Nonlinear Elliptic Boundary Value Problems: A Numerical Approach.

by Michael Butros

A Thesis
Submitted in Partial Fulfillment
of the Requirements for the Degree of
Master of Science
in Mathematics

Northern Arizona University
August, 2000

Approved:

____
John M. Neuberger, Ph.D., Chair

____
Lawrence M. Perko, Ph.D.

____
James W. Swift, Ph.D.

Abstract

Nonlinear Elliptic Boundary Value Problems: A Numerical Approach.

Michael Butros

We study the nonlinear elliptic BVP

$$\begin{cases} \Delta u + f(u) = 0 & \text{in } \Omega \\ u = 0 & \text{on } \partial\Omega, \end{cases}$$

where Δ is the Laplacian operator, $\Omega \subseteq \mathbb{R}^2$ is the disk, $B_0(1)$, centered at the origin with radius $r = 1$. The nonlinear function $f : \mathbb{R} \longrightarrow \mathbb{R}$ satisfies $f(0) = 0$, and growth conditions $\lim_{|u|\longrightarrow\infty} \frac{f(u)}{u} = \infty$ and $f'(u) > \frac{f(u)}{u}$. The function we will consider for this work is $f(u) = \lambda u + u^3$. We seek solutions $u : \Omega \longrightarrow \mathbb{R}$ satisfying the BVP.

We will use Fourier expansion via an orthonormal basis. Our orthonormal basis is formed using Bessel functions. The zeroes of the Bessel functions determine the eigenvalues of $-\Delta$ on the disk.

The action functional on the Hilbert space $H = H_0^{1,2}$, $J : H \longrightarrow \mathbb{R}$, is defined to be

$$J(u) = \int (\frac{1}{2} \mid \nabla u \mid^2 - F(u)),$$

where $F(u) = \int_0^u f(s)\,ds$ is the primitive of $f(u)$.

The critical points of the J are then approximated using Newton's method. These critical points are solutions to the BVP. Using $\lambda = f'(0)$ as a parameter, a bifurcation diagram is then generated for several branches of the solutions to the BVP.

Acknowledgements

First I would like to thank my advisor, Dr. John M. Neuberger, for providing me with such an interesting topic. But more importantly, I would like to thank him for his constant assistance and support through every stage of this thesis. I have taken on this project without any prior experience in mathematical research or computer programming. His patience and teaching methods have taught me a great deal, not only about how to conduct mathematical research, but also on being a concerned academic advisor. This experience proved to me how fortunate I was to have Dr. Neuberger as my advisor. I truly hope that many NAU students get the chance to work with Dr. Neuberger in the future, and learn from him as I did. Dr. Neuberger has had a great influence on me in terms of developing my mathematical intuition, research skills, and my scientific writing skills. **I AM VERY GRATEFUL TO YOU DR. NEUBERGER.**

A special thanks goes to the other members of my committee, Dr. Lawrence Perko, and Dr. James Swift. Their comments and support also helped tremendously in producing this document. I also would like to thank Jacob Louchart for his assistance in editing this document, and Dr. Michael Falk for providing the necessary files to write this document in LaTeX.

My deepest gratitude and love goes out to my family who have been there for me in all my endeavors. Thank you for your never ending love and support, I am certain that I would not have realized this dream without you. All that I am today, and all I will ever be in the future, I owe to you. **I LOVE YOU ALL DEARLY.**

Contents

List of Tables

List of Figures

Chapter 1

INTRODUCTION AND PRELIMINAIRES

1.1 Introduction

The subject of partial differential equations, PDE, has many real-life applications. Nonlinear elliptic PDE have been used in physical problems such as fluid dynamics, chemical reactions, and steady state solutions of reaction-diffusion equations.

When studying a nonlinear PDE, one might be interested in finding solutions $u : \Omega \longrightarrow \mathbb{R}$ which satisfy some boundary value problem, BVP. One might also be interested in the nodal structure of the solutions. These BVP, in most cases, do not have closed form solutions. In other words, we are not able to obtain a function in closed form which satisfies the boundary value problem, and so, we need to use numerical algorithms to generate approximate solutions.

Our nonlinear elliptic boundary value problem has the form

$$\begin{cases} \Delta u + f(u) = 0 & \text{in } \Omega \\ u = 0 & \text{on } \partial\Omega, \end{cases} \tag{1.1}$$

where Δ is the Laplacian operator, Ω is a smooth bounded region in $\mathbb{R}^n$, and $f : \mathbb{R} \longrightarrow \mathbb{R}$, satisfies several assumptions. We will examine the BVP (1.1) on $\Omega = B_0(1)$, the disk in $\mathbb{R}^2$ centered at the origin with radius $r = 1$, and for nonlinearities f satisfying $f'(u) > \frac{f(u)}{u}$, $\lim_{|u| \longrightarrow \infty} \frac{f(u)}{u} = \infty$, and

$f(0) = 0$, among other conditions. The function we will consider is $f(u) = \lambda u + u^3$. It is important to realize that while known existence theorems rely on such specific assumptions on f, we may execute our numerical algorithm on a much wider class of nonlinearities.

It is well known that the eigenvalues of $-\Delta$ with zero Dirichlet boundary conditions satisfy

$$0 < \lambda_1 < \lambda_2 \leq \cdots \longrightarrow \infty.$$

That is, they are all positive, the first is simple (nonrepeated), and they increase without bound (see [7]). Let $\{\Psi_i\}$ denote the set of corresponding eigenfunctions normalized in $L^2 = L^2(\Omega)$, so that

$$\int_\Omega \Psi_i^2\, dx = 1 \text{ and } \int_\Omega \Psi_i \Psi_j\, dx = 0, \text{ for } i \neq j.$$

Nonlinear elliptic PDE problems have been the focus of many studies in the past 70 years. In certain special cases it has been shown that there are infinitely many solutions to the BVP (1.1). The case where $n = 1$, the ordinary differential equation case, has infinitely many solutions, as we will discuss in section 4.1. Also, nontrivial solutions to the general problem (1.1) have been found among the three different types: positive, negative and sign-changing.

Since we are studying the BVP on the disk, we will use polar coordinates instead of rectangular coordinates to obtain our solutions. We will discuss how this change in coordinates leads us to Bessel's differential equation. The relationship between the eigenvalues of the Laplacian on the disk and the zeroes of the Bessel functions is discussed. In section 3.2 we build the orthonormal basis necessary for the numerical algorithm, and discuss the methodology behind the numerical algorithm.

We use a variational method to obtain our solutions. In order to do that, we will define an action functional J on the Hilbert space $H = H_0^{1,2}$. Note that H is a subspace of L^2 with corresponding inner products

$$< u, v >_H = \int_\Omega \nabla u(x) \cdot \nabla v(x)\, dx, \quad < u, v >_2 = \int_\Omega uv\, dx,$$

and norms

$$\| u \|_H = (\int_\Omega | \nabla u |^2\, dx)^{1/2}, \quad \| u \|_2 = (\int_\Omega u^2 dx)^{\frac{1}{2}},$$

where ∇ denotes a generalized gradient

$$\nabla u = \left(\frac{\partial u}{\partial x_1}, \dots, \frac{\partial u}{\partial x_n}\right) \in (L_2(\Omega))^n.$$

The zeroes of the gradient of this action functional J are sought. As we will discuss in Chapter 3, these zeroes are precisely the solutions to (1.1).

1.2 Preliminaries

In this section we will look at some definitions and concepts that are of importance for the remainder of the project. Throughout this project we will use the notation u_x to represent $\frac{\partial u}{\partial x}$, u_y to represent $\frac{\partial u}{\partial y}$, and so on.

1.2.1 Laplacian in Polar Coordinates

In this section we will show how to obtain the Laplacian operator in polar coordinates. We use the transformations: $x = r\sin(\theta)$, $y = r\sin(\theta)$, and $x^2 + y^2 = r^2$. Differentiating the last equation with respect to x and y, respectively, we get

$$2rr_x = 2x \Rightarrow r_x = \cos(\theta)$$

and

$$2rr_y = 2y \Rightarrow r_y = \sin(\theta).$$

Applying the product rule to $x = r\cos(\theta)$ we obtain

$$1 = -r\sin(\theta)(\theta_x) + (r_x)\cos(\theta),$$

$$1 = -r\sin(\theta)(\theta_x) + \cos^2(\theta),$$

so,

$$\theta_x = -\frac{\sin(\theta)}{r}.$$

Similar arguments show that

$$\theta_y = \frac{\cos(\theta)}{r}.$$

Next, we find expressions for u_x and u_y. In particular,

$$u_x = (u_r)(r_x) + (u_\theta)(\theta_x),$$

so,

$$u_x = u_r \cos(\theta) - u_\theta \frac{\sin(\theta)}{r}.$$

Similarly,

$$u_y = u_r \sin(\theta) + u_\theta \frac{\cos(\theta)}{r}.$$

Using the same method to obtain expressions for u_{xx} and u_{yy}, we see that

$$\begin{aligned} u_{xx} &= (u_{xr})(r_x) + (u_{x\theta})(\theta_x) \\ &= (\cos(\theta) u_{rr} - (\tfrac{\sin(\theta)}{r}) u_{\theta r} + (\tfrac{\sin(\theta)}{r^2}) u_\theta) \cos(\theta) \\ &\quad + (\cos(\theta) u_{\theta r} - \sin(\theta)) u_r - (\tfrac{\sin(\theta)}{r}) u_{\theta\theta} (\tfrac{\cos(\theta)}{r}) u_\theta - \tfrac{\sin(\theta)}{r}. \end{aligned}$$

Thus, u_{xx} can be written as

$$u_{xx} = \cos^2(\theta) u_{rr} - (\frac{2\sin(\theta)\cos(\theta)}{r}) u_{\theta r}$$

$$+(\frac{(2\sin(\theta)\cos(\theta)}{r^2}) u_\theta + (\frac{\sin^2(\theta)}{r}) u_r + (\frac{\sin^2(\theta)}{r^2}) u_{\theta\theta}.$$

Using similar arguments we get

$$u_{yy} = (u_{yr})(r_y) + (u_{y\theta})(\theta_y).$$

Hence,

$$u_{yy} = \sin^2(\theta)u_{rr} + (\frac{2\sin(\theta)\cos(\theta)}{r})u_{\theta r}$$

$$-(\frac{2\sin(\theta)\cos(\theta)}{r^2})u_\theta + (\frac{\cos^2(\theta)}{r})u_r + (\frac{\cos^2(\theta)}{r^2})u_{\theta\theta}.$$

Thus, the Laplacian in polar coordinates can be written as

$$\Delta u = u_{xx} + u_{yy} = u_{rr} + (\frac{1}{r})u_r + (\frac{1}{r^2})u_{\theta\theta}.$$

Note that in the radially symmetric case where $u = u(r)$, we have $u_\theta = 0$ so
$\Delta u = u_{rr} + \frac{1}{r}u_r$.

1.2.2 Bessel Functions

In this section we will demonstrate how the zeroes of the Bessel functions form the eigenvalues of $-\Delta$ on the disk.

Consider the eigenvalue problem

$$\Delta u + \lambda u = 0,$$

with zero Dirichlet boundary conditions. Use separation of variables, i.e., let $u = f(r)\cos(m\theta)$ to obtain

$$\Delta u + \lambda u = \cos(m\theta)\left(f_{rr} + \frac{1}{r}f_r + (\lambda - \frac{m^2}{r^2})f\right) = 0.$$

Using $z = \sqrt{\lambda}\,r$ leads to Bessel's differential equation, see [17]

$$z^2(f_{zz}) + z(f_z) + (z^2 - m^2)f = 0. \tag{1.2}$$

Equation (1.2) is known to have solutions, (see [17]), of the form

$$f_m(r) = c_1 J_m(\sqrt{\lambda}r) + c_2 Y_m(\sqrt{\lambda}r),$$

where $J_m(r)$ and $Y_m(r)$ are Bessel functions of the first kind.

Now, by the zero Dirichlet conditions, and since $Y_m(\sqrt{\lambda}\, r),$ is not bounded at $r = 0,$ our solutions are forced to be of the form

$$f_m(r) = c_1 J_m(\sqrt{\lambda} r)$$

with $f_m(1) = 0$, thus $\sqrt{\lambda}$ is a zero of $J_m(r)$.
Thus, the zeroes of the Bessel function determine the eigenvalues of the Laplacian, as well as the corresponding eigenfunctions to those eigenvalues. We will refer to an eigenvalue-eigenfunction pair as an *eigenpair*.

1.2.3 Newton's Method

The method discussed in this section, which is used to find zeroes of functions, will be limited to the one dimensional case. In section 3.4 we will demonstrate how this method may be extended to higher dimensions and show how the higher dimensional method was used in our program.

