The Critical Exponent Problem for Radially Symmetric Solutions to a Nonlinear PDE

Northern Arizona University REU Program 2012*

Jacob N. Clark♣ and Nathaniel Veldt♠

♣ Dept. of Mathematics and Computer Science, Wheaton College, Wheaton, IL
♠School of Mathematical and Statistical Sciences, Arizona State University, Tempe, AZ

July 20, 2012

*This paper details the results of work done at Northern Arizona University under the guidance of Dr. John M. Neuberger and Dr. James W. Swift. During our research we collaborated with Rachel Bachman of Clarkson University, and Robert Ravier of Cornell University, both of whom produced reports on similar topics. A large portion of Section 1 in this paper was the work of Rachel Bachman, whose main results were based on much of the same background material. The program ran from 5/27/12 to 7/20/12 and was funded by the National Science Foundation.

Abstract

In this paper we are concerned with the family of PDE $\Delta u + f(u) = 0$ on the the ball in $\mathbb{R}^3$ with zero Dirichlet boundary condition. We specifically consider this family of equations when given in spherical coordinates, and are only concerned with radially symmetric solutions. We are especially interested in the critical exponent case where $f(u) = su + u^5$. Our numerical methods are variational in nature, i.e., they rely on finding the critical points of an associated action functional. Our principal algorithm makes use of Newton's method to find critical points of our functional. We use a discrete set of normalized eigenfunctions of the Laplacian to serve as a basis for the Galerkin expansion of our approximate solutions. In our research we use and improve numerical techniques that were developed for solving the PDE on other regions and with different functions f. We address the spurious solutions that result from discretization for the critical exponent case. We then approximate the values of the bifurcation parameter s which mark the boundary between true and spurious solutions found by our algorithm for different branches of our bifurcation diagram.

1 Introduction

1.1 Overview of the PDE

[1] The primary concern of this work is to find solutions to non-linear elliptic partial differential equations, in particular, the system:

$$\begin{cases} \Delta u + f(u) = 0 \\ u = 0 \text{ on } \quad \partial\Omega \end{cases} \tag{1}$$

on a piecewise smooth and bounded region $\Omega \subset \mathbb{R}^N$. Here, Δ is the Laplacian operator, and we consider f to be a superlinear, critical function. In other words, there exists $B > 0$ such that $|f(u)| \leq B(|u|^p + 1)$ for $p = \frac{N+2}{N-2}$, thus f is critical. Also, f satisfies the criteria of "superlinear":

$$f'(u) > \frac{f(u)}{u} \text{ for } u \neq 0 \text{ and } \lim_{|u|\to\infty} \frac{f(u)}{u} = \infty.$$

To further define our problem, we review the concepts of a metric space, Newton's Method and the Secant method, and also define terms such as $action\ functional$, $Morse\ Index$ and $bifurcation$.

1.2 Important Concepts

In order to understand our solutions to the PDE, it is helpful to review the metric spaces where they lie. One particularly important metric space, called a *Banach space*, is complete normed vector space. A *vector space* is a set of vectors paired with the binary operations of addition and scalar multiplication such that the group axioms hold. Therefore, in a Banach space there exists a norm function that assigns a length in $\mathbb{R}^+$ to each vector, and length zero to the zero vector. Another example of such a space is a *Hilbert Space*, or a space that is complete under an inner product. Solutions to (1) more specifically lie in a subset of a Hilbert space called a *Sobolev space*, denoted H, with inner product $\langle u, v\rangle = \int_\Omega \nabla u \cdot \nabla v \, dx$.

[1]In this paper, we are assuming an audience with an undergraduate mathematics background, i.e., basic exposure to concepts of differential equations, linear algebra, and calculus.

The concept of an *action functional* is also very important in our research. In simple terms, a functional is a map from the vector space, in this case H, into the real numbers. It can be thought of as a “function on functions.” Here, the action functional we choose to study is $J : H \to \mathbb{R}$, defined by:

$$J(u) = \int_{\Omega} \left(\frac{|\nabla u|^2}{2} - F(u) \right) dx, \tag{2}$$

where ∇u is the vector gradient of u and $F(u)$ is the anti-derivative of f, from above, with $F(0) = 0$.

Another term to define is *Morse Index*. The Morse Index (MI) of J is the number of linearly independent directions v in the function space in which J “curves down”, i.e. $J''(u)(v, v) < 0$ [4]. Thus, a MI = 0 solution indicates a local minimum, and a MI = 1 solution indicates a saddle point. For our purposes, the Morse Index is the same as the *signature* of the Hessian matrix, which holds the second partial derivative terms of J. The signature of a matrix is the number of negative eigenvalues.

