Towards the Resolution of an Eigenvalue Problem for the 2-Hessian Operator

BY

KARA NEELY
B.S., Spelman College, 2008

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master's of Science in Mathematics
in the Graduate College of the
University of Illinois at Chicago, 2013

Chicago, Illinois

ACKNOWLEDGMENTS

I would like to thank my thesis committee–Gerard Awanou, Jerry Bona and Charles Knessl–for their guidance. They provided unwavering support in every aspect to assist me in accomplishing my research goals.

I would especially like to thank my thesis advisor, Gerard Awanou, for his patience, motivation, and immense knowledge. He helped me in both the research and writing of this thesis and I could not have finished my work without him.

I would also like to thank my family and friends who always encourage me with their best wishes.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

SUMMARY

This document attempts to provide a numerical solution to the Eigenvalue Problem for the 2-Hessian operator. The topics covered are as follows:

Chapter 1 introduces the k-Hessian eigenvalue problem. It also discusses what can be be accomplished with the work done in this thesis which is to solve numerically the eigenvalue problem for the 2-Hessian operator on various convex domains in $\mathbb{R}^3$ and display their log-concavity properties.

Chapter 2 discusses the Poisson equation and gives numerical solutions in both two and three dimensions.

Chapter 3 discusses the Laplacian eigenvalue problem and gives numerical solutions in both two and three dimensions.

Chapter 4 discusses the Monge-Ampère equation and gives numerical solutions in two dimensions.

Chapter 5 discusses the multivariate Newton's method and gives a numerical solution to the Monge-Ampère eigenvalue problem using Newton's method in two dimensions.

Chapter 6 discusses the eigenvalue problem for the Monge-Ampère equation and gives numerical solutions in two dimensions using several iterative techniques.

All the Octave codes required to implement the numerical solutions in this document are provided in the appendix.

CHAPTER 1

INTRODUCTION

1.1 A Nonlinear Eigenvalue Problem

We look at a nonlinear eigenvalue problem which involves the k-Hessian operator. The Hessian matrix is a square matrix of second-order partial derivatives of a function. It describes the local curvature of a function of several variables. The k-Hessian is the k-trace, or the kth elementary symmetric polynomial of eigenvalues of the Hessian matrix. When $k = 1$ the k-Hessian reduces to the Laplace operator and when $k \geq 2$, the k-Hessian equation is a fully nonlinear partial differential equation.

1.1.1 Overview

For a C^2 function u on a bounded connected open set Ω of $\mathbb{R}^n$, let $\lambda_1, \ldots, \lambda_n$ denote the eigenvalues of the Hessian D^2u. The k-Hessian operator, $1 \leq k \leq n$ is defined as

$$S_k(D^2u) = \sum_{i_1 < i_2 \ldots < i_k} \lambda_{i_1} \cdot \lambda_{i_2} \ldots \lambda_{i_k}.$$

Notice that for $k = 1$, $S_1(D^2u)$ reduces to the Laplacian operator Δu. The Laplacian is a linear operator in the Euclidean n-space which will be further discussed in Chapter 3. On the contrary, when $k = n$ (for $n \geq 2$), $S_n(D^2u)$ is the Monge-Ampère operator $\det D^2u$. This is a special type of nonlinear second order partial differential equation which will be discussed in Chapters 4, 5 and 6.

1.1.2 Eigenvalue Problem for the 2-Hessian Equation

We want to consider the following eigenvalue problem

$$S_k(D^2u) = \lambda(-u)^k \text{ in } \Omega, \qquad u = 0 \text{ on } \partial\Omega. \tag{1.1}$$

It is known that the first eigenfunction of the Laplacian (for $k = 1$) is log-concave (Korevaar, 1983), that is its logarithm is concave for $u \geq 0$. However if u is not positive in Ω, we have to choose the values x, y in Ω and check the inequality $u(tx + (1 - t)y) \geq u(x)^t u(y)^{1-t}$ to determine whether the function u is log-concave. The result is open for other values of k on an arbitrary convex domain. There are results in (Wang, 1994; Gavitone, 2009) but for special domains not necessarily convex. One can contribute to the resolution of this problem by solving numerically the eigenvalue problem on various convex domains in $\mathbb{R}^3$ for $k = 2$ and displaying their log-concavity properties.

To provide the groundwork for this task we look at a simpler problem. We pose the question: is the finite difference solution of

$$\det(D^2u) = \lambda u \text{ in } \Omega, \qquad u = 0 \text{ on } \partial\Omega, \tag{1.2}$$

log concave?

It turns out that for the related Dirichlet problem for the Monge-Ampère equation

$$\det(D^2u) = f \text{ in } \Omega, \qquad u = 0 \text{ on } \partial\Omega, \tag{1.3}$$

where $f \geq 0$, Newton's method typically fails. It was proposed in (Awanou, 2010a) to use the sequence of Poisson problems

$$-\nu\Delta u_{p+1} = -\nu\Delta u_p + \det(D^2u_p) - f \text{ in } \Omega, \qquad u = 0 \text{ on } \partial\Omega, \tag{1.4}$$

where $\nu > 0$. It is reasonable to try to solve (Equation 1.2) by the sequence of Poisson equations

$$-\nu\Delta u_{p+1} = -\nu\Delta u_p + \det(D^2u_p) - \lambda_p u_p \text{ in } \Omega, \qquad u = 0 \text{ on } \partial\Omega. \tag{1.5}$$

However it is not clear how to update λ. For the moment a very good initial guess for λ would be necessary. Extending this approach to the three dimensional problem should not be very difficult but the problem of updating λ would remain.

1.2 Outline of the Thesis

I began by writing a finite difference method for the two dimensional Poisson equation by updating the basic code in (Sauer, 2006). Then I extended the code to solve the three dimensional Poisson equation. Another problem similar to (Equation 1.2) is the eigenvalue problem for the Poisson equation, also known as the Lapacian eigenvalue problem. This is solved in this thesis in both two and three dimensions. It can be used as an initial guess for (Equation 1.2). The iterative method for the Monge-Ampère equation is implemented in Chapter 4 using the two dimensional finite difference code written for this thesis. The nonlinear problem (Equation 1.2) is solved in Chapter 5 by a finite difference method that reduces to a nonlinear system of equations that is numerically solved by Newton's multivariate method. This suggests a strategy to update λ_k. In the final chapter, we look again at (Equation 1.2) by using the approach in (Equation 1.5) and experimenting with several strategies for updating λ_k.

CHAPTER 2

THE POISSON EQUATION

The Poisson equation is a partial differential equation (PDE) of elliptic type. An elliptic PDE is one that models steady states. For example, the steady state distribution of heat on a plane region whose boundary is being held at specific temperatures is modeled by an elliptic equation. Time is usually not a factor in elliptic equations and they have a broad utility in electrostatics, mechanical engineering, theoretical physics, etc.

The Poisson equation is named after the French mathematician, geometer and physicist Simon-Denis Poisson. It is a very powerful tool used for modeling the behavior of electrostatic systems, but unfortunately can be solved analytically only for very simple models (Haberman, 2004). Consequently, numerical simulation must be utilized. Although there are several algorithms for achieving this goal, one of the simplest and more straightforward of these is the finite difference method.

The solution of the Poisson equation in a domain requires the specification of certain conditions that the unknown function must satisfy at the boundary of the domain. When the function itself is specified on the boundary, we have a Dirichlet boundary condition; when the normal derivative of the function is specified on the boundary, we have a Neumann boundary condition. A problem with Neumann boundary conditions specified on the entire boundary has a unique solution up to a constant. In some problems, a linear combination of the function and its normal derivative is specified; this is known as a Robin boundary condition (Sauer, 2006).

We seek a solution to the Poisson equation subject to Dirichlet boundary conditions. That is we consider the following problem:

$$\Delta u = f \text{ in } \Omega$$

$$u = g \text{ on } \partial\Omega,$$

with f and g continuous on Ω.

2.1 Finite Difference Method for the Two-Dimensional Poisson Equation

We assume Dirichlet boundary conditions on a rectangle $[x_l, x_r]$ x $[y_b, y_t]$ in the plane. Let M and N be the number of grid steps in the x and y directions respectively. Then $h = (x_r - x_l)/M$ and $k = (y_t - y_b)/N$ are the mesh sizes in the x and y directions. The solution is known on the boundary, so the solution will need to be computed at mn points, where $m = M - 1$ and $n = N - 1$.

The Poisson equation is

$$\Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f(x, y).$$

By the definition of a derivative we have

$$\frac{\partial u}{\partial x} \approx \frac{1}{2\Delta x}[u(x + \Delta x, y) - u(x - \Delta x, y)].$$

Then the second derivative is

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{1}{\Delta x}[\frac{\partial u}{\partial x}(x + \Delta x/2, y) - \frac{\partial u}{\partial x}(x - \Delta x/2, y)].$$

Applying the definition of a derivative again we get

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{1}{\Delta x}[\frac{u(x+\Delta x, y) - u(x, y)}{\Delta x} - \frac{u(x, y) - u(x - \Delta x, y)}{\Delta x}].$$

Which simplifies to

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{1}{\Delta x^2}[u(x-\Delta x, y) - 2u(x,y) + u(x+\Delta x, y)].$$

Similarly,

$$\frac{\partial^2 u}{\partial y^2} \approx \frac{1}{\Delta y^2}[u(x, y-\Delta y) - 2u(x,y) + u(x, y+\Delta y)].$$

Since $\Delta x = h$ and $\Delta y = k$, the finite difference approximation for the Poisson equation is

$$\frac{u(x-h,y)-2u(x,y)+u(x+h,y)}{h^2} + \frac{u(x,y-k)-2u(x,y)+u(x,y+k)}{k^2} \approx f(x,y).$$

Defining $r = h^2/k^2$ and writing in discretized form we have

$$u_{i-1,j} + u_{i+1,j} - 2(1+r)u_{ij} + r(u_{i,j-1} + u_{i,j+1}) = h^2 f(x_i, y_j),$$

where

$$x_i = x_l + ih$$

$$y_j = y_b + jk.$$

We divide the mesh points into three distinct classes depending on their four neighboring mesh points. The inner core of points have no neighbors on the boundary. They consist of u_{ij} with $1 < i < m$, $1 < j < n$. The outer ring of points have one neighbor on the boundary. This happens when exactly one of the following holds: $i = 1, i = m, j = 1$ or $j = n$. The remaining points are the four corners which have two neighbors on the boundary. These are the points: $(x_1, y_1), (x_1, y_n), (x_m, y_1)$ and (x_m, y_n).

We introduce an alternate numbering system for the approximate values. We set

$$u_{ij} = v_{i+(j-1)m},$$

which leads to the finite difference form

$$v_{i-1+(j-1)m} + v_{i+1+(j-1)m} - 2(1+r)v_{i+(j-1)m} + r(v_{i+(j-2)m} + v_{i,jm}) = h^2 f(x_i, y_j).$$

The v_k are determined by solving a matrix equation $Av = b$. The rows of the matrices A and b are filled in a straightforward way as shown in the following example.

2.1.1 Example

Apply the finite difference method with $M = N = 4$ to estimate the solution of the Poisson equation $\Delta u = f$ on $[0, 1]$ x $[0, 1]$ with the following Dirichlet boundary conditions:

$$u(x, 0) = x^4 \text{ for } 0 \leq x \leq 1$$

$$u(x, 1) = (x^2 + 1)^2 \text{ for } 0 \leq x \leq 1$$

$$u(0, y) = y^4 \text{ for } 0 \leq y \leq 1$$

$$u(1, y) = (1 + y^2)^2 \text{ for } 0 \leq y \leq 1.$$

We will use the correct solution $u(x, y) = (x^2 + y^2)^2$ to compare with the approximation at the grid points for a finite difference based on a nine point stencil. Here the right hand side of the Poisson equation is $f(x_i, y_j) = 16(x_i^2 + y_j^2)$. Since $M = N = 4$, the mesh sizes $h = k = 1/4$ and $r = h^2/k^2 = 1$.

The only inner core value is u_{22} which corresponds to v_5. Its finite difference equation is

$$u_{12} + u_{32} - 2(1+r)u_{22} + r(u_{21} + u_{23}) = h^2 f(x_2, y_2).$$

In the alternate numbering system, this equation is

$$v_4 + v_6 - 2(1+r)v_5 + r(v_2 + v_8) = h^2 f(x_2, y_2).$$

The corresponding row of the 9×9 matrix A is

[0 r 0 1 -2(1+r) 1 0 r 0].

The corresponding row of the 9×1 matrix b is $h^2 f(x_2, y_2)$. Thus, we get

$$h^2 f(x_2, y_2) = h^2[16(x_2^2 + y_2^2)] = 1/16[16((1/2)^2 + (1/2)^2)] = 1/2.$$

If we look at one of the outer ring values u_{21} which corresponds to v_2, it appears in the equation

$$u_{11} + u_{31} - 2(1+r)u_{21} + r(u_{20} + u_{22}) = h^2 f(x_2, y_1).$$

Because u_{20} lies on the boundary, it takes the boundary value:

$$u_{20} = u(x_2, y_0) = u(1/2, 0) = 1/16.$$

So the finite difference equation becomes

$$u_{11} + u_{31} - 2(1+r)u_{21} + ru_{22} = h^2 f(x_2, y_1) - r/16.$$

In the alternate numbering system, this equation is

$$v_1 + v_3 - 2(1+r)v_2 + rv_5 = h^2 f(x_2, y_1) - r/16.$$

The corresponding row of the 9×9 matrix A is

[1 -2(1+r) 1 0 1 0 0 0 0].

The corresponding row of the 9×1 matrix b is $h^2 f(x_2, y_2) - r/16$. Thus, we get

$$h^2 f(x_2, y_1) - r/16 = h^2[16(x_2^2 + y_1^2)] - r/16 = 1/16[16(1/4 + 1/16)] - 1/16 = 1/4.$$

We continue this pattern to find the other rows in both matrices A and b and end up with the following matrix equation:

$$\begin{bmatrix} -4 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & -4 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & -4 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & -4 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & -4 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & -4 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & -4 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & -4 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & -4 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \\ v_8 \\ v_9 \end{bmatrix} = \begin{bmatrix} 0.11719 \\ 0.25000 \\ -0.82031 \\ 0.25000 \\ 0.50000 \\ -0.75000 \\ -0.82031 \\ -0.75000 \\ -3.75781 \end{bmatrix}.$$

Solving the matrix equation $Av = b$ we obtain the following nine values for u:

$$\begin{bmatrix} 0.02637 & 0.11133 & 0.40137 \\ 0.11133 & 0.26758 & 0.67383 \\ 0.40137 & 0.67383 & 1.27637 \end{bmatrix}.$$

It compares well with the exact solution at the same points:

$$\begin{bmatrix} 0.01562 & 0.09766 & 0.39062 \\ 0.09766 & 0.25000 & 0.66016 \\ 0.39062 & 0.66016 & 1.26562 \end{bmatrix}.$$

We get a maximum error over the grid points of 0.017578. To decrease our maximum error we must also decrease the size of our mesh by increasing M and N. We wrote an Octave program possion2d.m to solve any two dimensional Poisson equation given the right hand side of the Poisson equation, f, and exact solution, g. We show the numerical results for the Poisson equation for increasing M and N in the following section.

2.1.2 Numerical Results

We let i be a number where $i \geq 2$ and $M = N = 2^i$. We get the following error table for the finite difference method for a two-dimensional Poisson equation.

i	$M = N$	$h = k$	error	convergence rate
2	4	1/4	0.017578	
3	8	1/8	0.0045489	1.950181307
4	16	1/16	0.0011476	1.986897847
5	32	1/32	0.0002876	1.996484292
6	64	1/64	0.0000719	2

2.2 Finite Difference Method for the Three-Dimensional Poisson Equation

We assume Dirichlet boundary conditions on a cube $[x_1, x_2]$ x $[y_1, y_2]$x $[z_1, z_2]$. Let M, N and P be the number of grid steps in the x, y and z directions respectively. Then $h = (x_2 - x_1)/M$, $k = (y_2 - y_1)/N$ and $c = (z_2 - z_1)/P$ are the mesh sizes in the x, y and z directions. The solution is known on the boundary, so the solution will need to be computed at mnp points, where $m = M - 1$, $n = N - 1$ and $p = P - 1$.