Derivation of Newton's Method

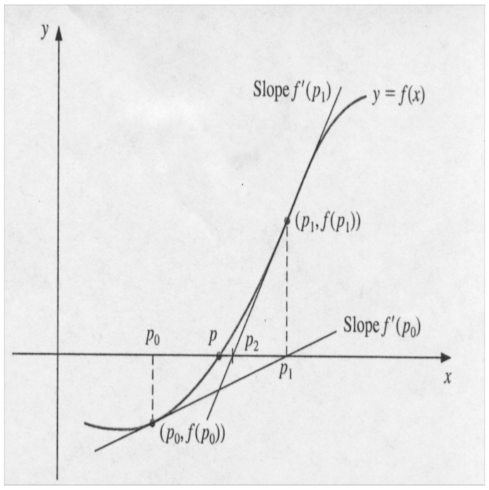

Figure 1.1: Diagram of Newton's Method

The objective here is to obtain the x-intercept of the the tangent line of a function at a given point. Let $y = mx + b$ be the the equation of the tangent line. Solving for the y-intercept b, we get

$$b = y - mx.$$

At a specific point $(x_n, f(x_n))$, m may be replaced with $f'(x_n)$ and so we have

$$b = f(x_n) - f'(x_n)x_n.$$

Hence,

$$y = f'(x_n)x + f(x_n) - f'(x_n)x_n \ = f'(x_n)(x - x_n) + f(x_n).$$

Next, we find the x-intercept of the above equation, by solving

$$0 = f'(x_n)(x - x_n) + f(x_n)$$

for x. Renaming the x-intercept to be called x_{n+1}, the above equation yields

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}. \tag{1.3}$$

This is the equation for Newton's method.

Newton's Methods Theorem

The following theorem is used to show sufficient conditions under which Newton's method convergers to a solution, (see [2]).

Theorem 1.2.1 *Let $f \in C^2[a, b]$. If $p \in (a, b)$ is such that $f(p) = 0$ and $f'(p) \neq 0$, then there exists a $\delta > 0$ such that Newton's method generates a sequence $\{p_n\}_{n=1}^{\infty}$ converging to p for any $p_0 \in [p - \delta, p + \delta]$.*

Note: In [2] the theorem states $p \in [a, b]$.

Chapter 2

HISTORY AND PREVIOUS WORK

In this chapter, we mention some of the work that has been done on this problem and some major results that helped in better understanding our problem. The reader is encouraged to check the referenced work for more details on the content of these works.

2.1 Odd Functions Case

In the 1920's, Ljusternik and Schnirelmann (see [12]) showed that if f was an odd function, then the BVP (1.1) has infinitely many solutions. Even though their result showed that infinitely many solutions exist, only three nontrivial solutions have been found in the general non-odd case (see [4]). The function we are considering for our project is odd, $f(u) = \lambda u + u^3$. Our numerical results supports that there are indeed, infinitely many solutions for the case when f is odd (see Chapter 4).

2.2 Two Types of Solutions

In 1973, Ambrosetti and Rabinowitz introduced the famous *Mountain Pass Lemma*, MPL, (see [1]). They used the MPL to show that in the general case (1.1) has at least two nontrivial solutions: one positive and one negative. That is to say, there exists solutions u_1 and u_2 satisfying (1.1), with $u_1(x) \geq$

0 and $u_2(x) \leq 0$, $\forall x \in \Omega$. Both solutions can be obtained by our algorithm and are discussed in Chapter 4.

2.3 Radially Symmetric Solutions on the Disk

In 1987, Castro and Kurepa (see [8]) showed that there are infinitely many radially symmetric solutions to (1.1) when Ω is the ball. They accomplished this by analyzing the energy and nodal structure of the solutions. This work is of importance to us since we are looking at the disk as the region. We would like to study the existence of solutions which are non-radially symmetric as well as radially symmetric. Both types of solutions can be found using our algorithm and they will be discussed in Chapter 4. As far as we know, it is not known whether there are infinitely many non-radially symmetric solutions.

2.4 Multiple Solutions: More than Two

The work of Castro, Cossio, and Neuberger, (see [4], [5], [6], [3], and [13]), has been of great help in this project. The authors of [4] proved the existence of a third nontrivial solution, u_3, besides the positive and negative solutions, to (1.1). This new solution, which we will refer to as the *CCN* solution, changes signs exactly once. That is, $u_3^{-1}((0,\infty))$ and $u_3^{-1}((-\infty,0))$ are connected nonempty subsets of Ω. The set $u^{-1}(0)$ is called the internal zero set, and the so-called *zero-set conjecture* (see [13]) states that $u^{-1}(0)$ intersects the boundary $\partial\Omega$.

The work of Choi and McKenna (see [9]) is also of importance to us. They used the MPL, to generate an algorithm which is considered to be the first in the spirit of our methods. The Mountain Pass Algorithm (MPA) (see [9]) was also developed independently by Neuberger (see [13]). The MPA can be used to approximate solutions which have only one sign. Neuberger also developed a sign-changing algorithm which has been refered to as the Modified Mountain Pass Algorithm (MMPA) (see [13]). There was then a need for a new algorithm due to the fact that MPA and MMPA are only capable of handling certain types of solutions. They can be used to find critical points of *Morse index* as high as 2, but not higher. Roughly speaking, the *Morse* index of a critical point is the number of linearly independent "down" directions in function space of a functional.

2.5 The BVP on the Square

Here we cover the recent work of Neuberger and Swift (see [14]), who investigated (1.1) on the region $\Omega = [0,1] \times [0,1]$. The eigenpairs of $-\Delta$, on the square are known to be of the form

$$\lambda_{m,n} = (m^2 + n^2)\pi^2 \quad \text{and} \quad \psi_{m,n} = 2\sin(m\pi x)\sin(n\pi y),$$

where $\lambda_{m,n}$ are the eigenvalues, and $\psi_{m,n}$ are the corresponding eigenfunctions of $-\Delta$ on the square with zero Dirichlet boundary conditions.

Neuberger and Swift used Newton's method, as we do here, to obtain the critical points of the action functional $J(u)$. They sought the *Morse index* of these critical points and analyzed the nodal structure of the solutions.

The algorithm the authors use in [14], which is the same as the algorithm used in this project, uses a much simpler orthonormal basis than ours. The algorithm in theory can be used for a wide variety of nonlinearities f and on different regions Ω. The authors also show that the L^2 inner product, norm, gradient, and Hessian perform as well as the *Sobolev* counterparts.

The authors discuss the symmetry of the PDE on the square, as well as the various bifurcations that occur from the trivial solution. In this project we also obtain a bifurcation diagram using the bifurcation parameter $\lambda = f'(0)$. We also investigate the possibility of secondary and tertiary bifurcation points similar to those found in [14].

Chapter 3

METHODOLOGY

In this section we discuss the methods we used to obtain approximate solutions to the nonlinear elliptic PDE (1.1).

3.1 Variational Methods

In this project we will use a variational technique to obtain approximate solutions to our PDE. A variational technique considers calculating the critical points of an action functional, in the cases where the critical points can be shown to be solutions to the BVP.

We define our action functional $J : H \longrightarrow \mathbb{R}$ by

$$J(u) = \int (\frac{1}{2} \mid \nabla u \mid^2 - F(u)),$$

where $F(u) = \int_0^u f(s)ds \, , \forall u \in \mathbb{R}$, defines the primitive F of f.

3.1.1 Critical Points Are Solutions

By regularity theory for elliptic BVP (see [10]) u , is a solution of (1.1) if and only if u is a critical point of the action functional $J(u)$.

Here, we sketch a proof of why the critical points of the action functional defined above are solutions to the BVP. Note that under our assumptions on f, $J(u)$ is C^2 on H (see [4]).

The directional derivative of J at u in the v direction is given by

$$J'(u)(v) = \lim_{t \longrightarrow 0} \frac{J(u + tv) - J(u)}{t}.$$

Applying the definition of the action functional we get

$$J'(u)(v) = \lim_{t\longrightarrow 0} \frac{\int_\Omega\{\frac{1}{2}(\nabla(u+tv))^2-F(u+tv)\}-\int_\Omega\{\frac{1}{2}(\nabla u)^2-F(u)\}}{t}$$

$$= \lim_{t\longrightarrow 0} \frac{\int_\Omega\{\frac{1}{2}(\nabla u)^2+\nabla u\cdot t\nabla v+\frac{1}{2}t^2(\nabla v)^2-F(u+tv)-\frac{1}{2}(\nabla u)^2+F(u)\}}{t}$$

$$= \lim_{t\longrightarrow 0} \frac{\int_\Omega\{\nabla u\cdot t\nabla v+\frac{1}{2}t^2(\nabla v)^2-(F(u+tv)-F(u))\}}{t}$$

$$= \int_\Omega \nabla u\cdot\nabla v - \lim_{t\longrightarrow 0}\int_\Omega \tfrac{1}{2}t(\nabla v)^2 - \lim_{t\longrightarrow 0}\int_\Omega \frac{(F(u+tv)-F(u))}{t}.$$

By the Lebesgue Dominated Convergence Theorem, we can move the "lim" inside the integral sign and calculate the limit to obtain

$$J'(u)(v) = \int_\Omega \nabla u\cdot\nabla v - \int_\Omega f(u)(v).$$

Using similar arguments we can obtain

$$J''(u)(v,w) = \int_\Omega \{\nabla v\cdot\nabla w - f'(u)vw\}.$$

Next, the so-called "bootstrap" method shows that u being a critical point implies $u \in C^2[a,b]$, (see [10]). Thus we can integrate the expression $\nabla u\cdot\nabla v$ in $J'(u)(v)$ by parts using the following substitutions

$$X = \nabla u \qquad dY = \nabla v$$

$$dX = \Delta u \qquad Y = v.$$

Using the formula $\int XdY = XY - \int YdX$ we get

$$J'(u)(v) = \int_{\partial\Omega} \frac{\partial u}{\partial\eta}v - \int_\Omega \{\Delta u\cdot v + f(u)v\}.$$

Now, since v satisfies the boundary conditions the first term becomes zero, leaving the following

$$J'(u)(v) = \int_\Omega -(\Delta u + f(u))v.$$

Finally, if $J'(u)(v) = 0 \ \forall v \in H$ then

$$\Delta u + f(u) = 0.$$

Thus, the critical points of the action functional are solutions to the BVP. It is easily shown that if u is a solution of the BVP, then u is also a critical point of the action functional.

3.2 Orthonormal Basis

As we have discussed earlier, the eigenvalues of $-\Delta$ on the disk are determined by the zeroes of the various Bessel functions of the first kind. The corresponding eigenfunctions, which form an orthonormal basis, are divided into three components. The radial component is defined by

$$\psi_i = a_i J_0(\sqrt{\lambda_i^0}\ r),$$

where J_0 is the Bessel Zero function, $\sqrt{\lambda_i^0}$ is the i^{th} zero of $J_0(r)$, and $a_i = \frac{1}{\sqrt{\int_\Omega J_0(\sqrt{\lambda_i^0}\ r)^2}}$. That is to say, a_i, is calculated so that $\int \psi_i^2 = 1$.

The remaining two components of the orthonormal basis are the nonradial components and they are given by

$$\varphi_{i,j} = b_{i,j} J_i(\sqrt{\lambda_j^i}\ r)\cos(i\theta)$$

and

$$\chi_{i,j} = b_{i,j} J_i(\sqrt{\lambda_j^i}\ r)\sin(i\theta).$$

Here J_i denotes the various Bessel functions for $i \neq 0$, $\sqrt{\lambda_j^i}$ is the j^{th} zero of the i^{th} Bessel function, and $b_{i,j}$ is calculated so that $\int \varphi_{i,j}^2 = 1$, and $\int \chi_{i,j}^2 = 1$. Thus, given $u \in L^2$ there exist constants $\{a_i\}$, $\{b_{i,j}\}$, and $\{c_{i,j}\}$ so that

$$u = \Sigma_i a_i \psi_i + \Sigma_i \Sigma_j b_{i,j} \varphi_{i,j} + \Sigma_i \Sigma_j c_{i,j} \chi_{i,j}.$$

We calculated the first ten zeroes of the first ten Bessel functions of the first kind, starting with the Bessel zero function, $J_0(r)$. The zeroes were calculated using *Mathematica*. The normalizing factors, a_i and $b_{i,j}$, were also calculated using *Mathematica*. These values were then hardcoded into our program (see Appendix C).

NOTE: We started coding using *Mathematica*, but we needed faster execution to handle our algorithm, necessitating a change to *FORTRAN*. In *FORTRAN*, we have access to the *NETLIB* libraries: *LAPACK*, *BLAS*, and *NAPACK*.

The eigenfunctions of $-\Delta$ on the disk play a major role in our study of PDE (1.1), as well as in the numerical algorithm. In the next section we show how these eigenfunctions are essential in obtaining the gradient vector and Hessian matrix.