Now we discuss the subject of *bifurcation*. Bifurcation theory, in our case, studies changes in the structure of solutions of the partial differential equations in question. A bifurcation is said to occur when a small smooth change to a specific parameter causes a sudden behavioral change in the system. For our research, the bifurcation parameter is s, and the locations of bifurcations are based on the locations where the signature of the Hessian changes. To understand what happens when we change s, we plot bifurcation diagrams, with s serving as the independent variable (our x-axis), and a scalar in the y-axis called a schematic function, representing a solution to the PDE. We later discuss in further detail what schematic functions we use in the bifurcation diagrams to represent solutions u which lie in the infinitely-dimensional Sobolev space. A branch of a bifurcation diagram is a continuum of solutions in the schematic function vs. s plane. Each point on a specific branch corresponds to a solution to (1) for a given value of s and and a solution u. Branches of the bifurcation begin at eigenvalues. However each branch may have multiple bifurcations branching from it.

Lastly, we review the basic root finding algorithms of Newton’s method and the secant method. Given a real valued function in C^2, say $g : \mathbb{R} \to \mathbb{R}$, Newton’s method can approximate roots by generating a sequence from an initial guess, and the value of the function and its derivative at that guess. The sequence is generated iteratively as follows:

$$x_{n+1} = x_n - \tfrac{g(x_n)}{g'(x_n)}.$$

Iterations are continued until the value of x_n is within an indicated tolerance of a zero of g. Proof that Newton’s method converges to a solution can be found in the appendix.

Similarly, the secant method can also be used to approximate solutions. Here, a succession of secant lines are drawn between two function values, and the root is calculated respectively. It can be thought of as a finite difference approximation of Newton’s method. Instead, the sequence generated here begins with two initial guesses, as follows:

$$x_n = x_{n-1} - f(x_{n-1}) \tfrac{x_{n-1} - x_{n-2}}{f(x_{n-1}) - f(x_{n-2})}.$$

The drawbacks to these methods lies in the necessity of a good initial guess. Without this, neither method is guaranteed to converge (again see proof).

2 Preliminary Review

2.1 Galerkin Expansion

It is known that solutions to the PDE (1) lie in the Sobolev space H, a complete vector space that has an inner product for $u, v \in H$ s.t.

$$\langle u, v\rangle_H = \int_\Omega \nabla u \cdot \nabla v dx$$

with corresponding norm for measuring functions $u \in H$:

$$||u||_H^2 = \int_\Omega (\nabla u)^2 \, dx.$$

Note that $H \subset L_2$, where

$$L_2 = L_2(\Omega, \mathbb{R}) = \{u : \Omega \to \mathbb{R} \mid \int_\Omega u^2 < \infty\}.$$

Therefore, we also make use of the L_2 inner product and norm:

$$\langle u, v\rangle_2 = \int_\Omega uv \, dx,$$

$$||u||_2^2 = \int_\Omega (u)^2 \, dx.$$

We observe that $H = \text{span}\{\psi_i\}_{i\in\mathbb{N}}$, where ψ_i are an orthonormal set of eigenfunctions in L_2 of $-\Delta$ with corresponding eigenvalues λ_i. We have that:

$$-\Delta\psi_i = \lambda_i\psi_i \text{ on } \Omega.$$

Thus, $\langle\psi_i, \psi_j\rangle_2 = 0$ if $i \neq j$ and $\|\psi_i\|_2^2 = 1$.

We take the Galerkin space G to be the finite dimensional subspace of H, defined by:

$$G = \text{span}\{\psi_1, \psi_2, \ldots, \psi_{\mathrm{m}}\} \text{ where } \mathrm{m} \in \mathbb{Z}^+.$$

A Galerkin expansion changes the PDE from a continuous function space to a discrete space. Solutions to the PDE u are written as a truncated Fourier series:

$$u = \sum_{k=1}^{m} a_k\psi_k,$$

where a_k values are the Fourier coefficients of the expansion.

2.2 Variational Methods

Our methods for finding solutions to (1) are variational. Variational methods are mathematical techniques based on finding the critical points of functionals. As previously mentioned, the functional we are concerned with is

$$J(u) = \int_\Omega \left(\frac{|\nabla u|^2}{2} - F(u)\right) dx. \tag{3}$$

The first two directional derivatives of this functional are

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

and

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

This functional is important to us because it has been proven in [6] that u is a critical point of J if and only if it is a solution to the PDE. Therefore, we are searching for functions u such that $J'(u)(v) = 0$ for all $v \in H$.