The Poisson equation is

$$\Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} = f(x, y, z).$$

By the definition of a derivative we have

$$\frac{\partial^2 u}{\partial x^2} \approx \frac{1}{\Delta x^2}[u(x - \Delta x, y, z) - 2u(x, y, z) + u(x + \Delta x, y, z)],$$

$$\frac{\partial^2 u}{\partial y^2} \approx \frac{1}{\Delta y^2}[u(x, y - \Delta y, z) - 2u(x, y, z) + u(x, y + \Delta y, z)],$$

and

$$\frac{\partial^2 u}{\partial z^2} \approx \frac{1}{\Delta z^2}[u(x, y, z - \Delta z) - 2u(x, y, z) + u(x, y, z + \Delta z)].$$

Since $\Delta x = h$, $\Delta y = k$ and $\Delta z = c$ the finite difference approximation for the Poisson equation is

$$\frac{u(x-h,y,z)-2u(x,y,z)+u(x+h,y,z)}{h^2} + \frac{u(x,y-k,z)-2u(x,y,z)+u(x,y+k,z)}{k^2} + \frac{u(x,y,z-c)-2u(x,y,z)+u(x,y,z+c)}{c^2} \approx f(x, y, z).$$

Defining $r = h^2/k^2$, $q = h^2/c^2$ and writing in discretized form we have

$$u_{i-1,j,l} + u_{i+1,j,l} - 2(1 + r + q)u_{ijl} + r(u_{i,j-1,l} + u_{i,j+1,l}) + q(u_{i,j,l-1} + u_{i,j,l+1}) = h^2 f(x_i, y_j, z_l),$$

where

$$x_i = x_1 + ih$$

$$y_j = y_1 + jk$$

$$z_l = z_1 + lc.$$

We divide the mesh points into four distinct classes depending on their four neighboring mesh points. The inner core of points have no neighbors on the boundary. They consist of u_{ijl} with

$1 < i < m$, $1 < j < n$ and $1 < l < p$. The first outer ring of points have one neighbor on the boundary. This happens when exactly one of the following holds: $i = 1$, $i = m$, $j = 1$, $j = n$, $l = 1$ or $l = p$. The second outer ring of points have two neighbors on the boundary. This happens when exactly two of the following holds: $i = 1$ or $i = m$, $j = 1$ or $j = n$, and $l = 1$ or $l = p$. The remaining points are the eight corners which have three neighbors on the boundary. These are the points: $(x_1, y_1, z_1), (x_1, y_n, z_1), (x_1, y_1, z_p), (x_1, y_n, z_p), (x_m, y_1, z_1), (x_m, y_n, z_1), (x_m, y_1, z_p)$ and (x_m, y_n, z_p).

We introduce an alternate numbering system for the approximate values. We set

$$u_{ijl} = v_{i+(j-1)m+(l-1)m^2},$$

which leads to the finite difference form

$$v_{i-1+(j-1)m+(l-1)m^2} + v_{i+1+(j-1)m+(l-1)m^2} - 2(1 + r + q)v_{i+(j-1)m+(l-1)m^2} +$$

$$r(v_{i+(j-2)m+(l-1)m^2} + v_{i+jm+(l-1)m^2}) + q(v_{i+(j-1)m+(l-2)m^2} + v_{i+(j-1)m+lm^2}) = h^2 f(x_i, y_j, z_l).$$

The v_k are determined by solving a matrix equation $Av = b$. The rows of the matrices A and b are filled in a straightforward way as shown in the following example.

2.2.1 Example

Apply the finite difference method with $M = N = P = 4$ to estimate the solution to the Poisson equation $\Delta u = f$ on $[0, 1]$ x $[0, 1]$ x $[0, 1]$ with the following Dirichlet boundary conditions:

$$u(x, 0, z) = x^4 \text{ for } 0 \le x \le 1,\ 0 \le z \le 1$$

$$u(x, 1, z) = (x^2 + 1)^2 \text{ for } 0 \le x \le 1,\ 0 \le z \le 1$$

$$u(0, y, z) = y^4 \text{ for } 0 \le y \le 1,\ 0 \le z \le 1$$

$$u(1, y, z) = (1 + y^2)^2 \text{ for } 0 \le y \le 1,\ 0 \le z \le 1$$

$$u(x, y, 0) = (x^2 + y^2)^2 \text{ for } 0 \le x \le 1,\ 0 \le y \le 1$$

$$u(x, y, 1) = (x^2 + y^2)^2 \text{ for } 0 \le x \le 1,\ 0 \le y \le 1.$$

We will use the correct solution $u(x, y, z) = (x^2 + y^2)^2$ to compare with the approximation at the twenty seven mesh points of the cube. Then the right hand side of the Poisson equation is $f(x_i, y_j, z_l) = 16(x_i^2 + y_j^2)$ Since $M = N = P = 4$, the mesh sizes $h = k = c = 1/4$, $r = h^2/k^2 = 1$ and $q = h^2/c^2 = 1$.

The only inner core value is u_{222} which corresponds to v_{14}. Its finite difference equation is

$$u_{122} + u_{322} - 2(1 + r + q)u_{222} + r(u_{212} + u_{232}) + q(u_{221} + u_{223}) = h^2 f(x_2, y_2, z_2).$$

In the alternate numbering system, this equation is

$$v_{13} + v_{15} - 2(1 + r + q)v_{14} + r(v_{11} + v_{17}) + q(v_5 + v_{23}) = h^2 f(x_2, y_2, z_2).$$

The corresponding row of the 27×27 matrix A is

$$[0\ 0\ 0\ 0\ q\ 0\ 0\ 0\ 0\ 0\ r\ 0\ 1\ \text{-}2(1{+}r{+}q)\ 1\ 0\ r\ 0\ 0\ 0\ 0\ 0\ q\ 0\ 0\ 0\ 0].$$

The corresponding row of the 27×1 matrix b is $h^2 f(x_2, y_2, z_2)$ where f is the right hand side of the Poisson equation. Thus, we get

$$h^2 f(x_2, y_2, z_2) = h^2[16(x_2^2 + y_2^2)] = 1/16[16((1/2)^2 + (1/2)^2)] = 1/2.$$

If we look at one of the outer ring 1 values u_{122} which corresponds to v_{13}, its finite difference equation is

$$u_{022} + u_{222} - 2(1 + r + q)u_{122} + r(u_{112} + u_{132}) + q(u_{121} + u_{123}) = h^2 f(x_2, y_2, z_2).$$

Because u_{022} lies on the boundary, it takes the boundary value:

$$u_{022} = u(x_0, y_2, z_2) = u(0, 1/2, 1/2) = 1/16.$$

So the finite difference equation becomes

$$u_{222} - 2(1 + r + q)u_{122} + r(u_{112} + u_{132}) + q(u_{121} + u_{123}) = h^2 f(x_2, y_2, z_2) - 1/16.$$

In the alternate numbering system, this equation is

$$v_{14} - 2(1 + r + q)v_{13} + r(v_{10} + v_{16}) + q(v_4 + v_{22}) = h^2 f(x_2, y_2, z_2) - 1/16.$$

The corresponding row of the 27×27 matrix A is

$$[0\ 0\ 0\ q\ 0\ 0\ 0\ 0\ 0\ r\ 0\ 0\ \text{-}2(1{+}r{+}q)\ 1\ 0\ r\ 0\ 0\ 0\ 0\ 0\ q\ 0\ 0\ 0\ 0\ 0].$$

The corresponding row of the 27×1 matrix b is $h^2 f(x_1, y_2, z_2) - 1/16$. Thus, we get

$$h^2 f(x_1, y_2, z_2) - 1/16 = h^2[16(x_1^2 + y_2^2)] - 1/16 = 1/16[16(1/16 + 1/4)] - 1/16 = 1/4.$$

We continue this pattern to find the other rows in both matrices A and b. We end up with the matrix equation $Av = b$ where A is a 27×27 square matrix with negative sixes on the diagonal and ones on the tridiagonal. Solving the matrix equation we obtain the following twenty seven approximate values for u.

For $z = 1$, the numerical solution is:

$$\begin{bmatrix} 0.02237 & 0.10593 & 0.39737 \\ 0.10593 & 0.26026 & 0.66843 \\ 0.39737 & 0.66843 & 1.27237 \end{bmatrix}.$$

For $z = 2$, the numerical solution is:

$$\begin{bmatrix} 0.02390 & 0.10792 & 0.39890 \\ 0.10792 & 0.26287 & 0.67042 \\ 0.39890 & 0.67042 & 1.27390 \end{bmatrix}.$$

For $z = 3$, the numerical solution is:

$$\begin{bmatrix} 0.02237 & 0.10593 & 0.39737 \\ 0.10593 & 0.26026 & 0.66843 \\ 0.39737 & 0.66843 & 1.27237 \end{bmatrix}.$$

It compares well with the exact solution at the same points.

For $z = 1$, the numerical solution is:

$$\begin{bmatrix} 0.01562 & 0.09766 & 0.39062 \\ 0.09766 & 0.25000 & 0.66016 \\ 0.39062 & 0.66016 & 1.26562 \end{bmatrix}.$$

For $z = 2$, the numerical solution is:

$$\begin{bmatrix} 0.01562 & 0.09766 & 0.39062 \\ 0.09766 & 0.25000 & 0.66016 \\ 0.39062 & 0.66016 & 1.26562 \end{bmatrix}.$$

For $z = 3$, the numerical solution is:

$$\begin{bmatrix} 0.01562 & 0.09766 & 0.39062 \\ 0.09766 & 0.25000 & 0.66016 \\ 0.39062 & 0.66016 & 1.26562 \end{bmatrix}.$$

We get a maximum error over the grid points of 0.012868. To decrease our maximum error we must also decrease the size of our mesh by increasing M, N and P. We wrote an Octave program possion3d.m to solve any three dimensional Poisson equation given the right hand side of the Poisson equation, f, and exact solution, g. We show the numerical results for the Poisson equation for increasing M, N and P in the following section.

2.2.2 Numerical Results

We let i be a number where $i \geq 2$ and $M = N = P = 2^i$. We get the following error table for the finite difference method for a three-dimensional Poisson equation.

i	$M = N = P$	$h = k = c$	error	convergence rate
2	4	1/4	0.012868	
3	8	1/8	0.0034324	1.906498247
4	16	1/16	0.0008731	1.974998883

CHAPTER 3

THE LAPLACIAN EIGENVALUE PROBLEM

The Laplace operator or Laplacian is a differential operator given by the divergence of the gradient of a function on the Euclidean space. In a Cartesian coordinate system, the Laplacian of a function is given by the sum of second partial derivatives of the function with respect to each independent variable.

The Laplace operator is named after the French mathematician Pierre-Simon de Laplace (1749 - 1827), who first applied the operator to the study of celestial mechanics. The Laplacian occurs in differential equations that describe many physical phenomena, such as electric and gravitational potentials, the diffusion equation for heat and fluid flow, wave propagation, and quantum mechanics (Sauer, 2006). The Laplacian eigenvalue problem is

$$\Delta u = \lambda u \text{ in } \Omega$$
$$u = 0 \text{ on } \partial\Omega.$$

We will solve this problem using a finite difference method on the domain $\Omega = [0,1]^m$ that will be reduced to a simple matrix eigenvalue problem. We start by introducing the matrix eigenvalue problem.

3.1 The Matrix Eigenvalue Problem

The matrix eigenvalue problem has many applications in both pure and applied mathematics. It is used in matrix factorization, quantum mechanics, and many other areas. The problem states that an eigenvector of a square matrix A is a non-zero vector x that when multiplied by A, yields the original vector multiplied by a single number λ; that is:

$$Ax = \lambda x \ (x \neq 0).$$

The number λ is called the eigenvalue of A corresponding to the eigenvector x.

It can be shown that if A is an $n \times n$ matrix, then det(A - λI) is a polynomial in the variable λ of degree n. We call this polynomial the characteristic polynomial of A. To find the eigenvalues of A we must find the roots of the characteristic polynomial. Thus we must solve the equation:

$$\det(A \text{ - } \lambda I) = 0.$$

To find the eigenvectors corresponding to each eigenvalue, we need to solve the equation

$$(A \text{ - } \lambda I)x = 0,$$

for each λ.

3.1.1 Example

We want to find the eigenvalues and eigenvectors of the following matrix. Consider the following matrix:

$$A = \begin{bmatrix} 2 & 2 \\ 5 & -1 \end{bmatrix}.$$

To find the eigenvalues of A, we must compute det(A - λI), set this expression equal to 0, and solve for λ. Note that:

$$A - \lambda I = \begin{bmatrix} 2-\lambda & 2 \\ 5 & -1-\lambda \end{bmatrix}.$$

The determinant of this 2 x 2 matrix is given by:

$$\det(A - \lambda I) = \lambda^2 - \lambda - 12.$$

Solving the characteristic equation $\lambda^2 - \lambda - 12 = 0$ for λ, we get that $\lambda = -3$ or 4. These are the two eigenvalues of A.

Now we want to find the eigenvectors of A so we want to solve$(A - \lambda I)x = 0$ when $\lambda = -3$ or 4. Note that if $\det(A - \lambda I) = 0$ then the equation $(A - \lambda I)x = b$ has either no solution or infinitely many. When we take $b = 0$ however, it is clear by the existence of the solution $x = 0$ that there are infinitely many solutions (i.e. we may rule out the "no solution" case). If we continue using the matrix A from the example above, we can expect infinitely many nonzero solutions of the equation $Ax = \lambda x$ precisely when $\lambda = -3$ or $\lambda = 4$. Let us proceed to characterize such solutions.

To find the eigenvectors corresponding to $\lambda = -3$, we need to solve:

$$(A + 3I)x = 0.$$

This becomes

$$Ax = -3x \text{ where } x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}.$$

So we get the matrix equation

$$\begin{bmatrix} 2x_1 + 2x_2 \\ 5x_1 - x_2 \end{bmatrix} = \begin{bmatrix} -3x_1 \\ -3x_2 \end{bmatrix}.$$

Here we have two equations and two unknowns. We use elementary algebra techniques to solve and get that $x_1 = -2/5x_2$. This means that, while there are infinitely many nonzero solutions (solution vectors) of the equation $Ax = 3x$, they all satisfy the condition that the first entry x_1 is $-2/5$ times the second entry x_2. Thus all solutions of this equation can be characterized by

$$\begin{bmatrix} 2t \\ \\ -5t \end{bmatrix} = t \begin{bmatrix} 2 \\ \\ -5 \end{bmatrix},$$

where t is any real number. The nonzero vectors x that satisfy $Ax = -3x$ are the eigenvectors associated with the eigenvalue $\lambda = -3$. One such eigenvector is

$$u_1 = \begin{bmatrix} 2 \\ \\ -5 \end{bmatrix},$$

and all other eigenvectors corresponding to the eigenvalue $\lambda = -3$ are simply scalar multiples of u_1. That is u_1 spans this set of eigenvectors.

Similarly, we can find eigenvectors associated with the eigenvalue $\lambda = 4$ by solving $Ax = 4x$

$$\begin{bmatrix} 2x_1 + 2x_2 \\ \\ 5x_1 - x_2 \end{bmatrix} = \begin{bmatrix} 4x_1 \\ \\ 4x_2 \end{bmatrix},$$

which reduces to $x_1 = x_2$. Hence the set of eigenvectors associated with $\lambda = 4$ is spanned by

$$u_2 = \begin{bmatrix} 1 \\ \\ 1 \end{bmatrix}.$$

Note that this can be done numerically using the Octave command $[V, D] = eig(A)$. This command returns two outputs that satisfy the equation, $Ax = \lambda x$ where A is an $n \times n$ matrix, x is a column vector of length n, and λ is a scalar. The output matrix V is a matrix whose columns are eigenvectors of A. The output matrix D is a diagonal matrix containing the eigenvalues of A along the main diagonal. We will use this command to numerically solve the Laplacian eigenvalue problem in the following section.

3.2 Finite Difference Method for the Two-Dimensional Laplacian Eigenvalue Problem

We want to use a finite difference method to solve the two-dimensional Laplacian eigenvalue problem.

$$\Delta u = \lambda u \text{ in } \Omega$$

$$u = 0 \text{ on } \partial\Omega.$$

We assume Dirichlet boundary conditions on a rectangle $[x_l, x_r]$ x $[y_b, y_t]$ in the plane. Let M and N be the number of grid steps in the x and y directions respectively. Then $h = (x_r - x_l)/M$ and $k = (y_t - y_b)/N$ are the mesh sizes in the x and y directions. So the equations will be solved on the same rectangular mesh described in Section 2.1.

The Laplacian eigenvalue problem

$$\Delta u = \tfrac{\partial^2 u}{\partial x^2} + \tfrac{\partial^2 u}{\partial y^2} = \lambda u(x, y),$$

has the following finite difference form:

$$\tfrac{u(x-h,y)-2u(x,y)+u(x+h,y)}{h^2} + \tfrac{u(x,y-k)-2u(x,y)+u(x,y+k)}{k^2} \approx \lambda u(x, y).$$

Defining $r = h^2/k^2$ and writing in discretized form we have

$$u_{i-1,j} + u_{i+1,j} - 2(1 + r)u_{ij} + r(u_{i,j-1} + u_{i,j+1}) = h^2 \lambda u_{ij}.$$

We use the alternate numbering system introduced in Section 2.1 and set

$$u_{ij} = v_{i+(j-1)m},$$

which has the finite difference form

$$v_{i-1+(j-1)m} + v_{i+1+(j-1)m} - 2(1 + r)v_{i+(j-1)m} + r(v_{i+(j-2)m} + v_{i,jm}) = h^2 \lambda v_{i+(j-1)m}.$$

The v_k and λ are then determined by solving the matrix eigenvalue problem $(1/h^2 A)v = \lambda v$ as shown in the following example.