Once the orthonormal basis is generated, the matter of implementing the singly indexed and doubly indexed components in our *FORTRAN* program arose. The radial components of the orthonormal basis are singly indexed while the two non-radial components are doubly indexed.

The number of the orthonormal basis components used is called on the number of *Fourier modes*. This is the finite number of *Fourier* coefficient used in the expansion performed on the eigenfunctions. The number of "modes" is dependent on the number of Bessel functions s_2, and Bessel zeroes s_1, used in the code. The number of modes m is given by $m = s_1 + 2s_1(s_2 - 1)$. Once the number of modes is set, then the orthonormal basis components are divided into three categories: The radial components is stored as a vector of length $s1$, the non-radial component containing the $\cos(i\theta)$ term is stored in a vector of size $s_1(s_2 - 1)$, and the non-radial component containing the $\sin(i\theta)$ term is stored in a vector of size $s_1(s_2 - 1)$.

The orthonormal basis allows us to perform a *Fourier* expansion on the eigenfunctions of $-\Delta$ on the disk. The *Fourier* expansion is then used to approximate both radial and non-radial functions. Figure 3.1 shows how accurate these approximations are when we used the orthonormal basis to approximate the radially symmetric function $f = r^2(1 - r^2)$, on $B_0(1)$, with $s_1 = s_2 = 3$.

The independent axis represents the radius, $r \in [0, 1]$, and the dependent axis represent the amplitude of the two functions. The function we approximated starts at the point $(0, 0)$ and ends at the point $(1, 0)$.

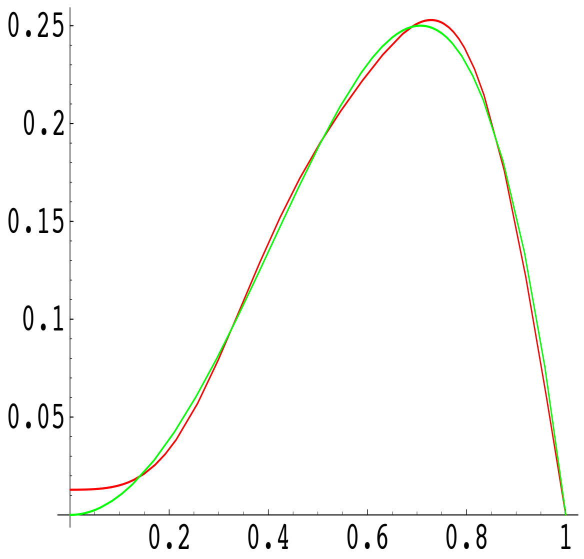

Figure 3.1: Orthonormal basis approximations

3.3 Gradient and Hessian

By regularity theory of elliptic BVP (see [10]), the critical points of the gradient are precisely the solution to (1.1). By the Riesz Representation Theorem (see [15]) one defines the L^2 gradient g to be the unique function which satisfies $J'(u)(v) =< g, v >_2$ for all $v \in L^2$. The subscript "2" will be dropped as we are only using the L^2 inner product, norm, gradient and Hessian. The reader is encouraged to read [14] for a more detailed discussion on the reason why we choose this gradient.

Our approximations will be restricted to a finite dimensional subspace K of L^2 of dimension $m = s_1+2s_1(s_2-1)$ containing functions which are twice differentiable for all $u \in K$. The subspace K is defined by $K = \text{span}\{\Psi_i\}_{i=1}^m$, where Ψ_i are the eigenfunctions discussed in previous sections, and m is the size of the finite dimensional subspace K.

Since the eigenfunctions of $-\Delta$ on the disk are normalized in L^2, we can define $\hat{J} : \mathbb{R}^m \longrightarrow \mathbb{R}$ by $\hat{J}(a) = J(u)$, where $u = \Sigma_{i=1}^m a_i \Psi_i \in K$. We are now seeking critical points of $\hat{J}$ instead of the infinite dimensional functional J.

Now, we can define the L^2 gradient of $\hat{J}$ to be the vector

$$g(a) = \left(\frac{\partial \hat{J}(a)}{\partial a_1}, \dots , \frac{\partial \hat{J}(a)}{\partial a_m} \right) = (J'(u)(\Psi_1), \dots , J'(u)(\Psi_m)) \in \mathbb{R}^m.$$

Integrating by parts yields

$$g_k(a) = \frac{\partial \hat{J}(a)}{\partial a_k} = - \int_\Omega (\Delta u + f(u))\Psi_k = a_k \lambda_k - \int_\Omega f(u)\Psi_k.$$

The self-adjoint Hessian $D^2 J(u)$ which represents the bilinear operator $J''(u)$, satisfies

$$J''(u)(v, w) =< D^2 J(u)v, w > .$$

Note that the subscript of 2 was left out. The Hessian matrix of $\hat{J}$ is given by

$$H(a) = \left(\frac{\partial^2 \hat{J}(a)}{\partial a_i \partial a_j} \right)_{i,j=1}^m = (J''(u)(\Psi_i, \Psi_j))_{i,j=1}^m.$$

Again, integrating by parts we obtain

$$H_{i,j}(a) = \delta_{ij} \lambda_i - \int_\Omega f'(u)\Psi_i \Psi_j,$$

where $\delta_{ij} = 1$ if $i = j$, otherwise $\delta_{ij} = 0$.

Since the Hessian matrix may be non-invertible we considered the use of singular value decomposition and pseudoinverses as tools in our algorithm (see Appendix B). In the case where H is invertable, the pseudoinverse is not needed, since it is the same as the regular inverse.

3.4 Newton's Method in Higher Dimensions

In Chapter 1, we derived the formula for Newton's method in one dimension. It is now necessary to extend that formula to higher dimensions in order for us to be able to use it for our problem.

Since we are interested in obtaining zeroes of the gradient g we use the gradient and Hessian developed in the previous section instead of f, and f' in (1.3). We now state the following formula for Newton's method in higher dimensions

$$a^{n+1} = a^n - H^{-1}(a^n)g(a^n),$$

where g represents $J'(u)(v)$, and H^{-1} represents either the inverse matrix of the Hessian, if H is invertible, or the pseudoinverse matrix $H^{\dagger}$, if H is noninvertible.

3.5 Numerical Algorithm

Now we have all the components needed for the numerical algorithm. We divide Ω into grid points (r_i, θ_i). The number of grid points has to be large enough in order for the code to produce satisfactory result. Our results were obtained using $30 - 50$ grid points. Since we are applying Newton's method we will need an initial guess vector for the *Fourier* coefficients. This initial guess is important to the convergence of Newton's method.

We truncated our infinite dimensional space H into a finite dimensional subspace K with dimension $m = s_1 + 2s_1(s_2 - 1)$. We will restrict all the code computations to the size of K.

Recall that the eigenfunctions were normalized in L^2 to satisfy $\int_\Omega \Psi_i \Psi_j = \delta_{ij}$ and $\int_\Omega \nabla\Psi_i \nabla\Psi_j = \delta_{ij}\lambda_i$, where δ_{ij} is the Kronecker delta function, and λ_i is the i^{th} eigenvalue of the Laplacian on the disk.

We define the finite m-dimensional function $\hat{J} : \mathbb{R}^m \longrightarrow \mathbb{R}$ by $\hat{J}(a) = J(u)$, where $u = \Sigma_{i=1}^m a_i \Psi_i \in K$. Thus we can now approximate the solutions to the BVP by computing the critical points of the $\hat{J}$.

With the gradient and Hessian now available, we have the following formula for Newton's iteration

$$a^{n+1} = a^n - H^{-1}(a^n)g(a^n).$$

One could also take smaller Newton's step iterations by introducing a step size δ to imitate a continuous Newton's flow. Then, Newton's iteration becomes:

$$a^{n+1} = a^n - \delta(H^{-1}(a^n))g(a^n).$$

When Newton's method generate a vector a, to which the sequence $\{a^n\}$ converges, we can approximate our solutions u by $u = \Sigma_{i=1}^m a_i \Psi_i$. If this solution is nondegenerate, then we can compute the *Morse* index of that critical point by counting the number of negative eigenvalues of the Hessian matrix.

PSEUDOCODE FOR ALGORITHM

We now provide a pseudocode for the algorithm, which can be used to approximate solutions in many regions and using many nonlinearities (see [14])

```
1. **Define** region $\Omega$, the nonlinearity $f$, and step size $\delta$.
2. **Obtain** orthonormal basis $\{\Psi_k\}_{k=1}^m$ for a large enough subspace $K \subset H$.
3. **Choose** initial guess $a = a^0 = \{a_k\}_{k=1}^m$, and set $u = u^0 = \Sigma_{i=1}^m a_i \psi_i$. Set the loop counter $n = 0$.
4. **Loop** until satisfied
    (a) Calculate the gradient vector of the action functional, $g = g^{n+1} = (J'(u)(\Psi_k)_{k=1}^m \in \mathbb{R}^m$.
    (b) Calculate the Hessian matrix, $H = H^{n+1} = (J''(u)(\Psi_j, \Psi_k))_{j,k=1}^m$.
    (c) Compute $W = W^{n+1} = H^{-1}g$, using the inverse or pseudoinverse.
    (d) Set $a = a^{n+1} = a^n - \delta W$. Update $u = u^{n+1} = \Sigma a_k \Psi_k$.
    (e) Increment loop counter $n$.
    (f) Calculate $\hat{J}(a)$, and the signature of $H$ if desired.
    (g) Calculate approximation $\sqrt{g \cdot g}$ of $\| \nabla J(u) \|$; STOP if small enough.
```

Chapter 4

NUMERICAL RESULTS

In this chapter we present the results we obtained from the algorithm developed in [14] applied to the disk instead of the square for the function $f(u) = \lambda u + u^3$.

4.1 Ordinary Differential Equation, ODE, Case

It is a good idea when studying a PDE problem, to first consider a simpler case. The ODE case, $n = 1$, is such a case. In this case, we are looking for solutions of the differential equation: $y'' + f(y) = 0$ on $[0, 1]$. By studying this case first, we were able to generate our first bifurcation diagram, (see Figure 4.1), and get a better understanding of what needs to be done for the PDE case. The bifurcation diagram was obtained by using a "shooting" argument (see [2]).

The shooting method can be used to solve the BVP

$$y'' = f(x, y, y'), \quad a \leq x \leq b, \quad y(a) = 0, \quad y(b) = 0,$$

by obtaining a sequence of solutions of the initial value problems, IVP, of the form

$$y'' = f(x, y, y'), \quad a \leq x \leq b, \quad y(a) = 0, \quad y'(a) = t,$$

involving a parameter t, to approximate the solutions to our BVP.

We start with an initial guess that determines the initial elevation at which the "object" is fired from the point $(a, 0)$, and follow the curve described by the IVP above. We solve the IVP using a *Euler's* numerical ODE solver, although more sophisticated solvers such as Runge-Kutta could be used. The code we used to solve the ODE case is found in Appendix D.

The bifurcation diagram obtained supports the result that the ODE case does indeed have infinitely many solutions. Clearly any vertical line drawn will intercept the graph infinitely many times. The system considered for the ODE case was $y'' = -(y^3 + \lambda y)$, $y(0) = 0 = y(\pi)$.

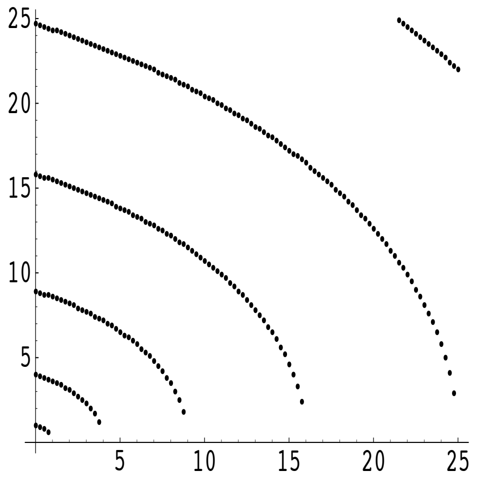

Figure 4.1: Bifurcation diagram for the ODE

The independent axis in the bifurcation diagram is the parameter λ and the dependent axis is the value of t which resulted in $y(\pi) = 0$.

After studying the ODE case, we now present some results obtained from executing the algorithm on the disk for the BVP. We need to mention here that there is a specific order associated with the eigenvalues of $-\Delta$ on the disk, and when we refer to a certain eigenfunction number we mean the eigenfunction corresponding to the eigenvalue with that number in the order of the eigenvalues of $-\Delta$ on the disk. For example, the third eigenfunction is the eigenfunction corresponding to the third smallest eigenvalue, and so on. The following table lists the first ten eigenvalues. Note: For $i \neq 0$ each eigenvalue has multiplicity two.