We can plot slices of this functional to help us visualize where we are searching for solutions. The domain of J is an infinite-dimensional Hilbert space H, where solutions to (1) reside. This space is spanned by the eigenfunctions of the negative Laplacian. Since we are unable to plot in infinite dimensions, we must be content with visualizing a single two or three dimensional slice. For example, let $q : \mathbb{R} \to \mathbb{R}$ defined by:

$$q(t) = J(t\psi_1) = t^2 \int \frac{|\nabla \psi_1|^2}{2} dx - \frac{t^4}{4} \int \psi_1^4 dx,$$

and let $q(s,t) = J(s\psi_1 + t\psi_2)$. The graph of these functions are displayed below in Figures 1 and 2.

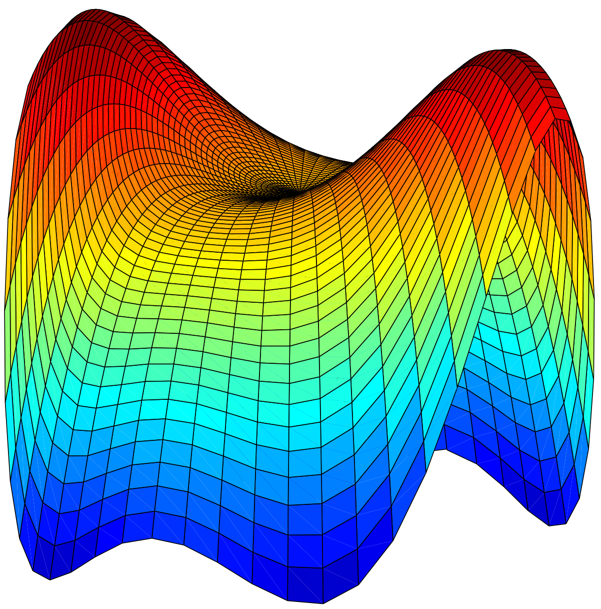


Figure 1: 3D slice of Functional J. We plot $q(s,t) = J(s\psi_1 + t\psi_2)$ using polar coordinates, i.e., $s = r\cos\theta$ and $t = r\sin\theta$ with $0 \le \theta \le 2\pi$, $0 \le r \le 6.5$

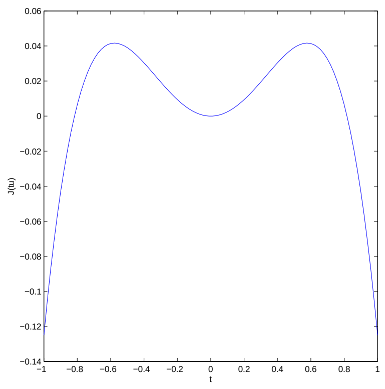


Figure 2: 2D slice of J

Note that the points at the top of the humps in Figure 2 and the valley at the origin are where the first derivative is zero, and thus are critical points. There are many specific properties of J that are of interest to us. For example, we can see in Figure 1 that $J''(0)(u,u) > 0$. Since we know $J'(0)(u) = 0$, we have that the origin is a local minimum. Also, for solutions $u \neq 0$, which are saddle points, we have $J''(u)(u,u) < 0$, indicating that solutions to the PDE are saddle points, as expected. We also observe that since our function f is superlinear, F is superquadratic and therefore larger than the gradient term. Thus $J(tu) \to -\infty$ as $t \to \infty$, so the minimum at the origin is unique.

3 The GNGA

3.1 Introduction to the Algorithm

The algorithm we propose was introduced by our REU mentors, Dr. Neuberger and Dr. Swift, and is known as the Gradient Newton Galerkin Algorithm (GNGA). The algorithm finds critical points in the gradient of the action functional of the PDE using Newton's method. Since we are concerned with a Hessian matrix filled with second partial derivatives, zeros indicate locations that are (local) maxima, minima, or saddle points, and thus solutions.

Algorithm 1 GNGA

```
Require: Region $\Omega$ and nonlinearity $f$
  Orthonormal basis $\{\psi_k\}_{k=1}^M$ for a sufficiently large subspace $G \subset H$
  Choose initial coefficients $a = a^0 = \{a_k\}_{k=1}^M$
  $u = u^0 = \Sigma a_k \psi_k$
  $j = 0$
  while $\|g\|_2 >$ tolerance $\vee$ $n >$ maxIterations do
      $g = g^{j+1} = (J'(u)(\psi_k))_{k=1}^M \in \mathbb{R}^M$ (Gradient vector)
      $A = A^{j+1} = (J''(u)(\psi_j, \psi_k))_{j,k=1}^M$ (Hessian matrix)
      $\chi = \chi^{j+1} = A^{-1}g$
      $a^{j+1} = a^j - \chi$
      $u^{j+1} = \sum_{k=1}^m a_k^{j+1}\psi_k$
      Calculate $sig(A(u))$ and $J(u)$ if desired.
      Calculate $\|g\|_2$, an approximation of $\|\nabla J(u)\|$.
      $j \leftarrow j + 1$
  end while
```