3.2.1 Example

Apply the finite difference method with $M = N = 4$ to estimate the solution to the Laplace eigenvalue problem $\Delta u = \lambda u$ on $[0, 1]$ x $[0, 1]$. Since $M = N = 4$, the mesh sizes $h = k = 1/4$ and $r = h^2/k^2 = 1$. For this type of problem, $u = 0$ on the boundary thus we will use the correct solution $u(x, y) = \sin(\pi x) \sin(\pi y)$ to compare with the approximation at the nine mesh points of the square. We can input our correct solution into the Laplacian eigenvalue problem to find λ_{exact} which will be used for comparison. Since

$$u_{xx} = -\pi^2 \sin(\pi x)\sin(\pi y),$$
$$u_{yy} = -\pi^2 \sin(\pi x)\sin(\pi y),$$

we have

$$\Delta u = u_{xx} + u_{yy} = \lambda u$$
$$-2\pi^2 \sin(\pi x)\sin(\pi y) = \lambda \sin(\pi x)\sin(\pi y).$$

Thus,

$$\lambda_{exact} = -2\pi^2 \approx -19.7392088.$$

We also want to find $u_{exact} = u((x_l + ih), (y_b + jk))$ over the grid points. We compute this and get the following vector:

$$u_{exact} = \begin{bmatrix} 0.50000 \\ 0.70711 \\ 0.50000 \\ 0.70711 \\ 1.00000 \\ 0.70711 \\ 0.50000 \\ 0.70711 \\ 0.50000 \end{bmatrix}.$$

Now we look at our numerical solution. The only inner core value is u_{22} which corresponds to v_5. The corresponding finite difference equation is

$$1/h^2(u_{12} + u_{32} - 2(1 + r)u_{22} + r(u_{21} + u_{23})) = \lambda u_{22}.$$

In the alternate numbering system, this equation is

$$1/h^2(v_4 + v_6 - 2(1 + r)v_5 + r(v_2 + v_8)) = \lambda v_5.$$

The corresponding row of the 9 x 9 matrix A is

$$1/h^2[0\ r\ 0\ 1\ \text{-}2(1{+}r)\ 1\ 0\ r\ 0].$$

If we look at one of the outer ring values u_{21} which corresponds to v_2, it appears in the equation

$$1/h^2(u_{11} + u_{31} - 2(1 + r)u_{21} + r(u_{20} + u_{22})) = \lambda u_{21}.$$

Because u_{20} lies on the boundary, it takes the boundary value:

$$u_{20} = 0.$$

So the finite difference equation becomes

$$1/h^2(u_{11} + u_{31} - 2(1 + r)u_{21} + ru_{22}) = \lambda u_{21}.$$

In the alternate numbering system, this equation is

$$1/h^2(v_1 + v_3 - 2(1 + r)v_2 + rv_5) = \lambda v_2.$$

The corresponding row of the 9 x 9 matrix A is

$$1/h^2[1\ \text{-}2(1{+}r)\ 1\ 0\ 1\ 0\ 0\ 0\ 0].$$

Notice that the left hand side of the equation is the same as in Example 2.1 scaled by a factor of $1/h^2 = 16$. Thus, we take 9 x 9 matrix A and multiply it by 16. We have the following matrix eigenvalue problem

$$\begin{bmatrix} -64 & 16 & 0 & 16 & 0 & 0 & 0 & 0 & 0 \\ 16 & -64 & 16 & 0 & 16 & 0 & 0 & 0 & 0 \\ 0 & 16 & -64 & 0 & 0 & 16 & 0 & 0 & 0 \\ 16 & 0 & 0 & -64 & 16 & 0 & 16 & 0 & 0 \\ 0 & 16 & 0 & 16 & -64 & 16 & 0 & 16 & 0 \\ 0 & 0 & 16 & 0 & 16 & -64 & 0 & 0 & 16 \\ 0 & 0 & 0 & 16 & 0 & 0 & -64 & 16 & 0 \\ 0 & 0 & 0 & 0 & 16 & 0 & 16 & -64 & 16 \\ 0 & 0 & 0 & 0 & 0 & 16 & 0 & 16 & -64 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \\ v_8 \\ v_9 \end{bmatrix} = \lambda \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \\ v_8 \\ v_9 \end{bmatrix}.$$

We can easily solve this matrix eigenvalue problem using the method described in Section 3.1 and we get the following

$$\lambda_k = \begin{bmatrix} -109.255 \\ -86.627 \\ -86.627 \\ -64.000 \\ -64.000 \\ -64.000 \\ -41.373 \\ -41.373 \\ -18.745 \end{bmatrix},$$

and

$$v_k = \begin{bmatrix} 0.25000 & -0.21758 & 0.45018 & 0.50662 & 0.00000 & 0.34400 & -0.17054 & -0.47002 & 0.25000 \\ -0.35355 & -0.16447 & -0.47218 & -0.05387 & 0.49072 & 0.07934 & 0.21176 & -0.45294 & 0.35355 \\ 0.25000 & 0.45018 & 0.21758 & 0.14943 & 0.11073 & -0.58344 & 0.47002 & -0.17054 & 0.25000 \\ -0.35355 & 0.47218 & -0.16447 & 0.05387 & -0.49072 & -0.07934 & -0.45294 & -0.21176 & 0.35355 \\ 0.50000 & -0.00000 & 0.00000 & -0.65605 & -0.11073 & 0.23944 & -0.00000 & 0.00000 & 0.50000 \\ -0.35355 & -0.47218 & 0.16447 & 0.05387 & -0.49072 & -0.07934 & 0.45294 & 0.21176 & 0.35355 \\ 0.25000 & -0.45018 & -0.21758 & 0.14943 & 0.11073 & -0.58344 & -0.47002 & 0.17054 & 0.25000 \\ -0.35355 & 0.16447 & 0.47218 & -0.05387 & 0.49072 & 0.07934 & -0.21176 & 0.45294 & 0.35355 \\ 0.25000 & 0.21758 & -0.45018 & 0.50662 & 0.00000 & 0.34400 & 0.17054 & 0.47002 & 0.25000 \end{bmatrix},$$

where λ_k are the eigenvalues of A and the matrix v_k are the corresponding column eigenvectors of A.

Notice that the numerical solution λ_9 and v_9 corresponds to our exact solution. Here v_9 is in the span of our exact eigenvector since $u_{exact} = 2v_1$. Also, λ_9 is a good approximation for the exact eigenvalue since $\lambda_{exact} \approx \lambda_1 - 1$. However, we want to decrease the error on λ so in the following section we compute the numerical solutions to the Laplacian eigenvalue problem over smaller mesh sizes. We wrote an Octave program lapacian2deig.m to solve any two dimensional Laplacian eigenvalue problem given the exact solution, g.

3.2.2 Numerical Results

We let i be a number where $i \geq 2$ and $M = N = 2^i$. We get the following error table for the finite difference method for a two-dimensional Laplacian eigenvalue problem.

i	$M = N$	$h = k$	λ	$error_\lambda$	convergence rate
2	4	1/4	-18.745	0.994208802	
3	8	1/8	-19.487	0.252208802	1.97893025
4	16	1/16	-19.68	0.059208802	2.090735058
5	32	1/32	-19.72	0.019208802	1.624044118
6	64	1/64	-19.735	0.004208802	2.190285999

3.3 Finite Difference Method for the Three-Dimensional Laplacian Eigenvalue Problem

We want to use a finite difference method to solve the three-dimensional Laplacian eigenvalue problem.

$$\Delta u = \lambda u \text{ in } \Omega$$

$$u = 0 \text{ on } \partial\Omega.$$

We assume Dirichlet boundary conditions on a cube $[x_1, x_2]$ x $[y_1, y_2]$ x $[z_1, z_2]$. Let M, N and P be the number of grid steps in the x, y and z directions respectively. Then $h = (x_2 - x_1)/M$, $k = (y_2 - y_1)/N$ and $c = (z_2 - z_1)/P$ are the mesh sizes in the x, y and z directions. So the equations will be solved on the same cubic mesh described in Section 2.2.

The Laplacian eigenvalue problem

$$\Delta u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} = \lambda u(x, y, z),$$

has the following finite difference form:

$$\frac{u(x-h,y,z)-2u(x,y,z)+u(x+h,y,z)}{h^2} + \frac{u(x,y-k,z)-2u(x,y,z)+u(x,y+k,z)}{k^2} + \frac{u(x,y,z-c)-2u(x,y,z)+u(x,y,z+c)}{c^2} \approx$$

$$\lambda u(x,y,z).$$

Defining $r = h^2/k^2$, $q = h^2/c^2$ and writing in discretized form we have

$$u_{i-1,j,l} + u_{i+1,j,l} - 2(1+r+q)u_{ijl} + r(u_{i,j-1,l} + u_{i,j+1,l}) + q(u_{i,j,l-1} + u_{i,j,l+1}) = h^2\lambda u_{ijl}.$$

We use the alternate numbering system introduced in Section 2.2 and set

$$u_{ijl} = v_{i+(j-1)m+(l-1)m^2},$$

which has the finite difference form

$$v_{i-1+(j-1)m+(l-1)m^2} + v_{i+1+(j-1)m+(l-1)m^2} - 2(1+r+q)v_{i+(j-1)m+(l-1)m^2} + r(v_{i+(j-2)m+(l-1)m^2} +$$

$$v_{i+jm+(l-1)m^2}) + q(v_{i+(j-1)m+(l-2)m^2} + v_{i+(j-1)m+lm^2}) = h^2\lambda v_{i+(j-1)m+(l-1)m^2}.$$

The v_k and λ are determined by solving the matrix eigenvalue problem $(1/h^2 A)v = \lambda v$ as shown in the following example.

3.3.1 Example

Apply the finite difference method with $M = N = P = 4$ to estimate the solution to the Laplace eigenvalue problem $\Delta u = \lambda u$ on $[0,1]$ x $[0,1]$ x $[0,1]$. Since $M = N = P = 4$, the mesh sizes $h = k = 1/4$, $r = h^2/k^2 = 1$ and $q = h^2/c^2 = 1$. For this type of problem, $u = 0$ on the boundary thus we will use the correct solution $u(x,y,z) = \sin(\pi x)\sin(\pi y)\sin(\pi z)$ to compare with the approximation at the twenty seven mesh points of the cube. We can input our correct solution into the Laplacian eigenvalue problem to find λ_{exact} which will be used for comparison. Since

$$u_{xx} = -\pi^2 \sin(\pi x)\sin(\pi y)\sin(\pi z)$$
$$u_{yy} = -\pi^2 \sin(\pi x)\sin(\pi y)\sin(\pi z)$$
$$u_{zz} = -\pi^2 \sin(\pi x)\sin(\pi y)\sin(\pi z),$$

we have

$$\Delta u = u_{xx} + u_{yy} + u_{zz} = \lambda u$$
$$-3\pi^2 \sin(\pi x)\sin(\pi y)\sin(\pi z) = \lambda \sin(\pi x)\sin(\pi y)\sin(\pi z).$$

Thus,

$$\lambda_{exact} = -3\pi^2 \approx -29.6088132.$$

We also want to find $u_{exact} = u((x_1 + ih), (y_1 + jk), (z_1 + lc))$ over the grid points. We compute this and get the following:

$$u_{exact} = \begin{bmatrix} 0.35355 \\ 0.50000 \\ ... \\ 0.50000 \\ 0.35355 \end{bmatrix}.$$

Now we look at our numerical solution. The only inner core value is u_{22} which corresponds to v_5. Its finite difference equation is

$$1/h^2(u_{122} + u_{322} - 2(1 + r + q)u_{222} + r(u_{212} + u_{232}) + q(u_{221} + u_{223})) = \lambda u_{22}.$$

In the alternate numbering system, this equation is

$$1/h^2(v_{13}+v_{15}-2(1+r+q)v_{14}+r(v_{11}+v_{17})+q(v_5+v_{23}))=\lambda v_5.$$

The corresponding row of the 27×27 matrix A is

$$1/h^2[0\ 0\ 0\ 0\ q\ 0\ 0\ 0\ 0\ 0\ r\ 0\ 1\ \text{-2(1+}r\text{+}q\text{)}\ 1\ 0\ r\ 0\ 0\ 0\ 0\ 0\ q\ 0\ 0\ 0\ 0].$$

If we look at one of the outer ring 1 values u_{122} which corresponds to v_{13}, its finite difference equation is

$$1/h^2(u_{022}+u_{222}-2(1+r+q)u_{122}+r(u_{112}+u_{132})+q(u_{121}+u_{123}))=\lambda u_{122}.$$

Because u_{022} lies on the boundary, it takes the boundary value:

$$u_{022}=0.$$

So the finite difference equation becomes

$$u_{222}-2(1+r+q)u_{122}+r(u_{112}+u_{132})+q(u_{121}+u_{123})=\lambda u_{122}.$$

In the alternate numbering system, this equation is

$$v_{14}-2(1+r+q)v_{13}+r(v_{10}+v_{16})+q(v_4+v_{22})=\lambda v_{13}.$$

The corresponding row of the 27×27 matrix A is

$$1/h^2[0\ 0\ 0\ q\ 0\ 0\ 0\ 0\ 0\ r\ 0\ 0\ \text{-2(1+}r\text{+}q\text{)}\ 1\ 0\ r\ 0\ 0\ 0\ 0\ 0\ q\ 0\ 0\ 0\ 0\ 0].$$

Notice that the left hand side of the equation is the same as matrix A in Example 2.2 scaled by a factor of $1/h^2 = 16$. Thus, we take 27×27 matrix A and multiply it by 16. Now we have a simple matrix eigenvalue problem. When solved using the methods described in Section 3.1 we get the following:

$$\lambda_k = \begin{bmatrix} -163.882 \\ -141.255 \\ ... \\ -50.745 \\ -28.118 \end{bmatrix},$$

and

$$v_k = \begin{bmatrix} 0.12500 & ... & -0.12500 \\ -0.17678 & ... & -0.17678 \\ ... & ... & ... \\ -0.17678 & ... & -0.17678 \\ 0.12500 & ... & -0.12500 \end{bmatrix},$$

where λ_k are the eigenvalues of A and the matrix v_k are the corresponding column eigenvectors of A.

Notice that the numerical solution λ_{27} and v_{27} corresponds to our exact solution. Here v_{27} is in the span of our exact eigenvector since $u_{exact} = -2.828 v_{27}$. Also, λ_{27} is a good approximation for the exact eigenvalue since $\lambda_{exact} \approx \lambda_1 - 1.5$. However, we want to decrease the error on λ so in the following section we compute the numerical solutions to the Laplacian eigenvalue problem over smaller mesh sizes. We wrote an Octave program lapacian3deig.m to solve any three dimensional Laplacian eigenvalue problem given the exact solution, g.

3.3.2 Numerical Results

We let i be a number where $i \geq 2$ and $M = N = P = 2^i$. We get the following error table for the finite difference method for a three-dimensional Laplacian eigenvalue problem.

i	$M = N = P$	$h = k = c$	λ	$error_\lambda$	convergence rate
2	4	1/4	= -28.118	1.4911	
3	8	1/8	-29.230	0.37855	1.977821241
4	16	1/16	-29.514	0.095004	1.994423706

CHAPTER 4

THE MONGE-AMPÈRE EQUATION

The elliptic Monge-Ampère equation is a fully nonlinear partial differential equation (PDE) first described in the late eighteenth century. Since then, the equation has arisen in a number of important applications and the associated regularity theory has received a great deal of attention. Despite the importance of the Monge-Ampère equation, until recently, very little progress had been made in actually solving the equation numerically.

The last several years have seen an explosion of interest in numerical methods for solving this and other fully nonlinear PDEs. Several methods have been developed for approximating solutions of the Monge-Ampère equation (Awanou, 2010a). However, the richness and complexity of the equation also lead to a number of important challenges that place limitations on these numerical methods. Consequently, the development of numerical methods for this PDE remains a challenging problem. The development of methods powerful enough to handle these challenges would have important implications for several interesting applications.

The Monge-Ampère operator is given by

$$S_n(D^2u).$$

4.1 Finite Difference Method for the Two-Dimensional Monge-Ampère Equation

We want to write a finite difference code to solve the two dimensional Monge-Ampère equation

$$\det D^2 u = f \text{ in } \Omega$$

$$u = g \text{ on } \partial\Omega,$$

by an iterative method with initial guess

$$\Delta u_0 = 2\sqrt{f} \text{ in } \Omega$$

$$u_0 = g \text{ on } \partial\Omega.$$

We assume Dirichlet boundary conditions on a rectangle $[x_l, x_r]$ x $[y_b, y_t]$ in the plane. Let M and N be the number of grid steps in the x and y directions respectively. Then $h = (x_r - x_l)/M$ and $k = (y_t - y_b)/N$ are the mesh sizes in the x and y directions. The equations will be solved on a rectangular mesh of points. The solution is known on the boundary, so the solution will need to be computed at mn points, where $m = M - 1$ and $n = N - 1$.