Number	Bessel Fcn (i)	Zero Number (j)	$\sqrt{\lambda_j^i}$
1	0	1	2.40483
2	1	1	3.83171
3	2	1	5.13562
4	3	1	5.38015
5	0	2	5.52008
6	1	2	7.01559
7	4	1	7.58834
8	2	2	8.41724
9	0	3	8.65373
10	5	1	8.77148

Table 4.1: First ten eigenvalues of $-\Delta$, on the disk.

4.2 One-Sign Solutions

The first solution obtained was the solution found in figure 4.2. This solution is positive on Ω, in other words, $u \geq 0$, for all $x \in \Omega$. This solution is also radially symmetric, that is $u_\theta = 0$ for all $(r, \theta) \in \Omega$. The positive solution corresponds to the first eigenfunction. It is obtained from an initial guess which includes non-zero entries in the radial component of our basis only.

The solutions obtained for this work were obtained after using different initial guesses for the *Fourier* coefficients in the algorithm. Each initial guess contained non-zero values in some of m available entries. The initial guess vector could be divided into three parts: the first s_1 entries would hold values for the coefficient of the radial component ψ_i of the basis, the next $s_1(s_2-1)$ entries would hold values for the coefficients of the non-radial component of the basis containing the $\cos(i\theta)$ terms φ_{ij}, and the last $s_1(s_2-1)$ entries would hold values for the coefficients of the non-radial component of the basis containing the $\sin(i\theta)$ terms χ_{ij}. For example, consider the case were $s_1 = s_2 = 3$, i.e., $m = 15$ modes. The initial guess would consist of 15 entries. The first three would be for the ψ_i component, the next six entries would be for the φ_{ij} component, and the last six entries would hold values for the χ_{ij} component. A "good" initial guess for the *Fourier* coefficients determines the speed of the convergence of our algorithm to a solution and more importantly, to which solution it converges.

The positive solution belongs to the first curve in the bifurcation diagram found in Figure 4.7. Table 4.2 lists the values for the solution coefficients when $\lambda = 0$ and $m = 15$.

λ	a_1	a_2	a_3	$a_{4\ldots15}$
0	2.6248	0.35973	0.049529	0

Table 4.2: positive solution coefficients at $\lambda = 0$.

Table 4.3 lists the coefficients of the positive solution on the positive solution branch, just before it bifurcates from the trivial solution branch.

λ	a_1	a_2	$a_{3\ldots15}$
5.75	0.23163	.00015298	0

Table 4.3: positive solution coefficients at $\lambda = 5.75$.

Note that the value of λ used in the Table 4.3 is very close to the 5.78 , the first eigenvalue of $-\Delta$ with zero Dirichlet boundary conditions on $B_0(1)$.

The negative solution is obtained by using the negative values for the initial guess used to obtained the positive solution. As expected, since our choice of f is odd, the negative solution has the same magnitude as the positive solution, and is also radially symmetric. Since the positive and negative solutions are radially symmetric then any rotation of these solutions will yield the same solution. We will discuss the case for the non-radially symmetric solutions later in this chapter.

4.3 Sign-Changing Solutions

The results obtained in section 4.2 were one-signed and radially symmetric. We want to show next that our algorithm can be used to find sign-changing solutions as well as non-radially symmetric solutions. This will show that our algorithm can be used to find the different types of solutions to the BVP:

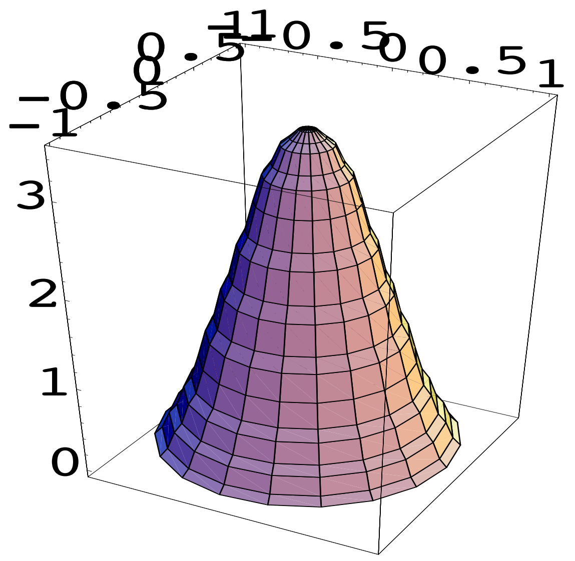

Figure 4.2: Positive solution

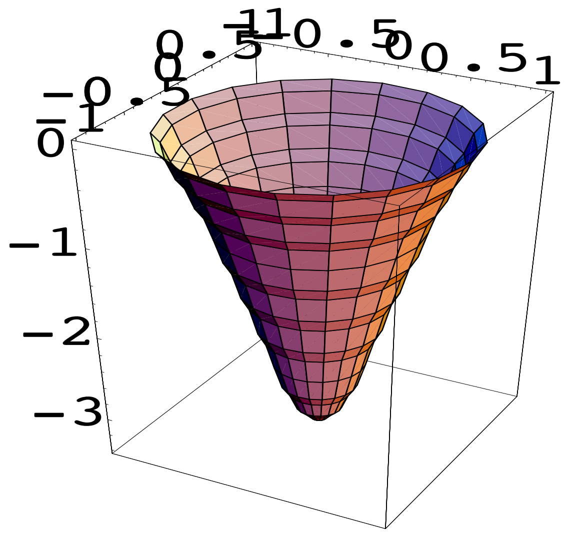

Figure 4.3: Negative solution

positive, negative and sign-changing, as well as radially symmetric and non-radially symmetric.

A sign-changing exactly once non-radially symmetric solution is obtained using an initial guess with non-zero entries in the φ_{ij} component of the basis. The sign-changing solution of lowest action value has been refered to as the *CCN* solution (see [4]). This solution corresponds to the second eigenfunction and belongs to the second curve in the bifurcation diagram. A listing of the solution coefficients is found in Table 4.4 for $\lambda = 0$.

λ	$a_{1\ldots3}$	a_4	a_5	a_6	$a_{7\ldots15}$
0	0	4.217408	0.4133865	-0.2021605	0

Table 4.4: coefficients for sign-changing, exactly-once, non-radially symmetric at $\lambda = 0$.

As the solution converges to the trivial solution the magnitude of the $\| u \|_\infty$ decreases. The coefficients of the *CCN* solution at $\lambda = 14.65$, before it converges to the trivial solution, are found in the following table

λ	$a_{1\ldots3}$	a_4	$a_{5\ldots15}$
14.65	0	0.2178988	0

Table 4.5: coefficients for sign-changing, exactly-once, non-radially symmetric at $\lambda = 14.65$.

We were also capable of obtaining a sign-changing non-radially symmetric solution which changes signs exactly twice. This solution corresponds to the third eigenfunction and is also obtained from an initial guess with non-zero entries in the ϕ_{ij} components. Tables 4.6 and 4.7 show the values of the solution coefficients at $\lambda = 0$ and $\lambda = 26$, respectively.

λ	$a_{1\ldots6}$	a_7	a_8	a_9	$a_{10\ldots15}$
0	0	5.68765	0.33685	-0.49658	0

Table 4.6: sign-changing twice solution coefficients at $\lambda = 0$.

λ	$a_{1\ldots6}$	a_7	a_8	a_9	$a_{10\ldots15}$
26.00	0	0.412829	0	-0.000154192	0

Table 4.7: sign-changing twice solution coefficients at $\lambda = 26$.

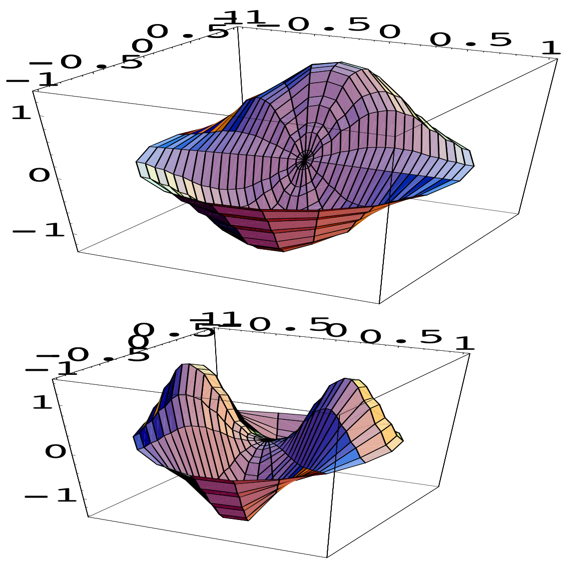

Figure 4.4: Sign-changing non-radially symmetric solutions

The existence of non-radially symmetric solutions lead us to the following question: What happens to a non-radially symmetric solution under a rotation?

To answer this question we considered the following experiment. Take the *CCN* solution which has non zero entries in the φ component of our orthonormal basis only. That is to say, $u_1 = \Sigma_i \Sigma_j b_{i,j} \varphi_{i,j}$. We observed that when we translated the non zero coefficients from the φ to the χ component of our basis, we obtained a solution u_2 which is a rotation by $\frac{\pi}{2}$ of u_1 of the form $u_2 = \Sigma_i \Sigma_j b_{i,j} \chi_{i,j}$.

Choosing various initial guesses of the form $\tilde{u}_t = \cos(t)u_1 + \sin(t)u_2$ leads to solutions u_t on the continuum of rotations of u_1. The solutions obtained in this manner are depicted in Figure 4.5

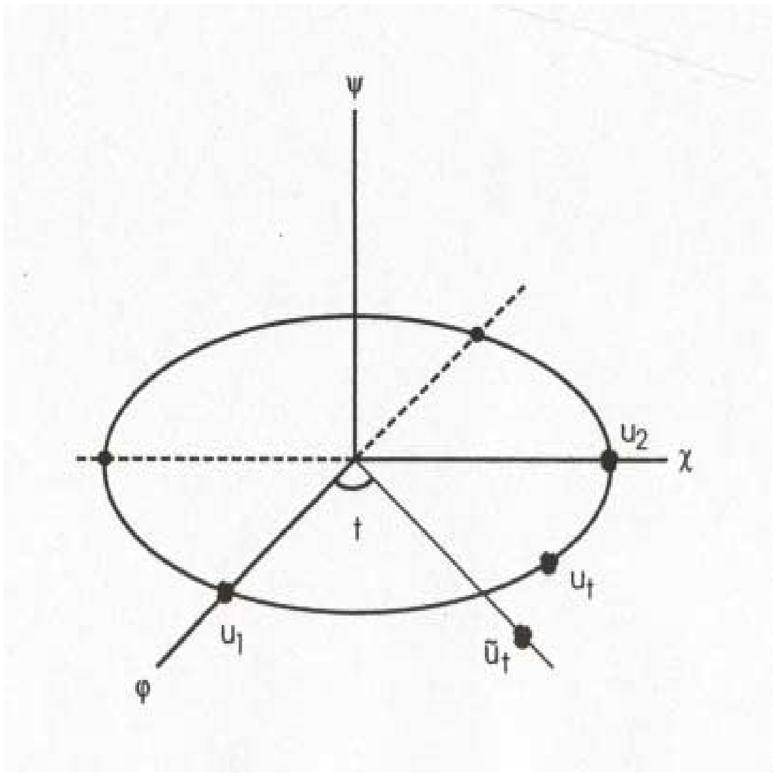

Figure 4.5: Rotation in the $\varphi\chi$-plane.

To summarize, if a solution is rotated by an angle θ then the rotation yields another solution, thus providing a continuum of solutions. That is to say, taking a non-radially symmetric solution and rotating it will yield another non-radially symmetric solution to BVP (1.1). The graphs in Figure 4.6 demonstrate this result. Note that this is also true for radially symmetric solutions since any rotation of a radially symmetric solution is equal to that prior to the rotation.

We also obtained a sign-changing radially symmetric solution. This solu-

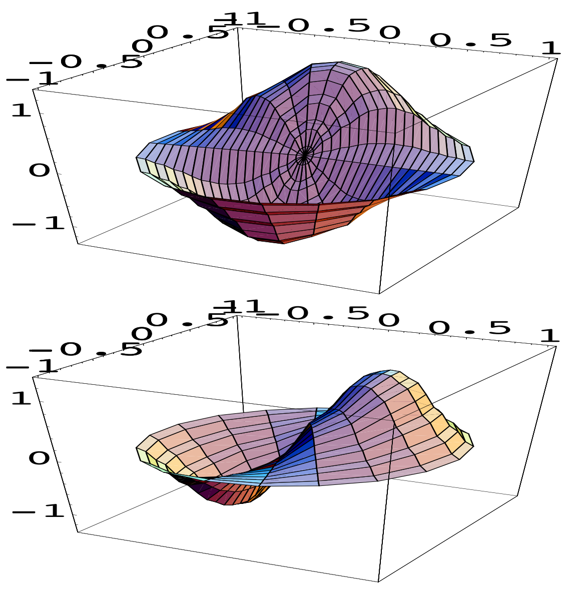

Figure 4.6: Rotation of non-radially symmetric solutions

tion was obatined with an initial guess containing non-zero entries for the ψ_i component of the basis. The solution corresponds to the fifth eigenfunction, which corresponds to the second zero of the Bessel Zero function. We are confident we could find many more solutions, although we would need to increase the number of modes in order to find these solutions. We have not yet attempted to construct the branch of solutions corresponding to the fourth eigenfunction. The values of the coefficients at $\lambda = 0$ are listed in Table 4.8

λ	a_1	a_2	a_3	$a_{4\ldots15}$
0	-2.7653657	5.902088	1.5634856	0

Table 4.8: coefficients of sign-changing, radially symmetric solution at $\lambda = 0$.