3.2 The tGNGA

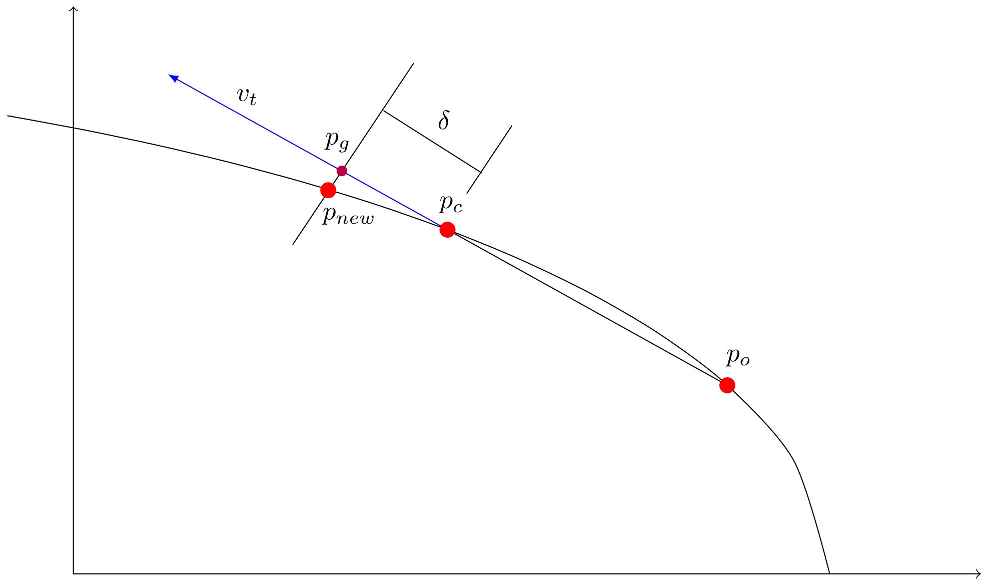


Figure 3: The tGNGA. Both p_c and p_o are known solutions, v_t is the vector in between them, p_g is the guess at a new solution and p_{new} is the new point fulfilling the solution constraints.

One problem with the GNGA is one must have a good initial guess to have convergence. In order to fulfill the condition of using good initial guesses, we implement what is called the Tangent Augmented Gradient Newton Galerkin Algorithm (tGNGA).

The tGNGA is a refinement of the GNGA using a continuation method. It takes two consecutive solutions on the branch, p_c, the current point and p_o, the previous point, and computes the approximate tangent vector in between the two points. Specifically,

$$v_t = \frac{p_c - p_o}{\|p_c - p_o\|}.$$

Using this approximate tangent vector, v_t, we calculate a guess at a new solution,

$$p_g = p_c + \delta v_t.$$

The tGNGA then attempts to find a solution using Newton's Method in much the same way as the GNGA. If it is successful, the tGNGA returns a new point, p_n, which is also a solution to (1) that is located on the hyperplane passing through the initial guess, p_g. This allows us to travel along a bifurcation branch, finding points in close proximity to each other to later plot and connect together for our bifurcation diagram. This algorithm is depicted in graphical form in Figure 3. The outline of the tGNGA is located below as Algorithm 2. For more detailed explanation of the tGNGA, we recommend [5].

```
Algorithm 2 tGNGA
Require: Two consecutive solutions on branch, $p_o$ and $p_c$
  $v_t = \frac{(p_c - p_o)}{\|p_c - p_o\|}$
  $p_{guess} = p_c + \delta v_t$
  Constraint: create hyperplane through $p_{guess}$ perpendicular to $v$
  Run Newton's Method on $p_{guess}$ subject to above constraint to find $p_{new}$ on branch
```

4 Preliminary Results on the Interval

First, we tested our code on an interval. While this is not anything particularly new, it was an interesting preliminary exercise.

The equation corresponding to Figures 4-6 is

$$u'' + su + u^3 = 0, \tag{4}$$

with the Dirchlet boundary conditions, $u(0) = 0$ and $u(\pi) = 0$. We used $n = 81$ grid points in our numerical integration, and $m = 40$ modes in our Galerkin expansion. The length of our interval, L, is π. Our speed on the branches is 0.1. The bifurcation points encountered are at $s = 1$ and $s = 4$. Our numerical approximations return what is expected on the interval, so now we turn to the PDE on the Ball.

