm. If we have one vector in our set $\{u\}$, then to construct an orthonormal basis we can set

$$\mathbf{u} = \frac{\mathbf{v}}{\|\mathbf{v}\|}$$

For the case when we have two vectors in our set $\{\mathbf{v}_1, \mathbf{v}_2\}$ we start off by setting

$$\mathbf{u}_1 = \frac{\mathbf{v}_1}{\|\mathbf{v}_1\|}$$

We now have to find a vector that is orthogonal to $\mathbf{v}_1$. We can write a vector in terms of its components parallel and perpendicular to the line spanned by $\mathbf{u}_1$.

$$\mathbf{v}_2 = \mathbf{v}_2^{\parallel} + \mathbf{v}_2^{\perp}$$

Then the vector

$$\mathbf{v}_2^{\perp} = \mathbf{v}_2 - \text{proj}_{\text{span}\{\mathbf{u}_1\}}(\mathbf{v}_2) = \mathbf{v}_2 - (\mathbf{u}_1 \cdot \mathbf{v}_2)\mathbf{u}_1$$

is orthogonal to $\mathbf{u}_1$. We divide $\mathbf{v}_2^{\perp}$ by its length to get the second vector $\mathbf{u}_2$ of the orthonormal basis.

$$\mathbf{u}_2 = \frac{\mathbf{v}_2^{\perp}}{\|\mathbf{v}_2^{\perp}\|}$$

For general number of vectors we have the following algorithm

$$\mathbf{u}_1 = \mathbf{v}_1, \quad \mathbf{u}_n = \mathbf{v}_n - \sum_{k=1}^{n-1} \frac{(\mathbf{u}_k, \mathbf{v}_n)}{\|\mathbf{u}_k\|^2} \mathbf{u}_k$$

where we are subtracting off the projection of $\mathbf{v}_n$ on to the subspace spanned by $\{\mathbf{u}_i\}_{i=1}^{n-1}$ to find a vector orthogonal to the previous vectors. The final step is to divide each vector by its norm to obtain an orthonormal basis.

1. Write a program that takes in a matrix of size 3×3 and uses the Gram-Schmidt algorithm to find an orthonormal basis for the column space of the matrix. Apply this program to A_3 . Verify your algorithm works by showing the dot product $(\mathbf{u}_i, \mathbf{u}_j) = 0$ or numerical zero for $i \neq j$. Also, check that each vector has length 1.

QR Factorization

The Gram-Schmidt process takes in our “old” basis and produces a “new” orthonormal basis. This transformation can be described by a change of basis matrix $\mathbf{R}$. We can write this as

$$[\mathbf{v}_1 \ \dots \ \mathbf{v}_n] = [\mathbf{u}_1 \ \dots \ \mathbf{u}_n] \mathbf{R}$$

Here $\mathbf{Q}$ is an orthogonal matrix by construction and $\mathbf{R}$ is an upper triangular matrix.

1. Use the file `cgs.m` to obtain the QR factorization of A_n matrices for size 3×3 and 4×4 .

QR Algorithm in Practice

The following algorithm is used to compute the eigenvalues of a matrix $\mathbf{A}$.

Step 0: Set $\mathbf{A}_0 = \mathbf{A}$

Step k: Set $\mathbf{A}_k = \mathbf{Q}_k \mathbf{R}_k$ then set $\mathbf{A}_{k+1} = \mathbf{R}_k \mathbf{Q}_k$

Here the matrix $\mathbf{A}_{k+1}$ is similar to the matrix $\mathbf{A}_k$ and under certain conditions $\mathbf{A}_k$ converge to an upper triangular matrix (or diagonal if $\mathbf{A}$ is symmetric).

1. Show that the matrix $\mathbf{A}_{k+1}$ is similar to the matrix $\mathbf{A}_k$. Recall a matrix $\mathbf{A}$ is similar to a matrix $\mathbf{B}$ if they have the same eigenvalues.
2. Apply the *QR Algorithm* 1000 times to $\mathbf{A}_n$ of size 10×10 , 20×20 and 100×100 give the smallest eigenvalue for each (smallest element on the diagonal) and compute the error between this value and λ_1 . (Hint: you can extract the diagonal of a matrix $\mathbf{A}$ using the `diag` command)

The Power Method

We now analyze the extreme eigenvalues via the power method. We begin by giving a description. If you assume that there is an eigenbasis (there is for our problem). Then we can write any vector as

$$\mathbf{x} = c_1 \mathbf{v}_1 + c_2 \mathbf{v}_2 + \dots + c_n \mathbf{v}_n$$

for $c_i \in \mathbb{R}$.