As this solution converged to the trivial solution, as one could observe from the fourth curve in Figure 4.7, the values of the solution coefficients decreased. Table 4.9 lists the values for the solution just before it converged to the trivial solution.

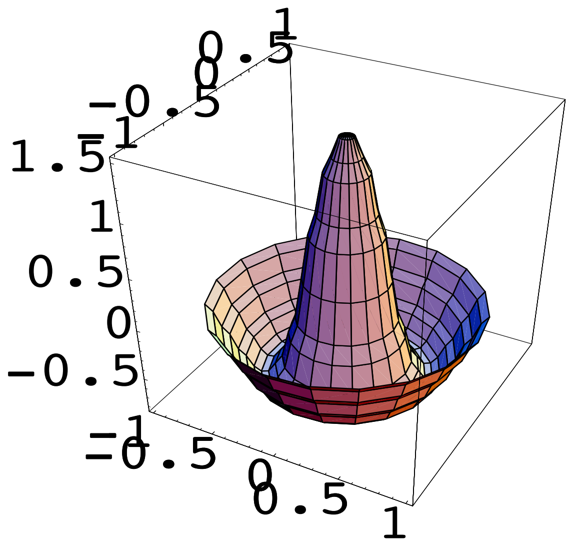

Figure 4.7: Sign-changing radially symmetric solution

λ	a_1	a_2	$a_{3\ldots15}$
30.45	0	0.166996	0

Table 4.9: coefficients of sign-changing, radially symmetric solution at $\lambda = 30.45$.

4.4 Bifurcation Diagram

With the assumption $f(0) = 0$, $u = 0$ is always a solution which is refered to as the trivial solution. Taking $f'(0) = \lambda$ as the bifurcation parameter, we will get a bifurcation of solutions from the trivial solution branch whenever λ passes through one of the eigenvalues of $-\Delta$. The bifurcations that might occur depend on the multiplicity of the eigenvalue, the parity of the eigenfunction, and the function f. The graph in Figure 4.7 shows a portion of the bifurcation diagram obtained for several solution curves of our BVP. The first curve corresponds to the positive/negative solutions, the second curve corresponds to the sign-changing exactly once non-radially symmetric solution of minimum action value, *CCN* solution, the third curve corresponds to the sign-changing twice solution and the last curve corresponds to the radially symmetric sign-changing solution.

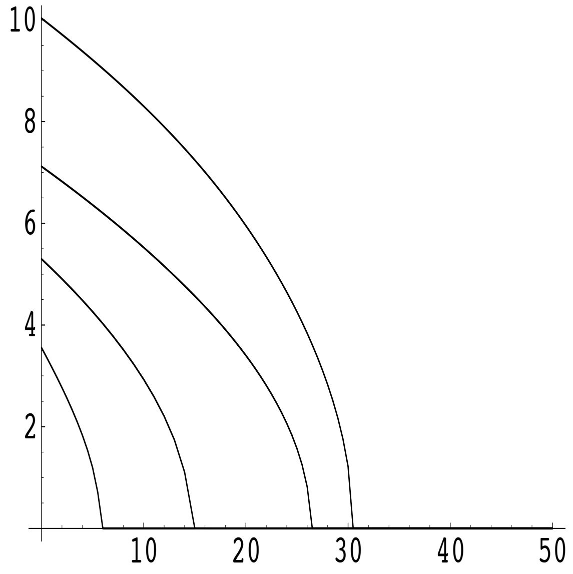

Figure 4.8: Bifurcation diagram for the PDE

The independent axis represent the value of λ at which the solution curve bifurcates from the trivial solution and the dependent axis represent the infinity norm of the solution, $\| u \|_\infty$. Note that each curve converges to the trivial solution. As expected, while the solution curves are converging to the trivial solution the magnitude of $\| u \|_\infty$ decreases. The value of

λ to which each curve converges should be the eigenvalue corresponding to the solution. For example, the last curve in Figure 4.7 corresponds to the fifth eigenfunction and bifurcates at the fifth eigenvalue, $\lambda_5 = (5.52008)^2 = 30.47128321$. Any point on a specific bifurcation curve represents a solution of the BVP which is the same type of solution (e.g., sign-changing non-radially symmetric). The amplitude of these solutions will always decrease as the branch approaches the trivial solution branch.

One interesting calculation one might consider is the signature of the Hessian matrix, that is, the number of negative eigenvalues in that matrix. The number of negative eigenvalues may be obtained by writing the Hessian matrix to a file. We used *Mathematica* to compute the eigenvalues of that matrix and count the number of negative eigenvalues. This is also known as the *Morse* index of a critical point if H is invertable. In the case were H is non-invertable, the critical points are degenerate and one cannot compute the *Morse* index. The *Morse* index of a critical point is important because by examining the signature of H along a bifurcation curve, we can determine whether we have secondary or higher bifurcation points on that curve. Along any bifurcation curve, if at a certain point the signature changes from one value to another, then that point produces a secondary bifurcation point. We did not observe any secondary or higher bifurcation points in our diagram, but we might observe them if we examine more bifurcation curves. We think that these secondary or higher bifurcation points will occur at solutions that have non-zero coefficients in more than one component of our orthonormal basis, since such a solution might cause a soluction curve corresponding to one eigenfunction to converge to a solution curve with a different corresponding eigenfunction.

Chapter 5

CONCLUSIONS AND FUTURE STUDIES

In this project, we approximated the solutions of a nonlinear elliptic BVP using Newton's method. An action functional was defined and Newton's method was applied to the gradient of that action functional. The results we obtained were compared to the results obtained by Neuberger, (see [13]), using a modified Mountain Pass Lemma technique. Our results were consistent with those obtained in the works mention in Chapter 2, and they indicate that our algorithm works for finding approximate solution to BVP (1.1) on the disk.

When we ran our code using a larger number of modes, i.e., a larger number of basis components, our results did not vary much. The bifurcation diagram and solution graphs were obtained using 66 modes and when compared to those obtained using only 15 modes, they were very similar. The code was tested with modes going as high as 190. The results remained very close regardless of the number of modes used. This gives confidence in the quality of the results obtained by our algorithm. Figure 5.1 shows the difference in $\parallel u \parallel_\infty$, as the number of modes increase. This graph was obtained for the positive solution at $\lambda = 0$. In the figure, the independent axis is the common value used for $s_1 = s_2$ instead of the total number of modes $m = s_1 + 2s_1(s_2 - 1)$.

Our results did not show any secondary bifurcation point on the first four computed bifurcation curves. We expect that secondary bifurcation points, or higher, might exist if we investigate more bifurcation curves. This is a change from the result obtained for the same BVP on the square, (see [14]).

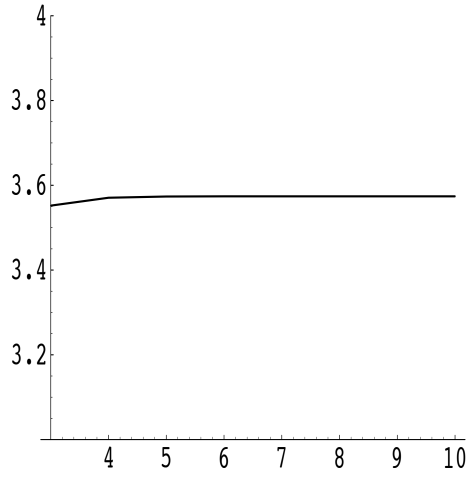

Figure 5.1: Infinity norm vs. number of modes

Their results showed a couple of bifurcation curves converging to the second eigenvalue, and they observed secondary and tertiary bifurcation points in their fourth bifurcation curve. We think this is due to the difference in nature between the eigenpairs of each region, and the fact that all of our solutions had non zero coefficients in only one component of the orhonormal basis.

Our results show that the method used in [14] can be used on the disk. This is good motivation to try and apply this algorithm for more complicated regions. This leads to another possible future study which would be to consider non uniform (arbitrary) regions. These regions would have no known closed form for the orthonormal basis, which would have to be approximated numerically.

We think it would be of value to consider the BVP on an annulus. This is because the annulus has Bessel functions as the orthonormal basis, and we can use our algorithm to approximate solutions on the annulus by considering it to be made up of many circles. The Bessel functions in that experiment would include both Bessel functions of the first kind $J_m(r)$ and $Y_m(r)$.

Another investigation to consider would be a triangle as the domain. We believe that the triangle might have an orthonormal basis in closed form. We have shown that the algorithm used for the square also works on the disk. It would be worth while to see whether this algorithm could also be applied to the triangle.

Bibliography

[1] A. Ambrosetti, and P. Rabinowitz. *Dual Variational Methods in Critical Point Theory and Applications.* Journal of Functional Analysis, 14, p 417-437, 1993.

[2] R.L. Burden and J.D. Faires. *Numerical Analysis.* PWS-Kent, 1989.

[3] A. Castro, and J. Cossio. *Multiple Solutions for a Nonlinear Dirichlet Problem.* SIAM Journal of Mathematical Analysis, Vol. 25, No. 6, p 1554-1561, 1994.

[4] A. Castro, J. Cossio, and J.M. Neuberger. *Sign-Changing Solutions of a Nonlinear Dirichlet Problem.* Rocky Mountain Journal of Mathematics, Vol. 27, No. 4, 1997.

[5] A. Castro, J. Cossio, and J.M. Neuberger. *On Multiple Solutions of a Nonlinear Dirichlet Problem.* Proceedings of the Second World Congress of Nonlinear Analysis, part 6 (Athens, 1996). Nonlinear Analysis Theory, Methods and Applications, Vol. 30, No. 6, p 3657-3662, 1997.

[6] A. Castro, J. Cossio, and J.M. Neuberger. *A Minimax Principle, Index of the Critical Point and Existence of Sign-Changing Solutions to Elliptic Boundary Value Problems.* Electronic Journal of Differential Equations, Vol. 1998, No. 2 p 1-18, 1998.

[7] R. Courant, and D. Hilbert. *Methods of Mathematical Physics, Volumes I and II.* New York: Interscience, 1953 and 1962.

[8] A. Castro, and A. Kurepa. *Infinitely many Radially Symmetric Solutions to a Superlinear Dirichlet Problem in a Ball.* Proceedings of The American Mathematical Society, Vol. 101, No. 1, 1987.

[9] Y.S. Choi, and P.J. McKenna. *A Mountain Pass Method for The Numerical Solution of Semilinear Elliptic Problems.* Nonlinear Analysis Theory, Methods and Applications, Vol. 20, No. 4, p 417-437, 1993.

[10] D. Gilbarg, and N. Trudinger. *Elliptic Partial Differential Equations of Second Order.* Springer-Verlag: Berlin, New York, 1983.

[11] B. Jacob. *Linear Algebra.* W.H. Freeman and Company, 1990.

[12] L. Ljusternik, and L. Schnirelmann. *Methods Topologique dans les Problems Variational.* Hermann and Cie, Paris, 1934.

[13] J.M. Neuberger. *A Numerical Method for Finiding Sign-Changing Solutions of Superlinear Dirichlet Problems.* Nonlinear World, Vol. 4, No. 1, p 73-83, 1997.

[14] J.M. Neuberger, and J.W. Swift. *Newton's Method and Morse Index for Semilinear Elliptic PDEs.* to appear.

[15] H.L. Royden *Real Analysis, Third Edition.* Prentice Hall, 1988.

[16] E. Zeidler *Applied Funtional Analysis, Applications to Mathematical Physics.* Springer, 1997.

[17] D. Zwillinger. *CRC Standard Mathematical Tables and Formulae,* 30th *Edition.* CRC Press, 1996.

Appendix A

Functional Analysis

In this section we look at some definitions and results from functional analysis which we used in this project, see [16].

Definition A.0.1 A Linear Space X over $\mathbb{R}$ is a set X, together with an addition

$$u + v \qquad\qquad u, v \in X$$

and a scalar multiplication

$$\alpha u \qquad\qquad \alpha \in \mathbb{R}, u \in X.$$

With the following holding true for all $u, v, w \in X$ and $\alpha,\ \beta \in \mathbb{R}$

$u + v = v + u$,
$(u + v) + w = u + (v + w)$

$(\alpha + \beta)u = \alpha u + \beta u$,
$\alpha(u + v) = \alpha u + \alpha v$

$\alpha(\beta u) = (\alpha\beta)u$,
$\alpha u = u$ if $\alpha = 1$.