We start by looking at the two dimensional Hessian matrix, D^2u, which is a square matrix of second-order partial derivatives of $u(x, y)$. We have

$$D^2 u = \begin{bmatrix} u_{xx} & u_{xy} \\ u_{yx} & u_{yy} \end{bmatrix},$$

and the determinant is

$$\det D^2 u = u_{xx}u_{yy} - u_{xy}^2.$$

The second order partial derivatives have the following finite difference form:

$$(u_{xx})_{ij} \approx 1/h^2(u_{i+1,j} - 2u_{i,j} + u_{i-1,j})$$

$$(u_{yy})_{ij} \approx 1/h^2(u_{i,j+1} - 2u_{i,j} + u_{i,j-1})$$

$$(u_{xy})_{ij} \approx 1/4h^2(u_{i+1,j+1} + u_{i-1,j-1} - u_{i-1,j+1} - u_{i+1,j-1}).$$

Our goal is to solve the equation

$$(\det D^2 u)_{ij} = (u_{xx})_{ij}(u_{yy})_{ij} - (u_{xy})^2_{ij} = f(x_i, y_j),$$

where $x_i = x_l + ih$, $y_j = y_b + jk$.

We divide the mesh points into three distinct classes described in Section 2.1. We compute the determinant, $\det D^2 u$, at each of the mesh points and use the iterative method

$$\Delta u_{p+1} = \Delta u_p - 1/\nu(\det D^2 u_p - f),$$

to solve the Monge-Ampère equation.

4.1.1 Example

Apply the finite difference method with $M = N = 4$ to estimate the solution to the Monge-Ampère equation on $[0,1]$ x $[0,1]$ with the following Dirichlet boundary conditions:

$$u(x,0) = x^4 \text{ for } 0 \le x \le 1$$

$$u(x,1) = (x^2+1)^2 \text{ for } 0 \le x \le 1$$

$$u(0,y) = y^4 \text{ for } 0 \le y \le 1$$

$$u(1,y) = (1+y^2)^2 \text{ for } 0 \le y \le 1.$$

We will use the correct solution $u(x,y) = (x^2+y^2)^2$ to compare with the approximation at the nine mesh points of the square. We can input our correct solution into the Monge-Ampère equation to find the right hand side which will be used for comparison.

$$\det D^2u(x,y) = u_{xx}u_{yy} - u_{xy}^2.$$

Since $u_{xx} = 12x^2 + 4y^2$, $u_{yy} = 12y^2 + 4x^2$ and $u_{xy} = 8xy$ we get

$$\det D^2u(x,y) = 48(x_i^2 + y_j^2)^2 = 48u.$$

Now we look at our numerical solution. We start by using our two dimensional Poisson solver poisson2d.m with right hand side $2\sqrt{f} = 2\sqrt{16(x^2+y^2)}$ to find our initial guess u_0. We get

$$u_0 = \begin{bmatrix} 0.06822 & 0.22092 & 0.54862 \\ 0.22092 & 0.48384 & 0.92354 \\ 0.54862 & 0.92354 & 1.54989 \end{bmatrix}.$$

We can now find the determinant of u_0 by computing the second partial derivatives at the nine mesh points.

The only inner core value is $u_{0_{22}}$ which has the following finite difference equations for its partial derivatives

$$u_{0_{xx}}(2,2) = 1/h^2(u_{3,2} - 2u_{2,2} + u_{1,2}) = 2.82843$$

$$u_{0_{yy}}(2,2) = 1/h^2(u_{2,3} - 2u_{2,2} + u_{2,1}) = 2.82843$$

$$u_{0_{xy}}(2,2) = 1/4h^2(u_{3,3} + u_{1,1} - u_{1,3} - u_{3,1}) = 2.08346.$$

If we look at one of the outer ring values $u_{0_{21}}$ its partial derivatives have finite difference equations

$$u_{0_{xx}}(2,1) = 1/h^2(u_{3,1} - 2u_{2,1} + u_{1,1})$$

$$u_{0_{yy}}(2,1) = 1/h^2(u_{2,1} - 2u_{2,1} + u_{2,0})$$

$$u_{0_{xy}}(2,1) = 1/4h^2(u_{3,2} + u_{1,0} - u_{1,2} - u_{3,0}).$$

Because $u_{0_{10}}, u_{0_{20}}$ and $u_{0_{30}}$ lie on the boundary, they take the boundary values:

$$u_{0_{10}} = u(x_1, y_0) = u(1/4, 0) = 1/256$$

$$u_{0_{20}} = u(x_2, y_0) = u(1/2, 0) = 1/16$$

$$u_{0_{30}} = u(x_2, y_0) = u(3/4, 0) = 81/256.$$

So the finite difference equations become

$$u_{0_{xx}}(2,1) = 1/h^2(u_{3,1} - 2u_{2,1} + u_{1,1}) = 2.80006$$

$$u_{0_{yy}}(2,1) = 1/h^2(u_{2,1} - 2u_{2,1} + 1/16) = 1.67208$$

$$u_{0_{xy}}(2,1) = 1/4h^2(u_{3,2} + 1/256 - u_{1,2} - 81/256) = 1.56049.$$

We continue this pattern and get the following matrices for $u_{0_{xx}}, u_{0_{yy}}$ and $u_{0_{xy}}$

$$u_{0_{xx}} = \begin{bmatrix} 1.41421 & 1.67208 & 2.28325 \\ 2.80006 & 2.82843 & 4.02292 \\ 4.04130 & 3.18818 & 4.24264 \end{bmatrix},$$

$$u_{0_{yy}} = \begin{bmatrix} 1.41421 & 2.80006 & 4.04130 \\ 1.67208 & 2.82843 & 3.18818 \\ 2.28325 & 4.02292 & 4.24264 \end{bmatrix},$$

and

$$u_{0_{xy}} = \begin{bmatrix} 1.43536 & 1.56049 & 0.56464 \\ 1.56049 & 2.08346 & 2.43951 \\ 0.56464 & 2.43951 & 5.43536 \end{bmatrix}.$$

Since $\det D^2 u_0 = u_{0_{xx}} u_{0_{yy}} - u_{0_{xy}}^2$ we get

$$\det D^2 u_0 = \begin{bmatrix} -0.0603 & 2.2468 & 8.9085 \\ 2.2468 & 3.6592 & 6.8746 \\ 8.9085 & 6.8746 & -11.5431 \end{bmatrix}.$$

By our iterative method we have

$$\Delta u_1 = \Delta u_0 - 1/\nu(\det D^2 u_0 - f(x_i, y_j)).$$

We let $\nu = 50$ and successfully simplify the right hand side of the equation getting a 3×3 matrix we call $F1$. We get the following matrix

$$F1 = \begin{bmatrix} 2.84463 & 4.52095 & 6.52139 \\ 4.52095 & 5.82367 & 7.70736 \\ 6.52139 & 7.70736 & 9.93114 \end{bmatrix}.$$

Note that we can use the Octave program RightHand2d.m to compute the second partial derivatives of u_0 at the mesh points and to compute $F1$.

Now we have the Poisson equation

$$\Delta u_1 = F1.$$

We solve this by using an alternate form of the finite difference solver called poisson2dv2.m. In this alternate form, the right hand side of the Poison equation is an $m \times n$ matrix.

We get the following nine values for u_1

$$u_1 = \begin{bmatrix} 0.06302 & 0.21103 & 0.53701 \\ 0.21103 & 0.46416 & 0.89928 \\ 0.53701 & 0.89928 & 1.51517 \end{bmatrix}.$$

It compares well with the exact solution at the same points:

$$\begin{bmatrix} 0.01562 & 0.09766 & 0.39062 \\ 0.09766 & 0.25000 & 0.66016 \\ 0.39062 & 0.66016 & 1.26562 \end{bmatrix}.$$

Notice that u_1 is overall a better approximation than u_0. We create a loop that repeats the above steps until the data converges and/or the error increases. For this example, we loop through the steps a total of 116 times and get the following estimate for u:

$$u_{116} = \begin{bmatrix} 0.02363 & 0.11074 & 0.40198 \\ 0.11074 & 0.26769 & 0.67395 \\ 0.40198 & 0.67395 & 1.27531 \end{bmatrix}.$$

Here we get a maximum error over the grid points of 0.017693. We want to decrease the error on u so in the following section we compute the numerical solutions to the Monge-Ampère equation over smaller mesh sizes.

4.1.2 Numerical Results

We let i be a number where $i \geq 2$ and $M = N = 2^i$. We get the following error table for this finite difference method for the two-dimensional Monge-Ampère equation.

i	$M = N$	$h = k$	it count	error	convergence rate
2	4	1/4	116	0.017693	
3	8	1/8	123	0.0046767	1.919615899
4	16	1/16	1001	0.0011644	2.005904141
5	32	1/32	1001	0.00029213	1.994904318

CHAPTER 5

NONLINEAR EQUATIONS BY NEWTON'S METHOD

Newton's method is a way to find the solution of systems of nonlinear equations. We will use Newton's method to solve the nonlinear two-dimensional Monge-Ampère eigenvalue problem. The two-dimensional Monge-Ampère eigenvalue problem is given by

$$\det D^2 u = \lambda u \text{ in } \Omega$$

$$u = 0 \text{ on } \partial\Omega.$$

5.1 Multivariate Newton's Method

Recall the one-variable Newton's Method

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}.$$

This method builds a recursive sequence that converges to the root. This provides the main outline of the Multivariate Newton's Method (Sauer, 2006). If we let

$$f_1(u, v, w) = 0$$

$$f_2(u, v, w) = 0$$

$$f_3(u, v, w) = 0,$$

be three nonlinear equations in three unknowns u,v and w, the vector valued function $F(u,v,w) = (f_1, f_2, f_3)$ and $F(x) = 0$ where $x = (u,v,w)$. The derivative of f is replaced by the Jacobian Matrix defined by

$$DF(x) = \begin{bmatrix} \frac{\partial f_1}{\partial u} & \frac{\partial f_1}{\partial v} & \frac{\partial f_1}{\partial w} \\ \frac{\partial f_2}{\partial u} & \frac{\partial f_2}{\partial v} & \frac{\partial f_2}{\partial w} \\ \frac{\partial f_3}{\partial u} & \frac{\partial f_3}{\partial v} & \frac{\partial f_3}{\partial w} \end{bmatrix}.$$

The Taylor expansion for vector-valued functions around x_0 is

$$F(x) = F(x_0) + DF(x_0)(x - x_0) + O(x - x_0)^2.$$

Since Newton's method is based on a linear approximation we ignore the $O(h^2)$ terms. We let $x = r$ be the root and x_0 be the current guess. Then

$$0 = F(r) \approx F(x_0) + DF(x_0)(r - x_0)$$

or

$$r \approx x_0 - (DF(x_0))^{-1}F(x_0).$$

Then the multivariate Newton's method is

$$x_0 = \text{initial vector}$$

$$x_{k+1} = x_k - (DF(x_k))^{-1}F(x_k).$$

5.1.1 Example

Use Newton's method with starting guess (1,2) to find a solution of the system

$$v - u^3 = 0$$

$$u^2 + v^2 - 1 = 0.$$

Since $f_1(u,v) = v - u^3$ and $f_2(u,v) = u^2 + v^2 - 1$, the Jacobian matrix is

$$DF(u,v) = \begin{bmatrix} -3u^2 & 1 \\ 2u & 2v \end{bmatrix}.$$

Using the starting point $x_0 = (1,2)$ we have

$$x_1 = \begin{bmatrix} 1 \\ 2 \end{bmatrix} - \begin{bmatrix} -3 & 1 \\ 2 & 4 \end{bmatrix}^{-1} \begin{bmatrix} 1 \\ 4 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} + \frac{1}{14} \begin{bmatrix} 4 & -1 \\ -2 & -3 \end{bmatrix} \begin{bmatrix} 1 \\ 4 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

We continue this pattern and get

$$x_2 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} - \begin{bmatrix} -3 & 1 \\ 2 & 2 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{7}{8} \\ \frac{5}{8} \end{bmatrix}.$$

The exact solution to the system is $(0.8260, 0.5636)$ so we see that x_2 is a better approximation than x_1. We will look at how to apply Newton's method to solve a Monge-Ampère eigenvalue problem in the following section.

5.2 Newton's Method for the Two-Dimensional Monge-Ampère Eigenvalue Problem

We want to create a finite difference method using Newton's multivariate method to solve the two dimensional eigenvalue problem for the Monge-Ampère eigenvalue problem

$$\det D^2 u = \lambda u \text{ in } \Omega$$

$$u = 0 \text{ on } \partial\Omega.$$

Therefore we want to solve

$$\det D^2 u - \lambda u = 0$$

$$u_{xx}u_{yy} - (u_{xy})^2 - \lambda u = 0.$$

To solve this method over an $M \times N$ grid we must explicitly find the determinant at each grid point. So we have $M-1$ equations and M unknowns. To use Newton's method we must have an equal number of equations and unknowns so we add the equation $v_1 + v_2 + ... + v_m = 1$ in order to solve the problem using the methods described in Section 5.1. Here v_1, v_2,..., v_m are the the approximate solutions on the $M \times N$ grid.

5.2.1 Example

Apply the Newton's method with $M = N = 3$ to estimate the solution to the Monge-Ampère eigenvalue problem $\det D^2 u = \lambda u$. Since $M = N = 3$, the mesh size is $h = k = 1/3$. For this type of problem, $u = 0$ on the boundary. We will use Newton's multivariate method to estimate λ and u at the four mesh points of the square.

First we use the following finite difference equations to compute the second order partial derivatives at the grid points for any u_{ij}.

$$(u_{xx})_{ij} = 1/h^2(u_{i+1,j} - 2u_{i,j} + u_{i-1,j})$$
$$(u_{yy})_{ij} = 1/h^2(u_{i,j+1} - 2u_{i,j} + u_{i,j-1})$$
$$(u_{xy})_{ij} = 1/4h^2(u_{i+1,j+1} + u_{i-1,j-1} - u_{i-1,j+1} - u_{i+1,j-1}).$$

For $u_{11} = v_1$ we have

$$(u_{xx})_{ij} = 1/h^2(u_{2,1} - 2u_{1,1} + u_{0,1}) = 1/h^2(-2v_1 + v_2)$$
$$(u_{yy})_{ij} = 1/h^2(u_{1,2} - 2u_{1,1} + u_{1,0}) = 1/h^2(-2v_1 + v_3)$$
$$(u_{xy})_{ij} = 1/4h^2(u_{2,2} + u_{0,0} - u_{0,2} - u_{2,0}) = 1/4h^2(v_4).$$

Then

$$\det D^2 v_1 - \lambda v_1 = 1/h^4(4v_1^2 - 2v_1v_3 - 2v_1v_2 + v_2v_3 - 1/16v_4^2) - \lambda v_1 = 0.$$

We follow similar steps to compute the determinant at the other three grid points and end up with the following five equations.

$$\begin{aligned}
f_1 &= 1/h^4(4v_1^2 - 2v_1v_3 - 2v_1v_2 + v_2v_3 - 1/16v_4^2) - \lambda v_1 = 0\\
f_2 &= 1/h^4(4v_2^2 - 2v_1v_2 - 2v_2v_4 + v_1v_4 - 1/16v_3^2) - \lambda v_2 = 0\\
f_3 &= 1/h^4(4v_3^2 - 2v_1v_3 - 2v_3v_4 + v_1v_4 - 1/16v_2^2) - \lambda v_3 = 0\\
f_4 &= 1/h^4(4v_4^2 - 2v_2v_4 - 2v_3v_4 + v_2v_3 - 1/16v_1^2) - \lambda v_4 = 0\\
f_5 &= v_1 + v_2 + v_3 + v_4 - 1 = 0.
\end{aligned} \tag{5.1}$$

Since we have five equations and five unknowns, v_1, v_2, v_3, v_4 and λ, we can now use Newton's method to find an approximate solution. We end up with the following solution to the Monge-Ampère eigenvalue problem

$$v_k = \begin{bmatrix} 0.250000000000000 \\ 0.250000000000000 \\ 0.250000000000000 \\ 0.250000000000000 \end{bmatrix},$$

and

$$\lambda = 18.98437499999997.$$

It has been demonstrated that Newton's method typically fails in solving this type of problem. However Newton's does suggest an update for λ that will be helpful in the methods used in Chapter 6. We will derive λ_k in the following section.

5.2.2 Lambda Update Suggested by Newton's Method

In the previous section, we obtained (Equation 5.1) based on a 3×3 mesh. Using simple algebraic techniques we can solve for λ based on the unknowns v_1, v_2, v_3 and v_4. We get

$$\lambda = \tfrac{1}{h^4}[4v_1v_2 + 4v_1v_3 + 4v_2v_4 + 4v_3v_4 - 2v_1v_4 - 2v_2v_3 - \tfrac{63}{16}(v_1^2 + v_2^2 + v_3^2 + v_4^2)].$$

We extend this approach to find a solution for λ given any i and j on a $M \times N$ mesh and get the following:

$$\lambda_{k+1} = \tfrac{1}{h^4 \sum_{ij} u_{ij}} \textstyle\sum_{ij} \big[(u_{i+1,j} - 2u_{i,j} + u_{i-1,j})(u_{i,j+1} - 2u_{i,j} + u_{i,j-1}) - \tfrac{(u_{i+1,j+1} + u_{i-1,j-1} - u_{i-1,j+1} - u_{i+1,j-1})}{16}\big].$$

We will use this update for λ in the following chapter where we aim to find a numerical solution to the eigenvalue problem for the Monge-Ampere equation.