5 PDE on the Ball in $\mathbb{R}^3$

As previously mentioned, we are motivated in our research to find solutions to the following partial differential equation with zero Dirichlet boundary conditions:

$$\begin{cases} \Delta u + f(u) = 0 & \text{on } \Omega \in \mathbb{R}^n \\ u = 0 & \text{on } \partial\Omega \end{cases} \tag{5}$$

While many variations of this equation have been studied, we specifically seek solutions to the PDE in spherical coordinates, where the region Ω is a ball of radius π in three-space and when we only consider solutions with radial symmetry. By considering only solutions with radial symmetry, we are assuming that solutions to the PDE are functions only in terms of r, and not in terms of the other two spherical coordinates ϕ and θ. Our solutions thus can be written as $u(r)$, rather than $u(r, \phi, \theta)$. Under these conditions, our equation simplifies to an ordinary differential equation:

$$u'' + \frac{2}{r}u' + f(u) = 0, \tag{6}$$

where r is the radius of the sphere, and $r \in (0, \pi)$.

We use the normalized eigenfunctions of the Laplacian in spherical coordinates to serve as a generalized Fourier series basis for calculating u. For this region, the normalized eigenfunctions have the form

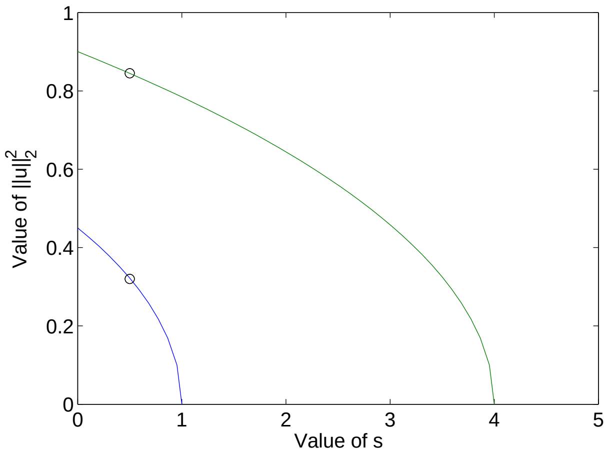


Figure 4: Bifurcation diagram for Dirichlet conditions, $m = 40$, $n = 2m + 1$, L_2 norm of solution plotted against s

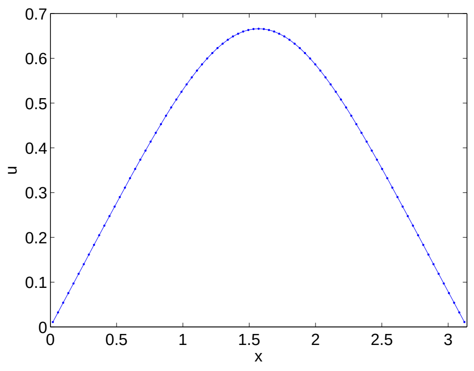


Figure 5: Solution, on first branch, located at dot.

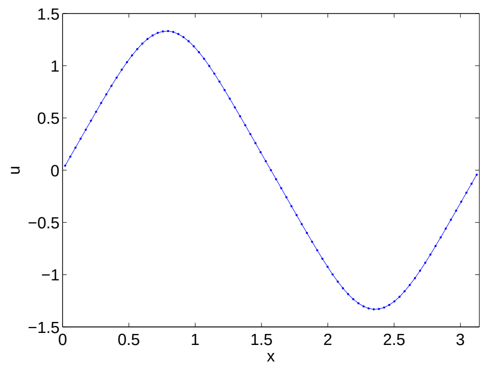


Figure 6: Solution, on second branch, located at dot.

$$\psi_k(r) = \frac{\sin(kr)}{\sqrt{2\pi}r} \text{ for } k \in \mathbb{Z}^+.$$

We detail the necessary calculations for these eigenfunctions and their corresponding eigenvalues in Appendix B.

We note the difference in numerical integration when Ω is a ball than when Ω is an interval. In our calculations on the interval, it can be shown that if our number of grid points n is greater than $2m$, our numerical integration is exact (see related calculations in Appendix D). On the ball, however, this is not the case. Our integration is never exact, but our approximation improves with higher values of n. We find that for particularly large values of m, our n to m ratio needs to be much greater. For the most part, though, taking n to be $3m$ or $4m$ is more than sufficient to give us good results.