If we apply $\mathbf{A}$ to this equation we obtain the equation

$$\mathbf{Ax} = c_1 \mathbf{Av}_1 + c_2 \mathbf{Av}_2 + \dots + c_n \mathbf{Av}_n$$

and applying the eigenvalue relationship ($\mathbf{Av}_i = \lambda_i \mathbf{v}_i$) we obtain the equation

$$\mathbf{Ax} = c_1 \lambda_1 \mathbf{v}_1 + c_2 \lambda_2 \mathbf{v}_2 + \dots + c_n \lambda_n \mathbf{v}_n.$$

We assume that the largest eigenvalue is distinct from the others i.e., $|\lambda_1| > |\lambda_2| \geq \dots |\lambda_n|$. Then we can factor out λ_1 .

$$\mathbf{Ax} = \lambda_1 \left(c_1 \mathbf{v}_1 + c_2 \frac{\lambda_2}{\lambda_1} \mathbf{v}_2 + \dots + c_n \frac{\lambda_n}{\lambda_1} \mathbf{v}_n \right)$$

Repeated applications of $\mathbf{A}$ give us the equation

$$\mathbf{A}^n \mathbf{x} = \lambda_1^n \left(c_1 \mathbf{v}_1 + c_2 \left(\frac{\lambda_2}{\lambda_1} \right)^n \mathbf{v}_2 + \dots + c_n \left(\frac{\lambda_n}{\lambda_1} \right)^n \mathbf{v}_n \right)$$

As $n \rightarrow \infty$ we have $\left(\frac{\lambda_j}{\lambda_1} \right)^n$ will go to zero for $j \neq 1$.

1. You are given the code that gives you the largest eigenvalue using this method `power_method.m`. Use this code to find the smallest eigenvalues of $\mathbf{A}_n$ of size 10×10 , 20×20 and 100×100 compute the error between this value and λ_1 . (Hint: what is the relationship between the eigenvalues of a matrix $\mathbf{A}$ and $\mathbf{A}^{-1}$)

Comparison

Compare the two methods. Which one did you like more?

5 References

References

- [1] H. Kierstead and L. Slobodkin. The size of water masses containing plankton blooms. *J. Marine Res.*, 12:141–147, 1953.
- [2] David Kincaid and Ward Cheney. *Numerical analysis*. American Mathematical Society, Providence, RI, third edition, 2002. Mathematics of scientific computing.
- [3] Nasa.gov. Chlorophyll a concentration, 2003. [Online; accessed April 03, 2017].

6 Project Guidelines

Your group will submit your project write-up on Canvas to the appropriate “Project Assignment” (you can find these under the “assignments” tab in Canvas). Adhere to the following guidelines:

- Do not put off finding a group (you must work in groups of 2-3). You should have a group set-up within one week of the project assignment due date.
- Submit your project in a **pdf** format and submit ALL code used for your project (.nb files for Mathematica, .m files for Matlab, .py or ipynb for python). Code in **Matlab, Mathematica, Maple, R, C, C++, Python or Julia** is acceptable (Matlab is recommended). Code in *Microsoft Word or Excel* (or any other spreadsheet program) is *not acceptable*. All other languages need instructor permission (please ask as soon as possible).

Code may be included in the appendix if you wish. DO NOT submit anything on Canvas as a .zip file. Contents of .zip files will not necessarily be graded.

- Have only ONE group member submit the project. Having multiple people in your group submit the project to Canvas will result in multiple grades, and we will take the LOWEST one.
- Include the names and recitation section numbers of all group members working on the project on the cover page of the report.
- When you submit the report to Canvas, please include each group member’s information (name, student number, and section number) in the comments. This allows us to quickly search for a student’s report.

Your report needs to accurately and consistently describe the steps you took to answer the posed questions. This report should have the look and feel of a technical paper. Presentation and clarity are very important. Here are some important items to remember:

- Remember: you are to submit a complete report for this project. Documents submitted with numbered responses will be severely penalized.
- Labs must be typed, including all equations (part of your learning experience is to learn how to use an equation editor). An exception can be made for lengthy calculations in the appendix, which may be hand written (as long as they are neat and clear), and minor labels on plots, arrows in the text, and a few subscripts.

- Write your report in an organized and logical fashion. Section headers such as Introduction, Background, Problem Statement, Calculations, Results, Conclusion, ... are not mandatory but are highly recommended. They not only help you write your report, but help the reader navigate your paper.
- Start with an introduction that describes what you will discuss in the body of the report. A brief summary of important concepts used in your discussion could be helpful here as well. Always introduce relevant equations that will be used or discussed in the report.
- Always include units in your answers
- DO include and label any plot that supports your conclusions or gives you insight in your investigations (these should be found in the body of your report). However, DO NOT use screen-shots of your figures.
- DO NOT include printouts of computer software screens or code snippets. You simply need to state which software you used in each step and what it did for you.
- DO include the results of any calculations in the main body of your report. DO NOT include lengthy calculations in the main body of your paper (calculations should be included in a labeled appendix and should be referred to in the main body).
- Your report does not have to be long. You need quality, not quantity, of work. Do not omit any important piece of information, but do not feel obligated to add any extras.
- Summarize what you have accomplished in the conclusion. No new information or new results should appear in your conclusion. You should only review the highlights of what you wrote about in the body of the report.