CHAPTER 6

THE EIGENVALUE PROBLEM FOR THE MONGE-AMPÈRE EQUATION

In this chapter, we aim to find numerical solutions to the Monge-Ampere eigenvalue problem. We will use a finite difference method to solve this type of problem.

The Monge-Ampère eigenvalue problem is given by

$$S_n(D^2u) = \lambda(-u)^n.$$

6.1 Finite Difference Method for the The Two-Dimensional Eigenvalue Problem for the Monge-Ampère Equation

We want to write a finite difference code to solve the two dimensional eigenvalue problem for the Monge-Ampère equation

$$\det D^2u = \lambda u \text{ in } \Omega$$

$$u = g \text{ on } \partial\Omega,$$

by an iterative method with initial guess

$$\Delta u_0 = 2\sqrt{f} \text{ in } \Omega$$

$$u_0 = g \text{ on } \partial\Omega,$$

for f to be specified.

We evaluate the $\det D^2 u$ using Dirichlet boundary conditions over the same mesh described in Section 4.1. However, we use the following iterative method:

$$\Delta u_{p+1} = \Delta u_p - 1/\nu(\det D^2 u_p - \lambda_p u_p), \text{ in } \Omega, \qquad u = 0 \text{ on } \partial\Omega,$$

to solve the eigenvalue problem for the Monge-Ampère equation.

In attempting to derive an appropriate iterative method, one faces the difficulty of how to update λ in the iterations. We look at three different approaches:

1. A simple average
2. An update suggested by Newton's method
3. An exact λ.

We will try theses approaches in the following three examples.

6.1.1 Example 1

We begin by attempting to solve the Monge-Ampère eigenvalue problem using the iterative method

$$\Delta u_{p+1} = \Delta u_p - 1/\nu(\det D^2 u_p - \lambda_p u_p),$$

where

$$\lambda_{p+1} = \text{mean}(\det u_p / u_p).$$

We apply the finite difference method with $M = N = 4$ to estimate the solution to the eigenvalue problem for the Monge-Ampère equation on $[0, 1]$ x $[0, 1]$ with the following Dirichlet boundary conditions:

$$u(x,0) = x^4 \text{ for } 0 \leq x \leq 1$$

$$u(x,1) = (x^2+1)^2 \text{ for } 0 \leq x \leq 1$$

$$u(0,y) = y^4 \text{ for } 0 \leq y \leq 1$$

$$u(1,y) = (1+y^2)^2 \text{ for } 0 \leq y \leq 1.$$

We will use the correct solution $u(x, y) = (x^2 + y^2)^2$ to compare with the approximation at the nine mesh points of the square. In Section 4.1 we found that the right hand side of the Monge-Ampère equation is $48(x_i^2 + y_j^2)^2 = 48u$. Thus the exact λ for this problem is 48. First we use our two dimensional Poisson solver poisson2d.m with right hand side $2\sqrt{f} = 2\sqrt{16(x^2 + y^2)}$ to find our initial guess u_0. We get

$$u_0 = \begin{bmatrix} 0.06822 & 0.22092 & 0.54862 \\ 0.22092 & 0.48384 & 0.92354 \\ 0.54862 & 0.92354 & 1.54989 \end{bmatrix}.$$

Since the left hand side of our equation hasn't changed we compute the $\det D^2 u_0 = u_{0_{xx}} u_{0_{yy}} - u_{0_{xy}}^2$ exactly as we did in Section 4.1 and get the following

$$\det D^2 u_0 = \begin{bmatrix} -0.0603 & 2.2468 & 8.9085 \\ 2.2468 & 3.6592 & 6.8746 \\ 8.9085 & 6.8746 & -11.5431 \end{bmatrix}.$$

By our iterative method we have

$$\Delta u_1 = \Delta u_0 - 1/\nu(\det D^2 u_0 - \lambda_0 u_0),$$

where

$$\lambda_1 = \text{mean}(\text{det}u_0/u_0).$$

We let $\nu = 50$ and $\lambda_0 = 48$ and successfully simplify the right hand side of the equation getting a 3×3 matrix. We call this matrix $F1$. We get the following matrix

$$F1 = \begin{bmatrix} 3.1699 & 5.0307 & 7.1945 \\ 5.0307 & 6.6741 & 8.9931 \\ 7.1945 & 8.9931 & 12.1300 \end{bmatrix}.$$

Note that we can use the Octave program RightHand2deig.m to compute the second partial derivatives of u_0 at the mesh points and to compute $F1$. Now we have the Poisson equation

$$\Delta u_1 = F1.$$

We solve this by using an alternate form of the finite difference solver called poisson2dv2.m. In this alternate form, the right hand side of the Poisson equation is an $m \times n$ matrix.

We get that $\lambda_1 = -6.6514$ and following nine values for u_1

$$\begin{bmatrix} 0.03859 & 0.17233 & 0.50159 \\ 0.17233 & 0.40106 & 0.83836 \\ 0.50159 & 0.83836 & 1.45035 \end{bmatrix}.$$

It compares well with the exact solution at the same points:

$$\begin{bmatrix} 0.01562 & 0.09766 & 0.39062 \\ 0.09766 & 0.25000 & 0.66016 \\ 0.39062 & 0.66016 & 1.26562 \end{bmatrix}.$$

Notice that u_1 is overall a better approximation than u_0. We create a loop that repeats the above steps until the data converges and/or the error increases. For this example, we loop through the steps a total of 1 time and the estimate for u remains the same. We get a maximum error over the grid points of 0.18473. We want to decrease the error on u so in the following section we compute the numerical solutions to the Monge-Ampère eigenvalue problem over smaller mesh sizes.

6.1.1.1 Numerical Results

We let i be a number where $i \geq 2$ and $M = N = 2^i$. We get the following error table for this finite difference method for the two-dimensional Monge-Ampère equation.

i	$M = N$	$h = k$	it count	λ	$error_u$
2	4	1/4	1	-6.6514	0.18473
3	8	1/8	1	-11.654	0.046801
4	16	1/16	1	-11.453	0.43146
5	32	1/32	4	23.059	0.45483

6.1.2 Example 2

We now attempt to solve the Monge-Ampère eigenvalue problem using the iterative method

$$\Delta u_{p+1} = \Delta u_p - 1/\nu(\det D^2 u_p - \lambda_p u_p),$$

where

$$\lambda = \frac{1}{h^4 \sum u_{ij}^2} \sum [(u_{i+1,j} - 2u_{i,j} + u_{i-1,j})(u_{i,j+1} - 2u_{i,j} + u_{i,j-1}) - \frac{(u_{i+1,j+1}+u_{i-1,j-1}-u_{i-1,j+1}-u_{i+1,j-1})}{16}].$$

We use the Octave program RightHand2deig.m and simply adjust the update for λ. When we run the code we get very large error on both λ and u so we conclude that this approach does not work.

6.1.3 Example 3

We now attempt to solve the Monge-Ampère eigenvalue problem using the iterative method

$$\Delta u_{p+1} = \Delta u_p - 1/\nu(\det D^2 u_p - \lambda u_p),$$

where

$$\lambda = 48.$$

We use the Octave program RightHand2deig.m and simply set λ constant at 48. The numerical results are given in the following section.

6.1.3.1 Numerical Results

We let i be a number where $i \geq 2$ and $M = N = 2^i$. We get the following error table for this finite difference method for the two-dimensional Monge-Ampère equation.

i	$M = N$	$h = k$	it count	$error_u$	convergence rate
2	4	1/4	49	0.010732	
3	8	1/8	1255	0.0028463	1.914759325
4	16	1/16	956	0.000737	1.949214185
5	32	1/32	538	0.00018512	1.99334087

CHAPTER 7

CONCLUSION

Our goal was to find a numerical solution to the following nonlinear eigenvalue problem

$$\det(D^2u) = \lambda u \text{ in } \Omega, \qquad u = 0 \text{ on } \partial\Omega,$$

using the iterative method

$$-\nu\Delta u_{p+1} = -\nu\Delta u_p + \det(D^2u_p) - \lambda_p u_p \text{ in } \Omega, \qquad u = 0 \text{ on } \partial\Omega.$$

We encountered a problem when trying to find a strategy for updating λ. We looked at three different approaches:

1. A simple average
2. An update suggested by Newton's method
3. An exact λ.

The first approach produced small error on u however the error on λ was still large and the error on u was increasing. The second approach produced large error on both λ and u so we concluded that it was not accurate. The third approach used the exact λ and produced small, deceasing error on u. Thus once one figures out the correct strategy to update λ, the work done in this thesis would immediately lead to a solution of the problem.

The methods of this paper can be applied to find numerical solutions for both the 2-Hessian equation and 2-Hessian eigenvalue problems.

APPENDICES

Appendix A

CODE FOR 2D POISSON EQUATION

Finite difference solver for a $2D$ Poisson equation with Dirichlet boundary conditions on a rectangle. The input is on a rectangle domain $[x_l, x_r] \times [y_b, y_t]$, covered by an $M \times N$ grid. The output is matrix w holding solution values on an $M \times N$ grid.

```
function w=poisson2d(xl,xr,yb,yt,M,N,f,g,caseNumb)
m=M-1;n=N-1;
h=(xr-xl)/M;h2=h^2;k=(yt-yb)/N;
r=h2/k^2;s=2*(1+r);
x=xl+(xr-xl)*(0:M)/M;
y=yb+(yt-yb)*(0:N)/N;
z=zeros(1,m-2);
a=zeros(m*n,m*n);b=zeros(m*n,1);
% inner core
for i=2:m-1
for j=2:n-1
a(i+(j-1)*m,:)=[zeros(1,i-1+(j-2)*m) r z 1 -s 1 z r zeros(1,(n-j)*m-i)];
b(i+(j-1)*m)=h2*f(x(i+1),y(j+1),caseNumb);
end
end
% outer ring
```

Appendix A (Continued)

```
j=1;
for i=2:m-1
a(i+(j-1)*m,:)=[zeros(1,i-2) 1 -s 1 z r zeros(1,(n-j)*m-i)];
b(i+(j-1)*m)=h2*f(x(i+1),y(j+1),caseNumb)-r*gbottom(x(i+1),yb,caseNumb);
end
j=n;
for i=2:m-1
a(i+(j-1)*m,:)=[zeros(1,i-1+(j-2)*m) r z 1 -s 1 zeros(1,m-i-1)];
b(i+(j-1)*m)=h2*f(x(i+1),y(j+1),caseNumb)-r*gtop(x(i+1),yt,caseNumb);
end
i=1;
for j=2:n-1
a(i+(j-1)*m,:)=[zeros(1,i-1+(j-2)*m) r z 0 -s 1 z r zeros(1,(n-j)*m-i)];
b(i+(j-1)*m)=h2*f(x(i+1),y(j+1),caseNumb)-gleft(xl,y(j+1),caseNumb);
end
i=m;
for j=2:n-1
a(i+(j-1)*m,:)=[zeros(1,(j-1)*m-1) r z 1 -s 0 z r zeros(1,(n-j)*m-i)];
b(i+(j-1)*m)=h2*f(x(i+1),y(j+1),caseNumb)-gright(xr,y(j+1),caseNumb);
end
% four corners
i=1;j=1;
a(i+(j-1)*m,:)=[-s 1 z r zeros(1,(n-1)*m-1)];
b(i+(j-1)*m)=h2*f(x(i+1),y(j+1),caseNumb)- r*gbottom(x(i+1),yb,caseNumb)- ...
    gleft(xl,y(j+1),caseNumb);
i=m;j=1;
```

```
a(i+(j-1)*m,:)=[z 1 -s 0 z r zeros(1,(n-2)*m)];
b(i+(j-1)*m)=h2*f(x(i+1),y(j+1),caseNumb)- r*gbottom(x(i+1),yb,caseNumb)-  ...
    gright(xr,y(j+1),caseNumb);
i=1;j=n;
a(i+(j-1)*m,:)=[zeros(1,(n-2)*m) r z 0 -s 1 zeros(1,m-2)];
b(i+(j-1)*m)=h2*f(x(i+1),y(j+1),caseNumb)- r*gtop(x(i+1),yt,caseNumb)-  ...
    gleft(xl,y(j+1),caseNumb);
i=m;j=n;
a(i+(j-1)*m,:)=[zeros(1,(n-1)*m-1) r z 1 -s];
b(i+(j-1)*m)=h2*f(x(i+1),y(j+1),caseNumb)- r*gtop(x(i+1),yt,caseNumb)-  ...
    gright(xr,y(j+1),caseNumb);
v=a\b; % solve for solution
w=zeros(m,n);
for i=1:m % put solution into mesh
for j=1:n
w(i,j)=v(i+(j-1)*m);
end
end
%boundary conditions
function u=gbottom(x,yb,caseNumb) % bottom of rectangle
u=g(x,yb,caseNumb);
function u=gtop(x,yt,caseNumb) % top of rectangle
u=g(x,yt,caseNumb);
function u=gleft(xl,y,caseNumb) % left side of rectangle
u=g(xl,y,caseNumb);
function u=gright(xr,y,caseNumb) % right side of rectangle
```

```
u=g(xr,y,caseNumb);
```

Appendix B

CODE FOR 3D POISSON EQUATION

Finite difference solver for a $3D$ Poisson equation with Dirichlet boundary conditions on a cube. The input is on the cubic domain $[x_1, x_2] \times [y_1, y_2] \times [z_1, z_2]$, covered by an $M \times N \times P$ grid. The output is matrix w holding solution values on an $M \times N \times P$ grid.

```
function w=poisson3d(x1,x2,y1,y2,z1,z2,M,N,P,f,g,caseNumb)
 m=M-1;n=N-1;p=P-1;
h=(x2-x1)/M;
k=(y2-y1)/N;
c=(z2-z1)/P;
h2=h^2;r=h2/k^2;q=h2/c^2;
s=2*(1+r+q);
x=x1+(x2-x1)*(0:M)/M;
y=y1+(y2-y1)*(0:N)/N;
z=z1+(z2-z1)*(0:P)/P;
a=zeros(m*n*p,m*n*p);b=zeros(m*n*p,1);
% inner core
for i=2:m-1
for j=2:n-1
for l=2:p-1
d1=i-1+(j-1)*m+(l-1)*m^2;
```

```
d2=i+1+(j-1)*m+(l-1)*m^2;
d3=i+(j-1)*m+(l-1)*m^2;
d4=i+(j-2)*m+(l-1)*m^2;
d5=i+j*m+(l-1)*m^2;
d6=i+(j-1)*m+(l-2)*m^2;
d7=i+(j-1)*m+l*m^2;
    a(i+(j-1)*m+(l-1)*m^2,:)=[zeros(1,d6-1) q zeros(1,d4-d6-1) r  ...
        zeros(1,d1-d4-1) 1 zeros(1,d3-d1-1) -s zeros(1,d2-d3-1) 1  ...
        zeros(1,d5-d2-1) r zeros(1,d7-d5-1) q zeros(1,m*n*p-d7)];
    b(i+(j-1)*m+(l-1)*m^2)=h2*f(x(i+1),y(j+1),z(l+1),caseNumb);
end
end
end
%Outer ring 1 values
i=1;
for l=2:p-1
    for j=2:n-1
    a(i+(j-1)*m+(l-1)*m^2,:)=[zeros(1,d6-1) q zeros(1,d4-d6-1) r  ...
        zeros(1,d1-d4-1) 0 zeros(1,d3-d1-1) -s zeros(1,d2-d3-1) 1  ...
        zeros(1,d5-d2-1) r zeros(1,d7-d5-1) q zeros(1,m*n*p-d7)];
    b(i+(j-1)*m+(l-1)*m^2)=h2*f(x(i+1),y(j+1),z(l+1),caseNumb)-  ...
        g5(x1,y(j+1),z(l+1),caseNumb);
end
end
i=m;
for l=2:p-1
```

```
    for j=2:n-1
    a(i+(j-1)*m+(l-1)*m^2,:)=[zeros(1,d6-1) q zeros(1,d4-d6-1) r  ...
        zeros(1,d1-d4-1) 1 zeros(1,d3-d1-1) -s zeros(1,d2-d3-1) 0  ...
        zeros(1,d5-d2-1) r zeros(1,d7-d5-1) q zeros(1,m*n*p-d7)];
    b(i+(j-1)*m+(l-1)*m^2)=h2*f(x(i+1),y(j+1),z(l+1),caseNumb)-  ...
        g6(x2,y(j+1),z(l+1),caseNumb);
end
end
j=1;
for l=2:p-1
    for i=2:m-1
    a(i+(j-1)*m+(l-1)*m^2,:)=[zeros(1,d6-1) q zeros(1,d4-d6-1) 0  ...
        zeros(1,d1-d4-1) 1 zeros(1,d3-d1-1) -s zeros(1,d2-d3-1) 1  ...
        zeros(1,d5-d2-1) r zeros(1,d7-d5-1) q zeros(1,m*n*p-d7)];
    b(i+(j-1)*m+(l-1)*m^2)=h2*f(x(i+1),y(j+1),z(l+1),caseNumb)-  ...
        r*g1(x(i+1),y1,z(l+1),caseNumb);
end
end
j=n;
for l=2:p-1
    for i=2:m-1
    a(i+(j-1)*m+(l-1)*m^2,:)=[zeros(1,d6-1) q zeros(1,d4-d6-1) r  ...
        zeros(1,d1-d4-1) 1 zeros(1,d3-d1-1) -s zeros(1,d2-d3-1) 1  ...
        zeros(1,d5-d2-1) 0 zeros(1,d7-d5-1) q zeros(1,m*n*p-d7)];
    b(i+(j-1)*m+(l-1)*m^2)=h2*f(x(i+1),y(j+1),z(l+1),caseNumb)-  ...
        r*g2(x(i+1),y2,z(l+1),caseNumb);
```

```
end
end
l=1;
for i=2:m-1
    for j=2:n-1
    a(i+(j-1)*m+(l-1)*m^2,:)=[zeros(1,d4-1) r zeros(1,d1-d4-1) 1  ...
        zeros(1,d3-d1-1) -s zeros(1,d2-d3-1) 1 zeros(1,d5-d2-1) r  ...
        zeros(1,d7-d5-1) q zeros(1,m*n*p-d7)];
    b(i+(j-1)*m+(l-1)*m^2)=h2*f(x(i+1),y(j+1),z(l+1),caseNumb)-  ...
        q*g3(x(i+1),y(j+1),z1,caseNumb);
end
end
l=p;
for i=2:m-1
    for j=2:n-1
    a(i+(j-1)*m+(l-1)*m^2,:)=[zeros(1,d6-1) q zeros(1,d4-d6-1) r  ...
        zeros(1,d1-d4-1) 1 zeros(1,d3-d1-1) -s zeros(1,d2-d3-1) 1  ...
        zeros(1,d5-d2-1) r zeros(1,m*n*p-d5)];
    b(i+(j-1)*m+(l-1)*m^2)=h2*f(x(i+1),y(j+1),z(l+1),caseNumb)-  ...
        q*g4(x(i+1),y(j+1),z2,caseNumb);
end
end
%Outer ring 2 values
i=1;j=1;
for l=2:p-1
```