6 The Critical Exponent

Often we are concerned with finding solutions when the non-linearity, function f, is both superlinear and subcritical, namely when $f(u) = su + u^3$. Another interesting problem to consider, however, is the critical exponent case. In this case, since we are in $\mathbb{R}^3$, $N = 3$ and the critical exponent $p = \frac{N+2}{N-2} = 5$ and the non-linear function in question is $f(u) = su + u^5$.

Previous research questions have already been explored and answered for the critical exponent case for solutions on the ball with radial symmetry. One important theorem in [2] states there exists some value $s_*^1 > 0$, such that no solutions exist for $s < s_*^1$. However, when we numerically analyze the ODE using our tGNGA algorithm, we find spurious solutions that cross the theoretical asymptotic value, $s_*^1 = \frac{1}{4}$, for the first branch. Even though [2] proves there are no continuous solutions to (1) that exist for $s < s_*^1$, our algorithm finds solutions to the discretized problem for even negative s values. This discrepancy between the proven theorem and our numerical results leads us to examine why we obtain these spurious solutions when we implement our algorithm, and to study how we can better our calculations to distinguish between real and spuriously obtained solutions.

7 Spurious Solutions

Before tackling the problem of spurious solutions, we carefully outline the difference between the results we would expect and the solutions we actually obtain using our numerical methods.

We analyze our solutions to the ODE using bifurcation diagrams which plot the value of a given schematic function value versus the bifurcation parameter s. A schematic function is a function $\eta : H \to \mathbb{R}$ that takes a solution in H and returns a unique scalar representation of that solution. The main schematic function we choose for our work on the ball is the $u(0)$-schematic function for the y-axis because it is a good scalar representation of our solutions, and it helps us analyze our results by means of a 2-dimensional cross section. Note that we calculate the value $u(0)$ as follows:

$$\begin{aligned}
u(0) &= \lim_{r\to 0} \sum_{k=1}^{\infty} a_k \psi_k \\
&= \lim_{r\to 0} \sum_{k=1}^{\infty} a_k \frac{\sin(kr)}{\sqrt{2}\pi r} \\
&= \sum_{k=1}^{\infty} a_k \lim_{r\to 0} \frac{\sin(kr)}{\sqrt{2}\pi r} \\
&= \sum_{k=1}^{\infty} a_k \lim_{r\to 0} \frac{k\cos(kr)}{\sqrt{2}\pi} \qquad (\textit{L'Hopital's rule}) \\
&= \sum_{k=1}^{\infty} a_k \frac{k\cos(0)}{\sqrt{2}\pi} \\
&= \sum_{k=1}^{\infty} a_k \frac{k}{\sqrt{2}\pi}
\end{aligned} \tag{7}$$

Other schematic functions we utilize in our bifurcation diagrams include the L_2 norm and the

Sobolev Space H norm. Appendix C explains the definitions of these schematic functions and how we calculate them for our purposes. It suffices to understand that these schematic functions are simply ways to represent a function u that is in our space. Since norms are both unique for each u and scalar, they provide us with a means by which to understand and plot functions that are otherwise hard to picture because they reside in an infinite-dimensional space.

If our results were to be consistent with the theorem given in [1, 2], we would expect the branches of our bifurcation diagram to approach vertical asymptotes at specific, positive values of s. We know from [2] that this s value is $\frac{1}{4}$ for the first branch. We will for the time being treat it as unknown, and later will come up with approximations based on our own algorithms. Despite the existence of this theoretical asymptote, when we run our code for the tGNGA and follow several branches, we see that we are still obtaining 'solutions' beyond the theoretical asymptotic value, as shown in Figure 7.