Furthermore, there exists exactly one element $\theta \in X$ so that

$$u + \theta = u \qquad\qquad\qquad \forall u \in X.$$

And for each given $u \in X$, the equation

$$u + v = \theta,$$

has exactly one solution $v \in X$, $v = -u$.

Example A.0.2 Let $X := \mathbb{R}^N$, where $N = 1, 2, \cdots$; that is, the set of all the N-tuples

$$X = (\xi_1, \cdots, \xi_N)$$

Define

$$(\xi_1, \cdots, \xi_N) + (\eta_1, \cdots, \eta_N) = (\xi_1 + \eta_1, \cdots, \xi_N + \eta_N),$$

and

$$\alpha(\xi_1, \cdots, \xi_N) = (\alpha\xi_1, \cdots, \alpha\xi_N), \quad \alpha \in \mathbb{R}.$$

Then, X becomes a linear space over $\mathbb{R}$.

Definition A.0.3 Let X be a linear space over $\mathbb{R}$.

Then X is called a *normed space* over $\mathbb{R}$ iff there exist a norm $|| \cdot ||$ on X, i.e., for all $u, v \in X$ and α in $\mathbb{R}$ the following is true

(i) $|| u || \geq 0$, with equality iff $u = 0$.

(ii) $|| \alpha u || = | \alpha | \; || u ||$.

(iii) $|| u + v || \leq || u || + || v ||$.

Definition A.0.4 Let X be a normed space. For fixed $u_0 \in X$ and $\epsilon > 0$ the set:

$$U_\epsilon(u_0) := \{u \in X : \; || u - u_0 || < \epsilon\}$$

is called an ϵ-neighborhood of the point u_0.

The subset M of X is called *open* iff, for each point $u_0 \in M$, $\exists$ some ϵ-neighborhood, $U_\epsilon(u_0)$, such that

$$U_\epsilon(u_0) \subseteq M.$$

The subset M of X is called *closed* iff the set $X - M$ is open.

Definition A.0.5 Let M and Y be sets. An operator

$$A : M \longrightarrow Y$$

associates to each point u in M, a point v in Y, denoted by $v = Au$.

(i) $A : M \longrightarrow Y$ is called *surjective* iff $A(M) = Y$

(ii) $A : M \longrightarrow Y$ is called *injective* iff $Au = Av$ implies $u = v$

The operator $A : M \longrightarrow Y$ is *bijective* iff A is both surjective and injective.

If $A : M \longrightarrow Y$ is bijective, then there exists the so-called *inverse operator*

$$A^{-1} : Y \longrightarrow M$$

defined by:

$$A^{-1}v = u \quad \text{iff} \quad Au = v.$$

Example A.0.6 Let $M = \mathbb{R}$, and $Y = \mathbb{R}$.
Define the operator

$$A : \mathbb{R} \longrightarrow \mathbb{R}$$

by

$$Ax = x^3.$$

Definition A.0.7 Let X and Y be normed spaces over $\mathbb{R}$. The operator

$$A : M \subseteq X \longrightarrow Y$$

is called *continuous* iff, for each point $u \in M$ and each $\epsilon > 0$, there is a $\delta(\epsilon, u) > 0$ such that $|| \; v - u \; ||< \delta(\epsilon, u)$ and $v \in M$ implies $|| \; Av - Au \; ||< \epsilon$.

In addition, if it is possible to choose the $\delta(\epsilon, u) > 0$ in such a way that does not depend on the $u \in M$, then the operator $A : M \subseteq X \longrightarrow Y$ is *uniformly continuous.*

Definition A.0.8 The normed space X is called a *Banach space* iff each Cauchy sequence is convergent. Recall, a sequence $\{u_n\}$ in a normed space X is called *Cauchy* if for each $\epsilon > 0$ there exists a number $n_0(\epsilon)$ such that $|| \; u_n - u_m \; ||< \epsilon$ for all $n, m \geq n_0(\epsilon)$.
Banach spaces are also called *complete normed spaces.*

Example A.0.9 Let $N = 1, 2, \cdots$. Then the space $X := \mathbb{R}^N$ is a Banach space over $\mathbb{R}$ with the norm $|| \; x \; ||:=| \; x \; |_\infty$, where

$$| \; x \; |_\infty = max \; | \; \xi_j \; | \quad 1 \leq j \leq N \quad , \qquad\qquad x = (\xi_1, \cdots, \xi_N).$$

Definition A.0.10 Let X be a normed space. A subset M of X is called *dense* in X iff $\overline{M} = X$, i.e., for each $u \in X$ and each $\epsilon > 0$ there is a $v \in M$ such that $|| \; v - u \; ||< \epsilon$.

Definition A.0.11 Let X be an N-dimensional linear space over $\mathbb{R}$, where $N = 1, 2, \cdots$. By a basis $\{e_1, \cdots, e_N\}$ of X we mean a set of elements $e_1, \cdots, e_N$ of X such that, for each $u \in X$,

$$u = \alpha_1 e_1 + \cdots + \alpha_N e_N,$$

where $\alpha_1, \cdots, \alpha_N \in \mathbb{R}$ are uniquely determined by u, and are called the components of u.

Proposition A.0.12 *Let $N = 1, 2, \cdots$. Then in each N-dimensional linear space X over $\mathbb{R}$ there exists a basis $\{e_1, \cdots, e_N\}$.*

Definition A.0.13 The two norms $|| \cdot ||$ and $|| \cdot ||_1$ on the normed space X are called *equivalent* iff there are positive numbers α and β so that

$$\alpha \mid\mid u \mid\mid \leq \mid\mid u \mid\mid_1 \leq \beta \mid\mid u \mid\mid \qquad \forall u \in X.$$

Proposition A.0.14 *Two norms on a finite-dimensional linear space X over $\mathbb{R}$ are always equivalent.*

Proposition A.0.15 *Let $\{u_n\}$ be a sequence in a finite-dimensional normed space X with $dim(X) > 0$. then*

$$u_n \longrightarrow u \ \ in \ \ X \ \ as \ \ n \longrightarrow \infty$$

iff the corresponding components with respect to any fixed basis converge to each other.

Example A.0.16 Let $x_n = (\xi_{1n}, \cdots, \xi_{Nn})$. Then

$$\lim_{n \longrightarrow \infty} \mid x_n - x \mid = 0$$

iff

$$\lim_{n \longrightarrow \infty} \xi_{kn} = \xi_k, \quad \forall k = 1, \cdots, n.$$

That is, the convergence $x_n \longrightarrow x$ as $n \longrightarrow \infty$ is equivalent to the convergence of the corresponding components.

Definition A.0.17 A subset L of the linear space X over $\mathbb{R}$ is called a *linear subspace* of X iff

$$u, v \in L \ \ and \ \ \alpha, \beta \in \mathbb{R} \ \ imply \ \ \alpha u + \beta v \in L.$$

Definition A.0.18 Let X and Y be linear spaces over $\mathbb{R}$. The operator $A : L \subseteq X \longrightarrow Y$ is called *linear* iff L is a linear subspace of X and

$$A(\alpha u + \beta v) = \alpha Au + \beta Av \qquad for \ \ all \ \ u, v \in L \ \ and \ \ \alpha, \beta \in \mathbb{R}.$$

Definition A.0.19 Let X be a linear space over $\mathbb{R}$. An inner product on X assigns to each u and v in X a number $< u, v >\in \mathbb{R}$ such that the following hold for all $u, v, w \in X$ and $\alpha, \beta \in \mathbb{R}$

(i) $< u, u >\geq 0$ with equality iff $u = 0$.

(ii) $< u, \alpha v + \beta w >= \alpha < u, v > +\beta < u, w >$.

(iii) $\overline{< u, v >} =< v, u >$, where $\overline{u}$ represents the complex conjugate of u.

For $u, v \in X$, u is said to be *orthogonal* to v if $< u, v >= 0$.

A linear space X over $\mathbb{R}$ together with an inner product is called a *pre-Hilbert.*

Proposition A.0.20 *Each pre-Hilbert X space over $\mathbb{R}$ is also a normed space over $\mathbb{R}$ with respect to the norm*

$$\| u \| :=< u, u >^{\frac{1}{2}}, \quad \forall u \in X.$$

Definition A.0.21 A *Hilbert space* is a pre-Hilbert space that is a Banach space with respect to the norm

$$\| u \| :=< u, u >^{\frac{1}{2}}, \quad \forall u \in X.$$

Example A.0.22 Let $X := \mathbb{R}$. Then, X is a *real* Hilbert space with the inner product

$$< u, v >:= uv \qquad\qquad \forall u, v \in \mathbb{R}.$$

The corresponding norm $\| u \| =< u, u >^2$ equals $| u |$.

Proposition A.0.23 *Each finite-dimensional pre-Hilbert space is a Hilbert space.*

Definition A.0.24 Let X be a normed sapce over $\mathbb{R}$. By a *bounded bilinear* form on X we mean an operator $a : X \times X \longrightarrow \mathbb{R}$, satisfying

(i) Bilinearity:

For all $u, v, w \in X$ and $\alpha, \beta \in \mathbb{R}$

$$a(\alpha u + \beta v, w) = \alpha a(u, w) + \beta a(v, w).$$

and

$$a(w, \alpha u + \beta v) = \alpha a(w, u) + \beta a(w, v).$$

(ii) Boundedness:

There is a constant $d > 0$ so that

$$\mid a(u, v) \mid \leq d \mid\mid u \mid\mid \ \mid\mid v \mid\mid, \quad \forall u, v \in X.$$

$a(u, v)$ is called *symmetric* iff

$$a(u, v) = a(v, u), \quad \forall u, v \in X.$$

Definition A.0.25 Let X be a Hilbert space over $\mathbb{R}$. Then the finite, or countable, system $\{u_0, u_1, \cdots\}$ is called *orthogonal*, if

$$< u_k, u_m >= \delta_{km}, \quad \forall k, m.$$

Appendix B

Linear Algebra

In this section we look at some results and definitions from Linear Algebra, see [11].

Definition B.0.26 An $m \times n$ matrix $D = (d_{ij})$ is called a *pseudodiagonal* matrix if $d_{ij} = 0$ whenever $i \neq j$.

Definition B.0.27 An $m \times m$ matrix A is called *orthogonal* if $AA^t = I_m = A^tA$.

Definition B.0.28 Let A be an $m \times n$ matrix. We call the product $A = U_1DU_2^t$ a *singular-value decomposition* for A if U_1 is an $m \times m$ orthogonal matrix, D is an $m \times n$ pseudodiagonal matrix all of whose entries are non-negative, and U_2 is an $n \times n$ orthogonal matrix.

The diagonal entries of matrix D σ_i $\{i = 1, 2, 3, \ldots, n\}$ are called the *singular values* of A.

Theorem B.0.29 *Let A be an $m \times n$ real matrix. Then A has a singular-value decomposition. Moreover, the singular values of A are uniquely determined (with multiplicity).*

Definition B.0.30 Suppose that $< \cdot, \cdot >$ is an inner product on a vector space V. We say that a basis $\{u_1, \cdots, u_n\}$ is an orthogonal basis for V if the u_i are mutually orthogonal; that is, $< u_i, u_j >= 0$ whenever $i \neq j$. In addition, if each $|| u_i ||= 1$, we sat that the basis is *orthonormal.*

Theorem B.0.31 *Any finite-dimensional inner product space V, has an orthogonal basis (and hence an orthonormal basis).*

Example B.0.32 Let

$$A = \begin{pmatrix} 1 & 2 \\ 1 & 2 \\ 1 & 2 \end{pmatrix}$$

A has rank 1.

Form the symmetric matrix

$$S = A^t A = \begin{pmatrix} 3 & 6 \\ 6 & 12 \end{pmatrix}$$

Computing the eigenvalues of S we get that the chracteristic polynomial $C_s(X) = X(X - 15)$, which has $X = 0$, and $X = 15$ as eigenvalues. It can be shown that $X = 0$ has $(2, -1)$ as an eigenvector and $X = 15$ has an eigenvector $(1, 2)$.

It can be shown that $\{(\frac{1}{\sqrt{5}}, \frac{2}{\sqrt{5}}), (\frac{2}{\sqrt{5}}, \frac{-1}{\sqrt{5}})\}$ is an orthonormal basis in $\mathbb{R}^2$ of eigenvectors of S.

Next,

$$A \begin{pmatrix} \frac{1}{\sqrt{5}} \\ \frac{2}{\sqrt{5}} \end{pmatrix} = \begin{pmatrix} \sqrt{5} \\ \sqrt{5} \\ \sqrt{5} \end{pmatrix}$$

which has norm $\sqrt{15}$, the square root of the eigenvalue of the eigenvector $(\frac{1}{\sqrt{5}}, \frac{2}{\sqrt{5}})$ of S.