Appendix B (Continued)

```
        a(i+(j-1)*m+(l-1)*m^2,:)=[zeros(1,d6-1) q zeros(1,d4-d6-1) 0  ...
            zeros(1,d1-d4-1) 0 zeros(1,d3-d1-1) -s zeros(1,d2-d3-1) 1  ...
            zeros(1,d5-d2-1) r zeros(1,d7-d5-1) q zeros(1,m*n*p-d7)];
    b(i+(j-1)*m+(l-1)*m^2)=h2*f(x(i+1),y(j+1),z(l+1),caseNumb)-  ...
        g5(x1,y(j+1),z(l+1),caseNumb)- r*g1(x(i+1),y1,z(l+1),caseNumb);
end
i=m;j=n;
for l=2:p-1
        a(i+(j-1)*m+(l-1)*m^2,:)=[zeros(1,d6-1) q zeros(1,d4-d6-1) r  ...
            zeros(1,d1-d4-1) 1 zeros(1,d3-d1-1) -s zeros(1,d2-d3-1) 0  ...
            zeros(1,d5-d2-1) 0 zeros(1,d7-d5-1) q zeros(1,m*n*p-d7)];
    b(i+(j-1)*m+(l-1)*m^2)=h2*f(x(i+1),y(j+1),z(l+1),caseNumb)-  ...
        g6(x2,y(j+1),z(l+1),caseNumb)- r*g2(x(i+1),y2,z(l+1),caseNumb);
end
j=1;l=1;
for i=2:m-1
        a(i+(j-1)*m+(l-1)*m^2,:)=[zeros(1,d1-1) 1 zeros(1,d3-d1-1) -s  ...
            zeros(1,d2-d3-1) 1 zeros(1,d5-d2-1) r zeros(1,d7-d5-1) q  ...
            zeros(1,m*n*p-d7)];
    b(i+(j-1)*m+(l-1)*m^2)=h2*f(x(i+1),y(j+1),z(l+1),caseNumb)-  ...
        q*g3(x(i+1),y(j+1),z1,caseNumb)- r*g1(x(i+1),y1,z(l+1),caseNumb);
end
j=n;l=p;
for i=2:m-1
```

```
        a(i+(j-1)*m+(l-1)*m^2,:)=[zeros(1,d6-1) q zeros(1,d4-d6-1) r  ...
            zeros(1,d1-d4-1) 1 zeros(1,d3-d1-1) -s zeros(1,d2-d3-1) 1  ...
            zeros(1,m*n*p-d2)];
    b(i+(j-1)*m+(l-1)*m^2)=h2*f(x(i+1),y(j+1),z(l+1),caseNumb)-  ...
        q*g4(x(i+1),y(j+1),z2,caseNumb)- r*g2(x(i+1),y2,z(l+1),caseNumb);
end
i=1;l=1;
for j=2:n-1
        a(i+(j-1)*m+(l-1)*m^2,:)=[zeros(1,d4-1) r zeros(1,d1-d4-1) 0  ...
            zeros(1,d3-d1-1) -s zeros(1,d2-d3-1) 1 zeros(1,d5-d2-1) r  ...
            zeros(1,d7-d5-1) q zeros(1,m*n*p-d7)];
    b(i+(j-1)*m+(l-1)*m^2)=h2*f(x(i+1),y(j+1),z(l+1),caseNumb)-  ...
        q*g3(x(i+1),y(j+1),z1,caseNumb)- g5(x1,y(j+1),z(l+1),caseNumb);
end
i=m;l=p;
for j=2:n-1
        a(i+(j-1)*m+(l-1)*m^2,:)=[zeros(1,d6-1) q zeros(1,d4-d6-1) r  ...
            zeros(1,d1-d4-1) 1 zeros(1,d3-d1-1) -s zeros(1,d2-d3-1) 0  ...
            zeros(1,d5-d2-1) r zeros(1,m*n*p-d5)];
    b(i+(j-1)*m+(l-1)*m^2)=h2*f(x(i+1),y(j+1),z(l+1),caseNumb)-  ...
        q*g4(x(i+1),y(j+1),z2,caseNumb)- g6(x2,y(j+1),z(l+1),caseNumb);
end
i=m;l=1;
for j=2:n-1
```

```
        a(i+(j−1)*m+(l−1)*m^2,:)=[zeros(1,d4−1) r zeros(1,d1−d4−1) 1  ...
            zeros(1,d3−d1−1) −s zeros(1,d2−d3−1) 0 zeros(1,d5−d2−1) r  ...
            zeros(1,d7−d5−1) q zeros(1,m*n*p−d7)];
    b(i+(j−1)*m+(l−1)*m^2)=h2*f(x(i+1),y(j+1),z(l+1),caseNumb)−  ...
        q*g3(x(i+1),y(j+1),z1,caseNumb)− g6(x2,y(j+1),z(l+1),caseNumb);
end
i=1;l=p;
for j=2:n−1
        a(i+(j−1)*m+(l−1)*m^2,:)=[zeros(1,d6−1) q zeros(1,d4−d6−1) r  ...
            zeros(1,d1−d4−1) 0 zeros(1,d3−d1−1) −s zeros(1,d2−d3−1) 1  ...
            zeros(1,d5−d2−1) r zeros(1,m*n*p−d5)];
    b(i+(j−1)*m+(l−1)*m^2)=h2*f(x(i+1),y(j+1),z(l+1),caseNumb)−  ...
        q*g4(x(i+1),y(j+1),z2,caseNumb)− g5(x1,y(j+1),z(l+1),caseNumb);
end
j=1;l=p;
for i=2:m−1
        a(i+(j−1)*m+(l−1)*m^2,:)=[zeros(1,d6−1) q zeros(1,d4−d6−1) 0  ...
            zeros(1,d1−d4−1) 1 zeros(1,d3−d1−1) −s zeros(1,d2−d3−1) 1  ...
            zeros(1,d5−d2−1) r zeros(1,m*n*p−d5)];
    b(i+(j−1)*m+(l−1)*m^2)=h2*f(x(i+1),y(j+1),z(l+1),caseNumb)−  ...
        q*g4(x(i+1),y(j+1),z2,caseNumb)− r*g1(x(i+1),y1,z(l+1),caseNumb);
end
j=n;l=1;
for i=2:m−1
```

```
        a(i+(j−1)*m+(l−1)*m^2,:)=[zeros(1,d4−1) r zeros(1,d1−d4−1) 1  ...
            zeros(1,d3−d1−1) −s zeros(1,d2−d3−1) 1 zeros(1,d5−d2−1) 0  ...
            zeros(1,d7−d5−1) q zeros(1,m*n*p−d7)];
    b(i+(j−1)*m+(l−1)*m^2)=h2*f(x(i+1),y(j+1),z(l+1),caseNumb)−  ...
        q*g3(x(i+1),y(j+1),z1,caseNumb)− r*g2(x(i+1),y2,z(l+1),caseNumb);
end
i=1;j=n;
for l=2:p−1
        a(i+(j−1)*m+(l−1)*m^2,:)=[zeros(1,d6−1) q zeros(1,d4−d6−1) r  ...
            zeros(1,d1−d4−1) 0 zeros(1,d3−d1−1) −s zeros(1,d2−d3−1) 1  ...
            zeros(1,d5−d2−1) 0 zeros(1,d7−d5−1) q zeros(1,m*n*p−d7)];
    b(i+(j−1)*m+(l−1)*m^2)=h2*f(x(i+1),y(j+1),z(l+1),caseNumb)−  ...
        r*g2(x(i+1),y2,z(l+1),caseNumb)− g5(x1,y(j+1),z(l+1),caseNumb);
end
i=m;j=1;
for l=2:p−1
        a(i+(j−1)*m+(l−1)*m^2,:)=[zeros(1,d6−1) q zeros(1,d4−d6−1) 0  ...
            zeros(1,d1−d4−1) 1 zeros(1,d3−d1−1) −s zeros(1,d2−d3−1) 0  ...
            zeros(1,d5−d2−1) r zeros(1,d7−d5−1) q zeros(1,m*n*p−d7)];
    b(i+(j−1)*m+(l−1)*m^2)=h2*f(x(i+1),y(j+1),z(l+1),caseNumb)−  ...
        r*g1(x(i+1),y1,z(l+1),caseNumb)− g6(x2,y(j+1),z(l+1),caseNumb);
end
% eight corners
i=1;j=1;l=1;
a(i+(j−1)*m+(l−1)*m^2,:)=[−s zeros(1,d2−d3−1) 1 zeros(1,d5−d2−1) r  ...
    zeros(1,d7−d5−1) q zeros(1,m*n*p−d7)];
```

```
b(i+(j-1)*m+(l-1)*m^2)=h2*f(x(i+1),y(j+1),z(l+1),caseNumb)-  ...
    r*g1(x(i+1),y1,z(l+1),caseNumb)- q*g3(x(i+1),y(j+1),z1,caseNumb)-  ...
    g5(x1,y(j+1),z(l+1),caseNumb);
i=m;j=1;l=1;
a(i+(j-1)*m+(l-1)*m^2,:)=[zeros(1,d1-1) 1 -s zeros(1,d2-d3-1) 0  ...
    zeros(1,d5-d2-1) r zeros(1,d7-d5-1) q zeros(1,m*n*p-d7)];
b(i+(j-1)*m+(l-1)*m^2)=h2*f(x(i+1),y(j+1),z(l+1),caseNumb)-  ...
    r*g1(x(i+1),y1,z(l+1),caseNumb)- q*g3(x(i+1),y(j+1),z1,caseNumb)-  ...
    g6(x2,y(j+1),z(l+1),caseNumb);
i=1;j=n;l=1;
a(i+(j-1)*m+(l-1)*m^2,:)=[zeros(1,d4-1) r zeros(1,d1-d4-1) 0  ...
    zeros(1,d3-d1-1) -s zeros(1,d2-d3-1) 1 zeros(1,d5-d2-1) 0  ...
    zeros(1,d7-d5-1) q zeros(1,m*n*p-d7)];
b(i+(j-1)*m+(l-1)*m^2)=h2*f(x(i+1),y(j+1),z(l+1),caseNumb)-  ...
    q*g3(x(i+1),y(j+1),z1,caseNumb)- r*g2(x(i+1),y2,z(l+1),caseNumb)-  ...
    g5(x1,y(j+1),z(l+1),caseNumb);
i=1;j=n;l=p;
a(i+(j-1)*m+(l-1)*m^2,:)=[zeros(1,d6-1) q zeros(1,d4-d6-1) r  ...
    zeros(1,d1-d4-1) 0 zeros(1,d3-d1-1) -s zeros(1,d2-d3-1) 1  ...
    zeros(1,m*n*p-d2)];
b(i+(j-1)*m+(l-1)*m^2)=h2*f(x(i+1),y(j+1),z(l+1),caseNumb)-  ...
    r*g2(x(i+1),y2,z(l+1),caseNumb)- q*g4(x(i+1),y(j+1),z2,caseNumb)-  ...
    g5(x1,y(j+1),z(l+1),caseNumb);
i=1;j=1;l=p;
```

```
a(i+(j-1)*m+(l-1)*m^2,:)=[zeros(1,d6-1) q zeros(1,d4-d6-1) 0  ...
    zeros(1,d1-d4-1) 0 zeros(1,d3-d1-1) -s zeros(1,d2-d3-1) 1  ...
    zeros(1,d5-d2-1) r zeros(1,m*n*p-d5)];
b(i+(j-1)*m+(l-1)*m^2)=h2*f(x(i+1),y(j+1),z(l+1),caseNumb)-  ...
    q*g4(x(i+1),y(j+1),z2,caseNumb)- g5(x1,y(j+1),z(l+1),caseNumb)-  ...
    r*g1(x(i+1),y1,z(l+1),caseNumb);
i=m;j=1;l=p;
a(i+(j-1)*m+(l-1)*m^2,:)=[zeros(1,d6-1) q zeros(1,d4-d6-1) 0  ...
    zeros(1,d1-d4-1) 1 zeros(1,d3-d1-1) -s zeros(1,d2-d3-1) 0  ...
    zeros(1,d5-d2-1) r zeros(1,m*n*p-d5)];
b(i+(j-1)*m+(l-1)*m^2)=h2*f(x(i+1),y(j+1),z(l+1),caseNumb)-  ...
    q*g4(x(i+1),y(j+1),z2,caseNumb)- g6(x2,y(j+1),z(l+1),caseNumb)-  ...
    r*g1(x(i+1),y1,z(l+1),caseNumb);
i=m;j=n;l=1;
a(i+(j-1)*m+(l-1)*m^2,:)=[zeros(1,d4-1) r zeros(1,d1-d4-1) 1  ...
    zeros(1,d3-d1-1) -s zeros(1,d2-d3-1) 0 zeros(1,d5-d2-1) 0  ...
    zeros(1,d7-d5-1) q zeros(1,m*n*p-d7)];
b(i+(j-1)*m+(l-1)*m^2)=h2*f(x(i+1),y(j+1),z(l+1),caseNumb)-  ...
    q*g3(x(i+1),y(j+1),z1,caseNumb)- g6(x2,y(j+1),z(l+1),caseNumb)-  ...
    r*g2(x(i+1),y2,z(l+1),caseNumb);
i=m;j=n;l=p;
a(i+(j-1)*m+(l-1)*m^2,:)=[zeros(1,d6-1) q zeros(1,d4-d6-1) r  ...
    zeros(1,d1-d4-1) 1 zeros(1,d3-d1-1) -s];
b(i+(j-1)*m+(l-1)*m^2)=h2*f(x(i+1),y(j+1),z(l+1),caseNumb)-  ...
    g6(x2,y(j+1),z(l+1),caseNumb)- r*g2(x(i+1),y2,z(l+1),caseNumb)-  ...
    q*g4(x(i+1),y(j+1),z2,caseNumb);
```

```
v=a\b; % solve for solution
w=zeros(m,n,p); %numerical solution at grid points
for i=1:m
    for j=1:n
        for l=1:p
    w(i,j,l)=v(i+(j-1)*m+(l-1)*m^2);
end
end
end
%Boundary Conditions
        function u=g1(x,y1,z,caseNumb)
            u=g(x,y1,z,caseNumb);
        function u=g2(x,y2,z,caseNumb)
            u=g(x,y2,z,caseNumb);
        function u=g3(x,y,z1,caseNumb)
            u=g(x,y,z1,caseNumb);
        function u=g4(x,y,z2,caseNumb)
            u=g(x,y,z2,caseNumb);
        function u=g5(x1,y,z,caseNumb)
            u=g(x1,y,z,caseNumb);
        function u=g6(x2,y,z,caseNumb)
            u=g(x2,y,z,caseNumb);
```