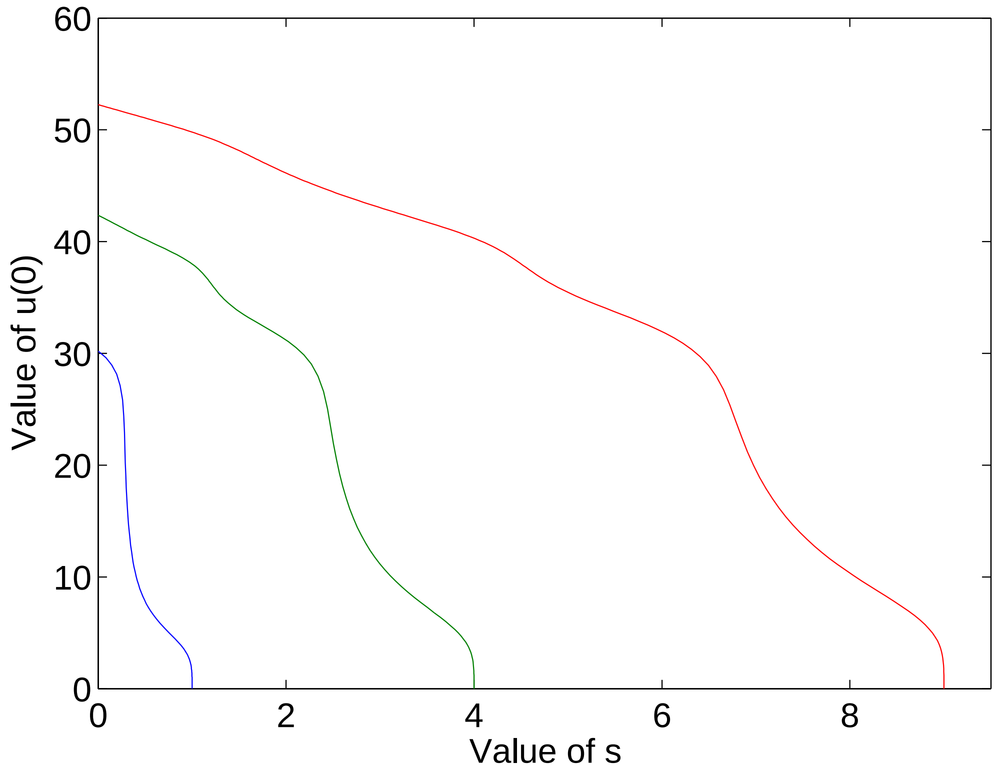


Figure 7: Bifurcation diagram for (6) with $f(u) = su + u^5$ for first three branches, when $m = 100$, $n = 4m + 1$, Ω is a ball in $\mathbb{R}^3$ of radius π, and we consider only solutions with radial symmetry. The colors represent the different bifurcations from the trivial branch at $s = 1, 4$ and 9, with the first denoted in blue, the second in green, and third in red.

To begin understanding our spurious solutions, we first review how we numerically calculate solutions u. Our solutions are approximated by a generalized Fourier series truncation and have the form $u = \sum_{k=1}^{m} a_k \psi_k$. In this equation, the a_k values are the coefficients of our series expansion, and the ψ_k are the normalized eigenfunctions of the Laplacian that we previously mentioned, detailed in Appendix B. Our basis $\{\psi_1, \psi_2, \ldots, \psi_m\}$ is referred to as a Galerkin basis; a truncation of our infinite-dimensional space. Here, m stands for the number of these eigenfunctions in our basis, namely where we truncate our infinite sum.

Truncating the expansion at some large m gives us fairly accurate solutions in other cases, such as when we are dealing with subcritical equations $f(u) = su + u^3$, where our calculations of u do not diverge anywhere to infinity. However, we note that in the critical exponent case we expect our calculation of $u(0)$ to blow up at the vertical asymptotes as $m \to \infty$. It makes sense that since we are dealing with solutions that diverge to positive infinity, our spurious results emerge because of problems that come with using a finite truncation to calculate u. We state the following conjecture:

Conjecture 1. *As we increase m towards ∞, our discrete numerical solutions converge to the continuous theoretical solutions of* (1)

If this conjecture is true, it would mean that the higher we increase m, the more our bifurcation branches would approximate the asymptotic behavior we would expect from the continuous theoretical solutions. Therefore, $u(0)$ will diverge to infinity for any value of $s \leq s_*^1$, and it will converge for any value to the right of our asymptote, namely $s > s_*^1$. We test our conjecture and proceed with our analysis of the ODE by observing what happens to our results when we run our code for higher values of m. In this way, we increase the number of ψ_k functions in our basis, which betters our approximation of u. We are hoping that better approximations of u within a neighborhood of s_*^1 will shed new light on our problem. Instead of running the tGNGA to plot multiple branches for our bifurcation diagram for a given m, we choose to plot only one branch, but plot it multiple times for differing values of m. Taking m to range from 30 to 150 in steps of 10, we obtain Figure 8.

From this figure we see that the solution is beginning to display an asymptotic behavior at some small positive value of s, which we conjecture to be the value s_*^1. This plot is an encouraging confirmation that we are on track to explain the spurious results we obtain by using our numerical approximations. It also leads us to believe we will be able to understand and predict how to use our algorithms in such a way that we know which solutions are true and which are spurious.

8 Implementation of *mGNGA*

In order to better test whether our solutions diverge to infinity at a small positive s value, we implement a new variation of the GNGA which we call the mGNGA.