This vector is normalized to the unit vector $(\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}})$, which we extend to an orthonormal basis of $\mathbb{R}^3$, say
$\{(\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}), (\frac{1}{\sqrt{2}}, 0, \frac{-1}{\sqrt{2}}), \frac{1}{\sqrt{6}}, \frac{-2}{\sqrt{6}}, \frac{1}{\sqrt{6}})\}$.

So, this gives the following singular-value decomposition of A

$$\begin{pmatrix} 1 & 2 \\ 1 & 2 \\ 1 & 2 \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{6}} \\ \frac{1}{\sqrt{3}} & 0 & \frac{-2}{\sqrt{6}} \\ \frac{1}{\sqrt{3}} & \frac{-1}{\sqrt{2}} & \frac{1}{\sqrt{6}} \end{pmatrix} \begin{pmatrix} \sqrt{15} & 0 \\ 0 & 0 \\ 0 & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{5}} & \frac{2}{\sqrt{5}} \\ \frac{2}{\sqrt{5}} & \frac{-1}{\sqrt{5}} \end{pmatrix}.$$

Definition B.0.33 (i) If $D = (d_{ij})$ is an $m \times n$ pseudodiagonal matrix, the pseudoinverse of D, denoted $D^{\dagger}$, is defined to be the $n \times m$ matrix given by $D^{\dagger} = (f_{kl})$ where $f_{ii} = d_{ii}^{-1}$ if $d_{ii} \neq 0$, and $f_{kl} = 0$ otherwise.

(ii) If A is an $m \times n$ matrix with singular-value decomposition $A = U_1 D U_2^t$, we define the *pseudoinverse* of A, $A^{\dagger}$, to be the $n \times m$ matrix $A^{\dagger} = U_2 D^{\dagger} U_1^t$. The pseudoinverse is also known as the *Moore-Penrose generalized inverse.*

Example B.0.34 The pseudoinverse of matrix A, of example (1.4.7), is given by

$$A^{\dagger} = \begin{pmatrix} \frac{1}{\sqrt{5}} & \frac{2}{\sqrt{5}} \\ \frac{2}{\sqrt{5}} & \frac{-1}{\sqrt{5}} \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{15}} & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \\ \frac{1}{\sqrt{2}} & 0 & \frac{-1}{\sqrt{2}} \\ \frac{1}{\sqrt{6}} & \frac{-2}{\sqrt{6}} & \frac{1}{\sqrt{6}} \end{pmatrix}.$$

REMARKS

(i) The matrix we will use in this project, the Hessian, is a symmetric, square matrix.

(ii) When A is a sqaure symmetric matrix, then $\sigma_i = \mid \lambda_i \mid$, where λ_i represents the eigenvalues of matrix A.

(iii) The pseudo-inverse of matrix A, is used to find the least norm squared solution to $Ax = b$, when A is singular (noninvertible).