Appendix C

CODE FOR 2D LAPLACIAN EIGENVALUE PROBLEM

Finite difference solver for a $2D$ Laplacian eigenvalue problem with Dirichlet boundary conditions on a rectangle. The input is on a rectangle domain $[x_l, x_r] \times [y_b, y_t]$, covered by an $M \times N$ grid. The output is matrix w holding solution values on an $M \times N$ grid.

```
function [w,lamb]=laplacian2deig(xl,xr,yb,yt,M,N,g,caseNumb)
m=M-1;n=N-1;
h=(xr-xl)/M;h2=h^2;k=(yt-yb)/N;
r=h2/k^2;s=2*(1+r);
x=xl+(xr-xl)*(0:M)/M;
y=yb+(yt-yb)*(0:N)/N;
z=zeros(1,m-2);
a=zeros(m*n,m*n);
% inner core
for i=2:m-1
for j=2:n-1
a(i+(j-1)*m,:)=(1/h2)*[zeros(1,i-1+(j-2)*m) r z 1 -s 1 z r zeros(1,(n-j)*m-i)];
end
end
% outer ring
j=1;
```

```
for i=2:m-1
a(i+(j-1)*m,:)=(1/h2)*[zeros(1,i-2) 1 -s 1 z r zeros(1,(n-j)*m-i)];
end
j=n;
for i=2:m-1
a(i+(j-1)*m,:)=(1/h2)*[zeros(1,i-1+(j-2)*m) r z 1 -s 1 zeros(1,m-i-1)];
end
i=1;
for j=2:n-1
a(i+(j-1)*m,:)=(1/h2)*[zeros(1,i-1+(j-2)*m) r z 0 -s 1 z r zeros(1,(n-j)*m-i)];
end
i=m;
for j=2:n-1
a(i+(j-1)*m,:)=(1/h2)*[zeros(1,(j-1)*m-1) r z 1 -s 0 z r zeros(1,(n-j)*m-i)];
end
% four corners
i=1;j=1;
a(i+(j-1)*m,:)=(1/h2)*[-s 1 z r zeros(1,(n-1)*m-1)];
i=m;j=1;
a(i+(j-1)*m,:)=(1/h2)*[z 1 -s 0 z r zeros(1,(n-2)*m)];
i=1;j=n;
a(i+(j-1)*m,:)=(1/h2)*[zeros(1,(n-2)*m) r z 0 -s 1 zeros(1,m-2)];
i=m;j=n;
a(i+(j-1)*m,:)=(1/h2)*[zeros(1,(n-1)*m-1) r z 1 -s];
lambda=eig(a);
[V,D]=eig(a); % solve for solution
```

```
lamb=lambda(n*m,1);
v=V(:,n*m);
w=zeros(m,n);
for i=1:m % put solution into mesh
for j=1:n
w(i,j)=v(i+(j-1)*m);
end
end
%boundary conditions
function u=gbottom(x,yb,caseNumb) % bottom of rectangle
u=g(x,yb,caseNumb);
function u=gtop(x,yt,caseNumb) % top of rectangle
u=g(x,yt,caseNumb);
function u=gleft(xl,y,caseNumb) % left side of rectangle
u=g(xl,y,caseNumb);
function u=gright(xr,y,caseNumb) % right side of rectangle
u=g(xr,y,caseNumb);
```

Appendix D

CODE FOR 3D LAPLACIAN EIGENVALUE PROBLEM

Finite difference solver for a $3D$ Laplacian eigenvalue problem with Dirichlet boundary conditions on a cube. The input is on the cubic domain $[x_1, x_2] \times [y_1, y_2] \times [z_1, z_2]$, covered by an $M \times N \times P$ grid. The output is matrix w holding solution values on an $M \times N \times P$ grid.

```
function [w,lamb]=laplacian3deig(x1,x2,y1,y2,z1,z2,M,N,P,g,caseNumb)
m=M-1;n=N-1;p=P-1;
h=(x2-x1)/M;
k=(y2-y1)/N;
c=(z2-z1)/P;
h2=h^2;r=h2/k^2;q=h2/c^2;
s=2*(1+r+q);
x=x1+(x2-x1)*(0:M)/M;
y=y1+(y2-y1)*(0:N)/N;
z=z1+(z2-z1)*(0:P)/P;
a=zeros(m*n*p,m*n*p);
% Inner core
for i=2:m-1
for j=2:n-1
for l=2:p-1
d1=i-1+(j-1)*m+(l-1)*m^2;
```

```
d2=i+1+(j-1)*m+(l-1)*m^2;
d3=i+(j-1)*m+(l-1)*m^2;
d4=i+(j-2)*m+(l-1)*m^2;
d5=i+j*m+(l-1)*m^2;
d6=i+(j-1)*m+(l-2)*m^2;
d7=i+(j-1)*m+l*m^2;
    a(i+(j-1)*m+(l-1)*m^2,:)=[zeros(1,d6-1) q zeros(1,d4-d6-1) r  ...
        zeros(1,d1-d4-1) 1 zeros(1,d3-d1-1) -s zeros(1,d2-d3-1) 1  ...
        zeros(1,d5-d2-1) r zeros(1,d7-d5-1) q zeros(1,m*n*p-d7)];
end
end
end
%Outer ring 1 values
i=1;
for l=2:p-1
    for j=2:n-1
    a(i+(j-1)*m+(l-1)*m^2,:)=[zeros(1,d6-1) q zeros(1,d4-d6-1) r  ...
        zeros(1,d1-d4-1) 0 zeros(1,d3-d1-1) -s zeros(1,d2-d3-1) 1  ...
        zeros(1,d5-d2-1) r zeros(1,d7-d5-1) q zeros(1,m*n*p-d7)];
end
end
i=m;
for l=2:p-1
    for j=2:n-1
```

Appendix D (Continued)

```
    a(i+(j−1)*m+(l−1)*m^2,:)=[zeros(1,d6−1) q zeros(1,d4−d6−1) r  ...
        zeros(1,d1−d4−1) 1 zeros(1,d3−d1−1) −s zeros(1,d2−d3−1) 0  ...
        zeros(1,d5−d2−1) r zeros(1,d7−d5−1) q zeros(1,m*n*p−d7)];
end
end
j=1;
for l=2:p−1
    for i=2:m−1
    a(i+(j−1)*m+(l−1)*m^2,:)=[zeros(1,d6−1) q zeros(1,d4−d6−1) 0  ...
        zeros(1,d1−d4−1) 1 zeros(1,d3−d1−1) −s zeros(1,d2−d3−1) 1  ...
        zeros(1,d5−d2−1) r zeros(1,d7−d5−1) q zeros(1,m*n*p−d7)];
end
end
j=n;
for l=2:p−1
    for i=2:m−1
    a(i+(j−1)*m+(l−1)*m^2,:)=[zeros(1,d6−1) q zeros(1,d4−d6−1) r  ...
        zeros(1,d1−d4−1) 1 zeros(1,d3−d1−1) −s zeros(1,d2−d3−1) 1  ...
        zeros(1,d5−d2−1) 0 zeros(1,d7−d5−1) q zeros(1,m*n*p−d7)];
end
end
l=1;
for i=2:m−1
    for j=2:n−1
    a(i+(j-1)*m+(l-1)*m^2,:)=[zeros(1,d4-1) r zeros(1,d1-d4-1) 1  ...
        zeros(1,d3-d1-1) -s zeros(1,d2-d3-1) 1 zeros(1,d5-d2-1) r  ...
        zeros(1,d7-d5-1) q zeros(1,m*n*p-d7)];
end
end
l=p;
for i=2:m-1
    for j=2:n-1
    a(i+(j-1)*m+(l-1)*m^2,:)=[zeros(1,d6-1) q zeros(1,d4-d6-1) r  ...
        zeros(1,d1-d4-1) 1 zeros(1,d3-d1-1) -s zeros(1,d2-d3-1) 1  ...
        zeros(1,d5-d2-1) r zeros(1,m*n*p-d5)];
end
end
%Outer ring 2 values
i=1;j=1;
for l=2:p-1
        a(i+(j-1)*m+(l-1)*m^2,:)=[zeros(1,d6-1) q zeros(1,d4-d6-1) 0  ...
            zeros(1,d1-d4-1) 0 zeros(1,d3-d1-1) -s zeros(1,d2-d3-1) 1  ...
            zeros(1,d5-d2-1) r zeros(1,d7-d5-1) q zeros(1,m*n*p-d7)];
end
i=m;j=n;
for l=2:p-1
        a(i+(j-1)*m+(l-1)*m^2,:)=[zeros(1,d6-1) q zeros(1,d4-d6-1) r  ...
            zeros(1,d1-d4-1) 1 zeros(1,d3-d1-1) -s zeros(1,d2-d3-1) 0  ...
            zeros(1,d5-d2-1) 0 zeros(1,d7-d5-1) q zeros(1,m*n*p-d7)];
end
```

Appendix D (Continued)

```
j=1;l=1;
for i=2:m-1
        a(i+(j-1)*m+(l-1)*m^2,:)=[zeros(1,d1-1) 1 zeros(1,d3-d1-1) -s  ...
            zeros(1,d2-d3-1) 1 zeros(1,d5-d2-1) r zeros(1,d7-d5-1) q  ...
            zeros(1,m*n*p-d7)];
end
j=n;l=p;
for i=2:m-1
        a(i+(j-1)*m+(l-1)*m^2,:)=[zeros(1,d6-1) q zeros(1,d4-d6-1) r  ...
            zeros(1,d1-d4-1) 1 zeros(1,d3-d1-1) -s zeros(1,d2-d3-1) 1  ...
            zeros(1,m*n*p-d2)];
end
i=1;l=1;
for j=2:n-1
        a(i+(j-1)*m+(l-1)*m^2,:)=[zeros(1,d4-1) r zeros(1,d1-d4-1) 0  ...
            zeros(1,d3-d1-1) -s zeros(1,d2-d3-1) 1 zeros(1,d5-d2-1) r  ...
            zeros(1,d7-d5-1) q zeros(1,m*n*p-d7)];
end
i=m;l=p;
for j=2:n-1
        a(i+(j-1)*m+(l-1)*m^2,:)=[zeros(1,d6-1) q zeros(1,d4-d6-1) r  ...
            zeros(1,d1-d4-1) 1 zeros(1,d3-d1-1) -s zeros(1,d2-d3-1) 0  ...
            zeros(1,d5-d2-1) r zeros(1,m*n*p-d5)];
end
i=m;l=1;
for j=2:n-1
```

```
        a(i+(j-1)*m+(l-1)*m^2,:)=[zeros(1,d4-1) r zeros(1,d1-d4-1) 1  ...
            zeros(1,d3-d1-1) -s zeros(1,d2-d3-1) 0 zeros(1,d5-d2-1) r  ...
            zeros(1,d7-d5-1) q zeros(1,m*n*p-d7)];
end
i=1;l=p;
for j=2:n-1
        a(i+(j-1)*m+(l-1)*m^2,:)=[zeros(1,d6-1) q zeros(1,d4-d6-1) r  ...
            zeros(1,d1-d4-1) 0 zeros(1,d3-d1-1) -s zeros(1,d2-d3-1) 1  ...
            zeros(1,d5-d2-1) r zeros(1,m*n*p-d5)];
end
j=1;l=p;
for i=2:m-1
        a(i+(j-1)*m+(l-1)*m^2,:)=[zeros(1,d6-1) q zeros(1,d4-d6-1) 0  ...
            zeros(1,d1-d4-1) 1 zeros(1,d3-d1-1) -s zeros(1,d2-d3-1) 1  ...
            zeros(1,d5-d2-1) r zeros(1,m*n*p-d5)];
end
j=n;l=1;
for i=2:m-1
        a(i+(j-1)*m+(l-1)*m^2,:)=[zeros(1,d4-1) r zeros(1,d1-d4-1) 1  ...
            zeros(1,d3-d1-1) -s zeros(1,d2-d3-1) 1 zeros(1,d5-d2-1) 0  ...
            zeros(1,d7-d5-1) q zeros(1,m*n*p-d7)];
end
i=1;j=n;
for l=2:p-1
```

Appendix D (Continued)

```
        a(i+(j−1)*m+(l−1)*m^2,:)=[zeros(1,d6−1) q zeros(1,d4−d6−1) r  ...
            zeros(1,d1−d4−1) 0 zeros(1,d3−d1−1) −s zeros(1,d2−d3−1) 1  ...
            zeros(1,d5−d2−1) 0 zeros(1,d7−d5−1) q zeros(1,m*n*p−d7)];
end
i=m;j=1;
for l=2:p−1
        a(i+(j−1)*m+(l−1)*m^2,:)=[zeros(1,d6−1) q zeros(1,d4−d6−1) 0  ...
            zeros(1,d1−d4−1) 1 zeros(1,d3−d1−1) −s zeros(1,d2−d3−1) 0  ...
            zeros(1,d5−d2−1) r zeros(1,d7−d5−1) q zeros(1,m*n*p−d7)];
end
% eight corners
i=1;j=1;l=1;
a(i+(j−1)*m+(l−1)*m^2,:)=[−s zeros(1,d2−d3−1) 1 zeros(1,d5−d2−1) r  ...
    zeros(1,d7−d5−1) q zeros(1,m*n*p−d7)];
i=m;j=1;l=1;
a(i+(j−1)*m+(l−1)*m^2,:)=[zeros(1,d1−1) 1 −s zeros(1,d2−d3−1) 0  ...
    zeros(1,d5−d2−1) r zeros(1,d7−d5−1) q zeros(1,m*n*p−d7)];
i=1;j=n;l=1;
a(i+(j−1)*m+(l−1)*m^2,:)=[zeros(1,d4−1) r zeros(1,d1−d4−1) 0  ...
    zeros(1,d3−d1−1) −s zeros(1,d2−d3−1) 1 zeros(1,d5−d2−1) 0  ...
    zeros(1,d7−d5−1) q zeros(1,m*n*p−d7)];
i=1;j=n;l=p;
a(i+(j−1)*m+(l−1)*m^2,:)=[zeros(1,d6−1) q zeros(1,d4−d6−1) r  ...
    zeros(1,d1−d4−1) 0 zeros(1,d3−d1−1) −s zeros(1,d2−d3−1) 1  ...
    zeros(1,m*n*p−d2)];
i=1;j=1;l=p;
```

```
a(i+(j−1)*m+(l−1)*m^2,:)=[zeros(1,d6−1) q zeros(1,d4−d6−1) 0  ...
    zeros(1,d1−d4−1) 0 zeros(1,d3−d1−1) −s zeros(1,d2−d3−1) 1  ...
    zeros(1,d5−d2−1) r zeros(1,m*n*p−d5)];
i=m;j=1;l=p;
a(i+(j−1)*m+(l−1)*m^2,:)=[zeros(1,d6−1) q zeros(1,d4−d6−1) 0  ...
    zeros(1,d1−d4−1) 1 zeros(1,d3−d1−1) −s zeros(1,d2−d3−1) 0  ...
    zeros(1,d5−d2−1) r zeros(1,m*n*p−d5)];
i=m;j=n;l=1;
a(i+(j−1)*m+(l−1)*m^2,:)=[zeros(1,d4−1) r zeros(1,d1−d4−1) 1  ...
    zeros(1,d3−d1−1) −s zeros(1,d2−d3−1) 0 zeros(1,d5−d2−1) 0  ...
    zeros(1,d7−d5−1) q zeros(1,m*n*p−d7)];
i=m;j=n;l=p;
a(i+(j−1)*m+(l−1)*m^2,:)=[zeros(1,d6−1) q zeros(1,d4−d6−1) r  ...
    zeros(1,d1−d4−1) 1 zeros(1,d3−d1−1) −s];
lambda=eig((1/h2)*a);
[V,D]=eig((1/h2)*a);  %solve for solution
lamb=lambda(n*m*p,1);
v=V(:,n*m*p);
w=zeros(m,n,p); %numerical solution at grid points
for i=1:m
    for j=1:n
        for l=1:p
    w(i,j,l)=v(i+(j−1)*m+(l−1)*m^2);
end
end
end
```

```
%Boundary Conditions
        function u=g1(x,y1,z,caseNumb)
            u=g(x,y1,z,caseNumb);
        function u=g2(x,y2,z,caseNumb)
            u=g(x,y2,z,caseNumb);
        function u=g3(x,y,z1,caseNumb)
            u=g(x,y,z1,caseNumb);
        function u=g4(x,y,z2,caseNumb)
            u=g(x,y,z2,caseNumb);
        function u=g5(x1,y,z,caseNumb)
            u=g(x1,y,z,caseNumb);
        function u=g6(x2,y,z,caseNumb)
            u=g(x2,y,z,caseNumb);
```

Appendix E

CODE FOR 2D MONGE-AMPÈRE EQUATION

Finite difference solver for a $2D$ Monge-Ampère Equation with Dirichlet boundary conditions on a rectangle. The input is on a rectangle domain $[x_l, x_r] \times [y_b, y_t]$, covered by an $M \times N$ grid. The output is matrix w holding solution values on an $M \times N$ grid.

E.1 Part 1

```
function U1=Monge2d(xl,xr,yb,yt,nu,M,N,f,g,sqrt2f,caseNumb)
%initial guess
U0=poisson2d(xl,xr,yb,yt,M,N,sqrt2f,g,caseNumb);
cvg = 0;
it_count = 0;
max_it = 1000;
tol = 1.0E-10;
oldErr = inf;
newErr = 10000;
m=M-1;n=N-1;
xl=0;yb=0;
h=1/M; k=1/N;
%Exact solution
utrue = zeros(m,n);
```

```
for i=1:m
for j=1:n
utrue(i,j) = g((xl+i*h),(yb+j*k),caseNumb);
end
end
while ¬cvg & (it_count ≤ max_it) & abs(newErr) ≤ abs(oldErr)
  F1 = RightHand2d(xl,xr,yb,yt,M,N,nu,U0,f,g,caseNumb);
  U1=poisson2dv2(xl,xr,yb,yt,M,N,F1,g,caseNumb);
  oldErr = newErr;
  newErr = norm(U1-utrue,inf);
  cvg = norm(U1-U0,inf) ≤ tol;
  it_count = it_count + 1;
  U0 = U1;
end;
```

E.2 Part 2

```
function F1 = RightHand2d(xl,xr,yb,yt,M,N,nu,U0,f,g,caseNumb)
m=M-1;n=N-1;
h=(xr-xl)/M;k=(yt-yb)/N;
x=xl+(xr-xl)*(0:M)/M;
y=yb+(yt-yb)*(0:N)/N;
U0xx=zeros(size(U0));U0xy=U0xx;U0yy=U0xx; %initialization
%inner core
for i=2:n-1
```

```
    for j=2:m-1
        U0xx(i,j) = 1/h^2*(U0(i+1,j)-2*U0(i,j)+U0(i-1,j));
        U0yy(i,j) = 1/h^2*(U0(i,j+1)-2*U0(i,j)+U0(i,j-1));
        U0xy(i,j) = 1/(4*h^2)*(U0(i+1,j+1)+ U0(i-1,j-1)- U0(i-1,j+1)-  ...
            U0(i+1,j-1));
    end
end
%four corners
i=1;j=1;
        U0xx(i,j) = 1/h^2*(U0(i+1,j)-2*U0(i,j)+gleft(xl,y(j+1),caseNumb));
        U0yy(i,j) = 1/h^2*(U0(i,j+1)-2*U0(i,j)+gbottom(x(i+1),yb,caseNumb));
        U0xy(i,j) = 1/(4*h^2)*(U0(i+1,j+1)+ gleft(xl,yb,caseNumb)-  ...
            gleft(xl,y(j+2),caseNumb)- gbottom(x(i+2),yb,caseNumb));
i=n;j=1;
        U0xx(i,j) = 1/h^2*(gright(xr,y(j+1),caseNumb)-2*U0(i,j)+U0(i-1,j));
        U0yy(i,j) = 1/h^2*(U0(i,j+1)-2*U0(i,j)+gbottom(x(i+1),yb,caseNumb));
        U0xy(i,j) = 1/(4*h^2)*(gright(xr,y(j+2),caseNumb)+  ...
            gbottom(x(i),yb,caseNumb)- U0(i-1,j+1)- gright(xr,yb,caseNumb));
i=1;j=m;
        U0xx(i,j) = 1/h^2*(U0(i+1,j)-2*U0(i,j)+gleft(xl,y(j+1),caseNumb));
        U0yy(i,j) = 1/h^2*(gtop(x(i+1),yt,caseNumb)-2*U0(i,j)+U0(i,j-1));
        U0xy(i,j) = 1/(4*h^2)*(gtop(x(i+2),yt,caseNumb)+  ...
            gleft(xl,y(j),caseNumb)- gleft(xl,y(j+2),caseNumb)- U0(i+1,j-1));
i=n;j=m;
        U0xx(i,j) = 1/h^2*(gright(xr,y(j+1),caseNumb)-2*U0(i,j)+U0(i-1,j));
        U0yy(i,j) = 1/h^2*(gtop(x(i+1),yt,caseNumb)-2*U0(i,j)+U0(i,j-1));
        U0xy(i,j) = 1/(4*h^2)*(gright(xr,yt,caseNumb)+ U0(i-1,j-1)-  ...
            gtop(x(i),yt,caseNumb)-  gright(xr,y(j),caseNumb));
%outer ring
j=1;
for i=2:m-1
        U0xx(i,j) = 1/h^2*(U0(i+1,j)-2*U0(i,j)+U0(i-1,j));
        U0yy(i,j) = 1/h^2*(U0(i,j+1)-2*U0(i,j)+gbottom(x(i+1),yb,caseNumb));
        U0xy(i,j) = 1/(4*h^2)*(U0(i+1,j+1)+ gbottom(x(i),yb,caseNumb)-  ...
            U0(i-1,j+1)- gbottom(x(i+2),yb,caseNumb));
end
j=m;
for i=2:m-1
        U0xx(i,j) = 1/h^2*(U0(i+1,j)-2*U0(i,j)+U0(i-1,j));
        U0yy(i,j) = 1/h^2*(gtop(x(i+1),yt,caseNumb)-2*U0(i,j)+U0(i,j-1));
        U0xy(i,j) = 1/(4*h^2)*(gtop(x(i+2),yt,caseNumb)+ U0(i-1,j-1)-  ...
            gtop(x(i),yt,caseNumb)- U0(i+1,j-1));
end
i=1;
for j=2:n-1
        U0xx(i,j) = 1/h^2*(U0(i+1,j)-2*U0(i,j)+gleft(xl,y(j+1),caseNumb));
        U0yy(i,j) = 1/h^2*(U0(i,j+1)-2*U0(i,j)+U0(i,j-1));
        U0xy(i,j) = 1/(4*h^2)*(U0(i+1,j+1)+ gleft(xl,y(j),caseNumb)-  ...
            gleft(xl,y(j+2),caseNumb)- U0(i+1,j-1) );
end
i=m;
for j=2:n-1
```

```
          U0xx(i,j) = 1/h^2*(gright(xr,y(j+1),caseNumb)-2*U0(i,j)+U0(i-1,j));
          U0yy(i,j) = 1/h^2*(U0(i,j+1)-2*U0(i,j)+U0(i,j-1));
          U0xy(i,j) = 1/(4*h^2)*(gright(xr,y(j+2),caseNumb)+ U0(i-1,j-1)-  ...
              U0(i-1,j+1)- gright(xr,y(j),caseNumb));
end
%right hand side of the monge-ampere equation evaluated at the grid points
fmonge = zeros(m,n);
for i=1:m
for j=1:n
fmonge(i,j) = f((xl+i*h),(yb+j*k),caseNumb);
end
end
DetU0 = U0xx.*U0yy-U0xy.^2;
F1 = (U0xx+U0yy)-(1/nu)*(DetU0-fmonge);
%boundary conditions
function u=gbottom(x,yb,caseNumb) % bottom of rectangle
u=g(x,yb,caseNumb);
function u=gtop(x,yt,caseNumb) % top of rectangle
u=g(x,yt,caseNumb);
function u=gleft(xl,y,caseNumb) % left side of rectangle
u=g(xl,y,caseNumb);
function u=gright(xr,y,caseNumb) % right side of rectangle
u=g(xr,y,caseNumb);
```

Appendix F

CODE FOR 2D MONGE-AMPÈRE EIGENVALUE PROBLEM

Finite difference solver for a $2D$ Monge-Ampère eigenvalue problem with Dirichlet boundary conditions on a rectangle. The input is on a rectangle domain $[x_l, x_r] \times [y_b, y_t]$, covered by an $M \times N$ grid. The output is matrix w holding solution values on an $M \times N$ grid.

F.1 Part 1

```
function U1=Monge2deig(xl,xr,yb,yt,nu,M,N,f,g,sqrt2f,caseNumb)
%Example 1 Initial guess
U0=poisson2d(xl,xr,yb,yt,M,N,sqrt2f,g,caseNumb);
lambda0= 48;
%Example 2 Initial guess
[U0,lambda0]=laplacian2deig(xl,xr,yb,yt,M,N,g,caseNumb)
cvg = 0;
it_count = 0;
max_it = 1000;
tol = 1.0E-10;
oldErr = inf;
newErr = 10000;
m=M-1;n=N-1;
%True solution
xl=0;yb=0;
```

```
h=1/M; k=1/N;
utrue = zeros(m,n);
for i=1:n
for j=1:m
utrue(i,j) = g((xl+i*h),(yb+j*k),caseNumb);
end
end
while ¬cvg & (it_count ≤ max_it) & newErr ≤ oldErr
  [F1,DetU0,lambda1] = RightHand2deig(xl,xr,yb,yt,M,N,nu,U0,lambda0,g,caseNumb);
  U1= poisson2dv2(xl,xr,yb,yt,M,N,F1,g,caseNumb);
  oldErr = newErr;
  newErr = norm(U1−utrue,inf);
  cvg = norm(U1−U0,inf) ≤ tol;
  it_count = it_count + 1;
  U0 = U1;
  lambda0 = lambda1;
end;
```

F.2 Part 2

```
function [F1,DetU0,lambda1] = ...
    RightHand2deig(xl,xr,yb,yt,M,N,nu,U0,lambda0,g,caseNumb)
m=M−1;n=N−1;
h=(xr−xl)/M;k=(yt−yb)/N;
x=xl+(xr−xl)*(0:M)/M;
```

Appendix F (Continued)

```
y=yb+(yt−yb)*(0:N)/N;
U0xx=zeros(size(U0));U0xy=U0xx;U0yy=U0xx; %initialization
%inner core
for i=2:n−1
    for j=2:m−1
        U0xx(i,j) = 1/h^2*(U0(i+1,j)−2*U0(i,j)+U0(i−1,j));
        U0yy(i,j) = 1/h^2*(U0(i,j+1)−2*U0(i,j)+U0(i,j−1));
        U0xy(i,j) = 1/(4*h^2)*(U0(i+1,j+1)+ U0(i−1,j−1)− U0(i−1,j+1)−  ...
            U0(i+1,j−1));
    end
end
%four corners
i=1;j=1;
        U0xx(i,j) = 1/h^2*(U0(i+1,j)−2*U0(i,j)+gleft(xl,y(j+1),caseNumb));
        U0yy(i,j) = 1/h^2*(U0(i,j+1)−2*U0(i,j)+gbottom(x(i+1),yb,caseNumb));
        U0xy(i,j) = 1/(4*h^2)*(U0(i+1,j+1)+ gleft(xl,yb,caseNumb)−  ...
            gleft(xl,y(j+2),caseNumb)− gbottom(x(i+2),yb,caseNumb));
i=n;j=1;
        U0xx(i,j) = 1/h^2*(gright(xr,y(j+1),caseNumb)−2*U0(i,j)+U0(i−1,j));
        U0yy(i,j) = 1/h^2*(U0(i,j+1)−2*U0(i,j)+gbottom(x(i+1),yb,caseNumb));
        U0xy(i,j) = 1/(4*h^2)*(gright(xr,y(j+2),caseNumb)+  ...
            gbottom(x(i),yb,caseNumb)− U0(i−1,j+1)− gright(xr,yb,caseNumb));
i=1;j=m;
        U0xx(i,j) = 1/h^2*(U0(i+1,j)−2*U0(i,j)+gleft(xl,y(j+1),caseNumb));
        U0yy(i,j) = 1/h^2*(gtop(x(i+1),yt,caseNumb)−2*U0(i,j)+U0(i,j−1));
```

Appendix F (Continued)

```
        U0xy(i,j) = 1/(4*h^2)*(gtop(x(i+2),yt,caseNumb)+  ...
            gleft(xl,y(j),caseNumb)- gleft(xl,y(j+2),caseNumb)- U0(i+1,j-1));
i=n;j=m;
        U0xx(i,j) = 1/h^2*(gright(xr,y(j+1),caseNumb)-2*U0(i,j)+U0(i-1,j));
        U0yy(i,j) = 1/h^2*(gtop(x(i+1),yt,caseNumb)-2*U0(i,j)+U0(i,j-1));
        U0xy(i,j) = 1/(4*h^2)*(gright(xr,yt,caseNumb)+ U0(i-1,j-1)-  ...
            gtop(x(i),yt,caseNumb)-  gright(xr,y(j),caseNumb));
%outer ring
j=1;
for i=2:m-1
        U0xx(i,j) = 1/h^2*(U0(i+1,j)-2*U0(i,j)+U0(i-1,j));
        U0yy(i,j) = 1/h^2*(U0(i,j+1)-2*U0(i,j)+gbottom(x(i+1),yb,caseNumb));
        U0xy(i,j) = 1/(4*h^2)*(U0(i+1,j+1)+ gbottom(x(i),yb,caseNumb)-  ...
            U0(i-1,j+1)- gbottom(x(i+2),yb,caseNumb));
end
j=m;
for i=2:m-1
        U0xx(i,j) = 1/h^2*(U0(i+1,j)-2*U0(i,j)+U0(i-1,j));
        U0yy(i,j) = 1/h^2*(gtop(x(i+1),yt,caseNumb)-2*U0(i,j)+U0(i,j-1));
        U0xy(i,j) = 1/(4*h^2)*(gtop(x(i+2),yt,caseNumb)+ U0(i-1,j-1)-  ...
            gtop(x(i),yt,caseNumb)- U0(i+1,j-1));
end
i=1;
for j=2:n-1
        U0xx(i,j) = 1/h^2*(U0(i+1,j)-2*U0(i,j)+gleft(xl,y(j+1),caseNumb));
        U0yy(i,j) = 1/h^2*(U0(i,j+1)-2*U0(i,j)+U0(i,j-1));
        U0xy(i,j) = 1/(4*h^2)*(U0(i+1,j+1)+ gleft(xl,y(j),caseNumb)-  ...
            gleft(xl,y(j+2),caseNumb)- U0(i+1,j-1) );
end
i=m;
for j=2:n-1
        U0xx(i,j) = 1/h^2*(gright(xr,y(j+1),caseNumb)-2*U0(i,j)+U0(i-1,j));
        U0yy(i,j) = 1/h^2*(U0(i,j+1)-2*U0(i,j)+U0(i,j-1));
        U0xy(i,j) = 1/(4*h^2)*(gright(xr,y(j+2),caseNumb)+ U0(i-1,j-1)-  ...
            U0(i-1,j+1)- gright(xr,y(j),caseNumb));
end
DetU0 = U0xx.*U0yy-U0xy.^2;
F1 = (U0xx+U0yy)-(1/nu)*(DetU0-lambda0*(U0));
%Example 1 update for $\lambda_k$
lambda1 = (mean(mean(DetU0/U0)));
%Example 2 update for $\lambda_k$
lambda1=(1/((h^4)*(sum(sum(U0.^2)))))*(sum(sum((U0xx*U0yy)-((U0xy.^2)/16))));
%boundary conditions
function u=gbottom(x,yb,caseNumb) % bottom of rectangle
u=g(x,yb,caseNumb);
function u=gtop(x,yt,caseNumb) % top of rectangle
u=g(x,yt,caseNumb);
function u=gleft(xl,y,caseNumb) % left side of rectangle
u=g(xl,y,caseNumb);
function u=gright(xr,y,caseNumb) % right side of rectangle
u=g(xr,y,caseNumb);
```

CITED LITERATURE

Gerard Awanou, *Pseudo time continuation and time marching methods for Monge-Ampère type Equations*, http://ima.umn.edu/preprints/nov2010/2350.pdf, 2010.11.22.

Gerard Awanou, *Pseudo transient continuation and time marching methods for Monge-Ampère type Equations*, http://arxiv.org/pdf/1301.5891.pdf, 2010.11.22.

Nunzia Gavitone, *Hessian equations, quermassintegrals and symmetrization*, Ph.D. thesis, Universita di Napoli Federico II, 2009.

Richard Haberman, *Applied Partial Differential Equations with Fourier Series and Boundary Value Problems, 4th ed.*, Prentice Hall, Upper Saddle River, NJ, 2004.

Nicholas J. Korevaar, *Convex solutions to nonlinear elliptic and parabolic boundary value problems*, Indiana Univ. Math. J. **32** (1983), no. 4, 603–614.

H.-D. Mittelmann, *A fast solver for nonlinear eigenvalue problems*, Iterative solution of nonlinear systems of equations (Oberwolfach, 1982), Lecture Notes in Math., vol. 953, Springer, Berlin, 1982, pp. 46–67.

Tim D. Sauer, *Numerical Analysis*, Addison-Wesley, Boston, MA, 2006.

Xu Jia Wang, *A class of fully nonlinear elliptic equations and related functionals*, Indiana Univ. Math. J. **43** (1994), no. 1, 25–54.

VITA

NAME:	Kara Neely
EMAIL:	kjneely1986@gmail.com
EDUCATION:	B.S., Mathematics, Spelman College, Atlanta, GA, 2008
	M.S., Applied Mathematics, University of Illinois at Chicago, Chicago, Illinois, 2013
TEACHING:	Department of Mathematics, Prairie State College, Chicago Heights, IL 2013
	Department of Mathematics, Moraine Valley Community College, Palos Hills, IL, 2012
	Department of Mathematics, East-West University, Chicago, IL, 2012
HONORS:	Pi Mu Epsilon, Spelman College 2007-2008
	Dean's List, Spelman College, 2004-2008
PROFESSIONAL EXPERIENCE:	Business Technology Analyst, Deloitte Consulting, Atlanta, GA, 2008-2010
	Actuarial Intern, New York Life, New York, NY, 2007