Appendix C

FORTRAN Code: Main Program

```
      program main

c
c This program approximates solutions of
c the Laplacian on the disk.
c

      integer   s1,  s2,  t ! # of modes, and
      integer   ss1, ss2, tt ! grid points
      integer   i,   j,   m, k ! counters
      integer   irank ! used in dgelss
      integer   tm    ! size of work
      integer   info ! used in dgelss
      integer   maxits          ! for convergence

      parameter (s1=3, s2=3, t=30)

      real*8    rc     !used in dgelss
      real*8    work(10*(s1+2*s1*(s2-1))) !dgelss
      real*8    s(s1+2*s1*(s2-1)) !dgelss

      real*8    ar(0:s2-1,s1,0:t) !populates basis
      real*8    g(s1+2*s1*(s2-1)) !gradient vector
      real*8    w(s1+2*s1*(s2-1))

c declare the Hessian matrix
      real*8    h(s1+2*s1*(s2-1), s1+2*s1*(s2-1))
      real*8    hs(s1+2*s1*(s2-1), s1+2*s1*(s2-1))

c norm and z are defined below
      real*8    norm(10, 0:9), z(10, 0:9)

      real*8    u(0:t,0:t),    v(0:t, 0:t)
      real*8    a(s1+2*s1*(s2-1))
      real*8    tol /.0010d0/
      real*8    step /0.25d0/, gnorm, svtol /-.010/
      real*8    sup, lamb
      real*8    inclam /0.250d0/ !increment of lambda
      real*8    beglam /0.0d0/, endlam /26.50d0/  !lambda loop boundaries

      common /dim/ ss1, ss2, tt, m
      common /lam/ lamb
c
c norm contains the normalizing factors of our
c orthonormal basis...it should be mentioned
c that norm reads the column number followed
c by the row number.  EX. norm(3,2)=.097943...
c
      data norm
     1  /0.8467035918146336, 0.363734032801459,
     1   0.2314924993891328, 0.169764042975790,
     1   0.1340248319919783, 0.110716333424303,
     1   0.0943139732886407, 0.0821444530321886,
     1   0.0727565,          0.0652943,
     2   0.2548069316531115, 0.141480784643807,
     2   0.0979431182186440, 0.074896873774668,
     2   0.0606306053654783, 0.050929645210038,
     2   0.0439048421817592, 0.0386916,
     2   0.0344163,          0.0310582,
     3   0.1812303981518046, 0.1156868087183309,
     3   0.0848698241788109, 0.0670087279938525,
     3   0.0553567583605326, 0.04715635764711681,
     3   0.0410719121321879, 0.0365114,
     3   0.0326505,          0.0216156,
     4   0.1397490018655017, 0.09771725696665015,
     4   0.0748417518750472, 0.06061205236292317,
     4   0.0509220230950026, 0.04390125238130618,
     4   0.0385811643092523, 0.0345026,
     4   0.0464232,          0.025683,
     5   0.1131271749192205, 0.0844561443625315,
     5   0.0668945410268326, 0.05531508095617517,
     5   0.0471382190756436, 0.04106299210306602,
     5   0.0363732917388703, 0.0327873,
     5   0.0345388,          0.0269494,
     6   0.09462155496668334,0.07425908879717315,
     6   0.06043552404622364,0.05085333508162501,
     6   0.04386992118119299,0.03856519103908102,
     6   0.03440199344895086,0.0311231,
     6   0.0287105,          0.0220264,
     7   0.08103574680160013,0.06617211487906287,
     7   0.05507950477521234,0.04704165096854257,
     7   0.04101718369144361,0.0363492109859799,
     7   0.03263062423864986,0.0297334,
     7   0.0271487,          0.0147849,
     8   0.0709218,0.0595902,0.056356,  0.0439807, 0.0384854,
     8   0.0438014,0.0310238,0.0283449, 0.0167437, 0.0213025,
     9   0.0625654,0.0615768,0.0536715, 0.0154881, 0.0289892,
     9   0.0486754,0.0295739,0.0271116, 0.0249543, 0.0245545,
     1   0.0559043,0.0495773,0.0502785, 0.0383306, 0.0287506,
     1   0.042403, 0.0282506,0.0205418, 0.0193991, 0.0223104/

c Z contains the zeros of the Bessel functions.
c It also reads the collumn number first and
c then the row number...z(3,2) is the third
c zero of the BesselJ(2)???????????????????????


      data z
     1        /2.40483, 5.52008, 8.65373, 11.7915, 14.9309, 18.0711,
     1         21.2116, 24.3525, 27.4935, 30.6346,
     2         3.83171, 7.01559, 10.1735, 13.3237, 16.4706, 19.6159,
     2         22.7601, 25.9037, 29.0468, 32.1897,
     3         5.13562, 8.41724, 11.6198, 14.796,  17.9598, 21.117,
     3         24.2701, 27.4206, 30.5692, 33.7165,
     4         6.38016, 9.76102, 13.0152, 16.2235, 19.4094, 22.5827,
     4         25.7482, 28.9084, 32.0649, 35.2187,
     5         7.58834243450397, 11.0647094886081, 14.3725366716403,
     5         17.615966049812,  20.8269329569653, 24.0190195247725,
     5         27.199087765982,  30.3710076671176, 33.5371377118195,
     5         36.6990011287448,
     6         8.77148, 12.3386, 15.7002, 18.9801, 22.2178, 25.4303,
     6         28.6266, 31.8117, 34.9888, 38.1599,
     7         9.93611, 13.5893, 17.0038, 20.3208, 23.5861, 26.8202,
     7         30.0337, 33.233,  36.422,  39.6032,
     8         11.0864, 14.8213, 18.2876, 21.6415, 24.9349, 28.1912,
     8         31.4228, 34.6371, 37.8387, 41.0308,
     9         12.2251, 16.0378, 19.5545, 22.9452, 26.2668, 29.5457,
     9         32.7958, 36.0256, 39.2404, 42.4439,
     1         13.3543, 17.2412, 20.807,  24.2339, 27.5837, 30.8854,
     1         34.1544, 37.4001, 40.6286, 43.8438/


      rc     = 1.0d-5 !dgelss

c for dim common block

      ss1 = s1
      ss2 = s2
      tt  = t
      m   = s1+2*s1*(s2-1)
      tm  = 10 * m

      maxits = 500

      write(*,*) m, " modes"
```

```
c***************************
c BEGIN MAIN BLOCK          *
c***************************

c populate the initial guess coefficients
c zero out u prior to initializing interior

      do i=0, t
         do j=0, t
            u(i,j) = 0.0d0
         end do
      end do

      do i=1, m
         a(i) = 0.0d0
      end do

c The following is a list of initial guesses
c which yielded solution to the BVP with
c s1=s2=3 and t=30

c positive sol at lam = 0.
c       a(1) = 2.62d0
c       a(2) = 0.36d0

c s-c e-o ccn J_1 psi_2c (slam_1 r) Cos... at lambda=0.
c       a(4) = 4.20d0
c       a(5) = 0.4d0

c s-c e-o J_1 psi_2c (slam_1 r) Cos ... at lam = 0
c with s1 = s2 = 6

c       a(7) = 4.20d0
c       a(8) = 0.4d0

c looking for psi_3c sol (J_2 (slam_1 r) Cos... at lambda = 0
        a(7) = 5.69d0
        a(8) = 0.34d0
```

```
c psi_2s
c       a(10) = 4.2d0
c       a(11) = 0.4d0

c p2c + p2s
c       do i=0,m
c          a(i) = .707 * a(i)
c       end do

c lam = 14, p2c
c       a(4) = 0.960d0

c  s-c r-s solution
c      a(2) = 7.0d0

c calling bsub to populate the  basis array
c used by psi, phi, and xhi in ppc.

c z and norm (i,j): i=1,s1 is zero #, j=0,s2-1 is bes #

      call bsub(ar, norm, z)

      open(unit=3, file="uj.txt")
      open(unit=2, file="bifj.txt")
      open(unit=4, file="hess.txt")
      open(unit=5, file="coff.txt")

c calculating the Fourier expansion of our basis

      call fourier(a, u, ar,sup)
      write(3,*) u



c
c Calculating the grad and hess and using
c Newton's method to get soln approx.
c
```

```
      do lamb = beglam, endlam,  inclam

         if (lamb .lt. endlam) then
             step = 0.95d0
         end if

         i = 0
         gnorm = 1.0d0

         do while ((i .lt. maxits) .and. (gnorm .gt. tol))

            i = i+1

c           write(*,*) "i=", i

c           write(*,*) "calling grad"
            call grad(a, u, g, z, v, gnorm, ar)
c           write(*,*) "G="
c           write(*,*) g

c           write(*,*) "calling hess"
            call hess(a, u, h, z, v, ar)
c     write(*,*) "H="
c     write(*,*) h

            do j=1,m
               do k=1, m
                  hs(j,k) = h(j,k)
               end do
            end do
c
c dgelss is a routine from the lapack library which
c finds the least norm squared solun to Ax=b
c

c           write(*,*) "calling dgelss"
            call dgelss(m,m,1,h,m,g,m,s,rc,irank,work,tm,info)
c     write(*,*) "output from dgelss="
```

```
c     write(*,*) g

c           write(*,*) "calling newt"
            call newt(a, g, w, step)
c     write(*,*) "output from newt"

            call fourier(a, u, ar, sup)

c           write(*,*) "Sup   = ", sup
    write(*,*) "gnorm = ", gnorm
c     write(3,*) u

c           write(*,10) (a(k), k=1,5)
c 10        format(5(f9.5,1x,\))


         end do
           write(*,*)"********************"
   write(*,*) "A="
           write(*,*) a(1),a(2),a(3),a(4)
           write(*,*) a(5),a(6),a(7),a(8)
           write(*,*) a(9),a(10),a(11),a(12)
   write(*,*) a(13),a(14),a(15)
         write(*,*) "********************"
         write(*,*) lamb, sup
         write(2,*) lamb, sup

         write(3,*) u
 write(4,*) hs
 write(5,*) a
      end do

      write(3,*) u

      close(5)
      close(4)
      close(3)
      close(2)
      end
c***************************
c END OF MAIN PROGRAM     *
c***************************

      real*8 function f(w)

c This function evaluates
c the cube of a number.

c This is the nonlinearity function

      real*8 w
      real*8 lamb
      common /lam/ lamb

      f =  w**3 + lamb*w

      end

c************************************

      real*8 function fp(w)

c this calculates the derivative of f(x)

      real*8 w
      real*8 lamb
      common /lam/ lamb

      fp =  3.0d0*w**2 + lamb

      end

c***********************************

      real*8 function integf(w)
```

```
c this is the primitive of f(w)

      real*8 w
      real*8 lamb
      common /lam/ lamb

      integf =  0.25d0*w**4 + (lamb/2.0d0)*w**2

      end

c*********************************

      integer function crondelta(i,j)

c this routine defines the Kronecker delta

      integer i, j

      if (i.eq.j) then
         crondelta = 1
      else
         crondelta = 0
      end if

      end

c*********************************

      real*8 function ppc(j, kr, kt, ar)

c this function eats j and spits out
c the appropriate basis element evaled
c at a point (kr,kt) in our grid.

      integer  j, j2, jb, jz
      integer  m, s1, s2, t
      integer  kr, kt
      real*8   ar(0:s2-1, s1, 0:t)
      real*8   psi, phi, xhi

      common /dim/ s1, s2, t, m

      if  (j.le.s1) then

         jb = 0
         jz = j
         ppc = psi(j,kr,ar)

      else

         if (j.le.s1+s1*(s2-1)) then

            j2 = j-s1
            jb = int((j2-0.5)/s1)+1
            jz = j2-(jb-1)*s1

            ppc = phi(jb,jz,kr,kt,ar)

         else

            j2 = j-s1-(s2-1)*s1
            jb = int((j2-0.5)/s1)+1
            jz = j2-(jb-1)*s1

            ppc = xhi(jb, jz, kr, kt, ar)

         end if

      end if

      end

c*****************************************************
```

```
      subroutine fourier(a, u, ar, sup)

c This routine calculates the Fourier expansion
c of our basis elements.

      real*8 a(s1+2*s1*(s2-1))
      real*8 u(0:t,0:t)
      real*8 ar(0:s2-1, s1, 0:t)
      real*8 sup

      real*8 sum, ppc

      integer i, m, s1, s2, t
      integer kr, kt

      common /dim/ s1, s2, t, m

c populating the interior of matrix u

      sup = 0.0d0
      do kr = 0, t-1
         do kt = 0, t

            sum = 0.0d0
            do i=1, m
               sum = sum + a(i) * ppc(i,kr,kt,ar)
            end do

            u(kr,kt) = sum
            if (abs(u(kr,kt)) .gt. sup) then
               sup = abs(u(kr,kt))
            end if

         end do
      end do

      end
```

```
c*******************************************************

      real*8 function lambda(j,z)

c This function eats an integer "j" and returns
c the appropriate bessel function and zero
c associated with that integer.

      integer j
      real*8  z(10, 0:9)

      integer s1, s2,t
      integer jb, jz

      common /dim/ s1, s2, t, m

      if (j.le.s1) then

         jb = 0
         jz = j

      else

         if (j.le.s1+s1*(s2-1)) then

            j2 = j-s1
            jb = int((j2-0.5)/s1)+1
            jz = j2-(jb-1)*s1

         else

            j2 = j-s1-(s2-1)*s1
            jb = int((j2-0.5)/s1)+1
            jz = j2-(jb-1)*s1

         end if
      end if
      lambda = z(jz, jb)

      end

c*****************************************************

      subroutine grad(a, u, g, z, v, gnorm, ar)

c Routine which calculates the grad
c of the action functional

c critical points of this grad are
c the approx solutions

      real*8  a(m)
      real*8  u(0:t, 0:t)
      real*8  g(m)
      real*8  z(10, 0:9)
      real*8  v(0:t, 0:t)
      real*8  gnorm
      real*8  ar(0:s2-1, s1, 0:t)

      integer i, kr, kt
      integer s1, s2, t, m

      real*8 integrate, f, ppc, lambda

      real*8 term

      common /dim/ s1, s2, t, m

c Begin calculating gradient

gnorm = 0.0d0
      do i=1, m

         do kr=0, t
            do kt=0, t
         v(kr, kt) = ppc(i,kr,kt,ar)*f(u(kr,kt))
            end do
         end do

         term = integrate(v)

         g(i) = a(i)*lambda(i,z)**2 - term

         gnorm = gnorm + g(i)**2

      end do

gnorm = gnorm ** .5

      end

c*************************************************

      subroutine hess(a, u, h, z, v, ar)

c This routine calculates the hessian matrix
c of the action functional

      real*8  a(m)
      real*8  u(0:t, 0:t)
      real*8  h(m,m)
      real*8  z(10, 0:9)
      real*8  v(0:t, 0:t)
      real*8  ar(0:s2-1, s1, 0:t)

      integer i, j, kr, kt

      real*8  integrate, fp, ppc, lambda
      integer crondelta

      real*8 term

      integer s1, s2, t, m
      common /dim/ s1, s2, t, m

c Begin calculating Hessian

      do i=1, m
         do j=1, m

            do kr=0, t
               do kt=0, t
               v(kr,kt) =
     1            ppc(i,kr,kt,ar)*ppc(j,kr,kt,ar)*fp(u(kr,kt))
               end do
            end do

            term = integrate(v)

            h(i,j) = lambda(i,z)**2.0*crondelta(i,j) - term


         end do
      end do

      end

c************************************************************

      subroutine newt(a, b, w, step)

c This routine implements Newton's
c method on the grad and hess "inverse"
c to calc critical points of the grad

      real*8  a(m)
      real*8  b(m) ! output of dgelss
      real*8  w(m)
      real*8  step
```

```
c     integer i, j
      integer k

      integer s1, s2, t, m

      common /dim/ s1, s2, t, m

c get the solution from dgelss

        do k=1,m
        a(k)=a(k)-step*b(k)

        end do

end
c****************************************************
subroutine bsub(ar, norm, z)

real*8  ar(0:s2-1,s1,0:t)
real*8  norm(10, 0:9), z(10, 0:9)

real*8  besselj, pi /3.141592653590d0/
real*8  a1, a2, a3, a4, a5, a55, a6

integer jb,jz,kr
integer s1, s2, t, m

common /dim/ s1, s2, t, m

c do jb=0,s2-1
c       do jz=1,s1
c        write(*,*) norm(jz, jb)
c       end do
c       end do

do jb = 0, s2-1
   do jz = 1, s1
              do kr = 0, t
```

```
                  a1 = z(jz,jb)
  a2 = float(kr)/t
  a3 = a1 * a2
  a4 = besselj(jb, a3)
  a5 = norm(jz,jb)
  a55 = sqrt(a5)
  a6 = a4/a55

          ar(jb, jz, kr) = a6

       end do
    end do
 end do

         end
c**********************************
      real*8 function besselj(n, x)

      integer n, nb, ncalc
      real*8  x,  b(100)
      real*8  alpha

nb    = n+1
alpha = 0.0d0

      call RJBESL(x, alpha, nb, b, ncalc)

      if (ncalc.ne.nb)then
       besselj = 999
      else
       besselj = b(nb)
      end if

      end
c*******************************************************
c* BesselJ(0, slam0[[j]]*r_k)
 *
c*******************************************************
```

```
      double precision function psi(j, k, ar)

integer ss1, ss2, tt
real*8  ar(0:ss2-1,ss1,0:tt)
common  /dim/ ss1, ss2, tt, m
integer j, k

psi = ar(0, j, k)

end
c*****************************************************
c* BesselJ(i, slam_i[[j]]*r_k)*cos(2 pi i theta)  *
c*****************************************************
      double precision function phi(jb, jz, kr, kt, ar)

integer jb, jz, kr, kt
real*8  ar(0:ss2-1,ss1,0:tt)

integer ss1, ss2, tt
common  /dim/ ss1, ss2, tt, m

real*8  pi /3.14159265359/, theta

theta = float(jb) * 2.0 * pi * float(kt) / float(tt)

phi = ar(jb, jz, kr) * cos(theta)

end
c*****************************************************
c* BesselJ(i, slam_i[[j]]*r_k)*sin(2 pi i theta)  *
c*****************************************************
      double precision function xhi(jb, jz, kr, kt, ar)

integer jb, jz, kr, kt
real*8  ar(0:ss2-1,ss1,0:tt)

integer ss1, ss2, tt
common  /dim/ ss1, ss2, tt, m
```

```
real*8  pi /3.14159265359/, theta

theta = float(jb) * 2.0 * pi * float(kt) / float(tt)

xhi = ar(jb, jz, kr) * sin(theta)

end
c*********************************
c integration                     *
c*********************************
real*8 function integrate(v)

integer  ss1, ss2, tt
common   /dim/ ss1, ss2, tt, m

integer  i, j, n
integer  ii, jj

real*8  r, dr, dt
real*8  uu, sum, rp1, rm1

real*8  v(0:tt, 0:tt)
real*8  pi /3.14159265359/

n=tt

dr = 1.0 / real(n)
dt = 2.0 * pi / real(n)

sum = 0.0

do i=1, n/2
   ii = 2*i - 1

   do j=1, n/2
      jj= 2*j - 1
```

```
           r   = real(ii)   * dr
      rp1 = real(ii+1) * dr
      rm1 = real(ii-1) * dr

      uu = (v(ii-1,jj-1) * rm1 + v(ii+1,jj+1) * rp1  +
     1                     v(ii+1,jj-1) * rp1 + v(ii-1,jj+1) * rm1) +
     2                    4 * (v(ii,jj-1) * r + v(ii,jj+1) * r      +
     3                        v(ii+1,jj) * rp1 + v(ii-1,jj) * rm1) +
     4                    16 * v(ii,jj) * r

                      sum = sum + uu
   end do
end do

             sum = sum * dr * dt / 9

     integrate = sum

end
c**********************************
```

Appendix D

Mathemactica Code: ODE bifurcation diagram

This is the code we used to generate the bifurcation diagram for the ODE case. We use Euler's method for higher order systems to solve our ODE problem.

```
Clear[a,b,x,y,n,i,alist,y0,yp0,yi1,yi2,f,dx];
Clear[difc,endc,c]; (* parameters for the initial value of the slope loop*)
Clear[diflambda,endlambda,lambda]; (* parameters for the lambda loop *)

f[y_,lambda_]:=-(lambda*y+y^3); (* system being solved *)

n         = 500; (* number of divisions for the interval [a,b] *)
a         = 0;
b         = Pi;
dx        = (b-a)/n;
difc      = .1;  (* increment of c.  c is the value of y'(a) *)
endc      = 25;  (* upper limit for c *)
diflambda = .25; (* increment of lambda *)
endlambda = 25;  (* upper limit for lambda *)
alist     = {};  (* creates a list *)

(* begin looking for values of lambda for which y(b) =0 *)
(* when one such value is found write it to alist as (lambda, c) *)
```

```
For[lambda=0, lambda<=endlambda, lambda+=diflambda,
For[c=difc, c<=endc, c+=difc,
yp0 = c;
yi2 = y0 //N;
yi1 = y0 +dx *yp0 //N;

For[i=2, i<=n, i++,
x    = a + i * dx;
y    = (dx^2)*(f[yi1,lambda]+2*yi1-yi2 //N;
temp = yi1;
yi1  = y;
yi2  = temp;
   ];

ysign = Sign[y];
If[c==difc, test = ysign, Null];
If[test==ysign,Null,
test = ysign;
Print["c=", c, " and lambda=", lambda];
alist = Append[alist, {lambda, c}];
];
];
];
ListPlot[alist];  (* graph the bifurcation diagram *)
```

Appendix E

Mathematica code: Solution plots

This code is used to plot the solutions from our *FORTRAN* program. After writing the solutions to a file in *FORTRAN* we read that file in *Mathematica* and obtain a parametric plot for the solution after changes the values in the file from rectangular to polar.

```
n     = 30; (* number of grid points *)
fmt   = Table[Table[Number, {i,0,n}],{j,0,n}]; (* formats the soluton file *)
alist = ReadList["file-path", fmt]; (* reads the contents of the file *)
c     = alist[[3]]; (* choose one set of solution coefficients *)


ii[r_,t_] := Floor[r*n]+1; (* generates a number between 1 and 31 *)
jj[r_,t_] := Floor[t*n/(2 Pi)] +1 (* same as above *)

f[r_,t_]  := c[[jj[r,t], ii[r,t]]];

ParametricPlot3D[{r Cos[t], r Sin[t], f[r,t]}, {r,0,1}, {t, 0, 2 Pi}];
```