8.1 *mGNGA*

The algorithm mGNGA is called for a specific value of s, which we refer to as s_0, and a corresponding approximation for the coefficient vector a of u, which we label a_0. We run the tGNGA and plot the first branch of our bifurcation diagram for some average-sized value m, such as 30. We follow the branch until we reach s_0, at which point we pass s_0 and the corresponding a_0 vector as parameters to the mGNGA. The code for mGNGA then runs Newton's Method for increasing values of m. For the first m value we choose, (here $m = 20$), we use a_0 as an initial guess. Newton's Method then computes a vector a_{sol} from which we can calculate the approximate solution of u at $s = s_0$ for the specified number of modes m. It also returns a value for $u(0)$ which we later use in our plot. For each consecutive m value for which we run the mGNGA, we update our Newton's

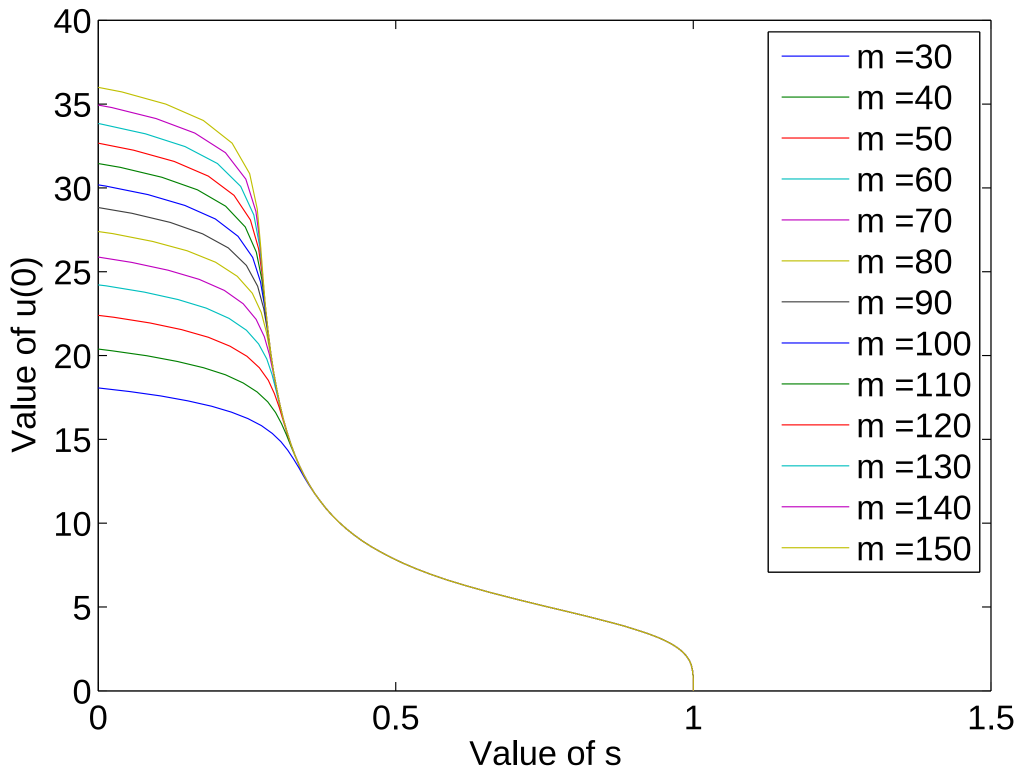


Figure 8: The $u(0)$ bifurcation diagram for (6) with $f(u) = su + u^5$ of first branch bifurcating from $s = 1$. Ω is a ball in $\mathbb{R}^3$ of radius π, and we consider only solutions with radial symmetry. Number of modes m ranges from 30 to 150 in steps of 10, and $n = 4m + 1$.

method guess to be the a_{sol} value most recently calculated, expanded with the appropriate number of zeros so that it continues to be an mx1 vector even when m increases.

We must note the importance of updating the initial guess every iteration, rather than just using a_0 for every new m. When we run the mGNGA without honing in more closely on the correct solution every new iteration, our results become chaotic and not useful. By beginning at a low m and allowing our initial guess to improve with every iteration, our results follow a more steady and understandable pattern. A more concise outline of the mGNGA is displayed below as Algorithm 3.

8.2 Results of *mGNGA*

The first value of s that we use to test the mGNGA is zero. We know that $s = 0$ lies to the left of the theoretical vertical asymptote for the first branch, which [2] explains is at $s_*^1 = \frac{1}{4}$. We know that there are no true solutions at this value of s, but we have conjectured that the value of $u(0)$ will diverge to infinity if we run Newton's Method for a value to the left of s_*^1 for higher and higher values of m. We can see that this is certainly the case for Figure 8, but we will confirm our belief that the solution diverges by printing out the exact values for $u(0)$ for increasingly values of m.

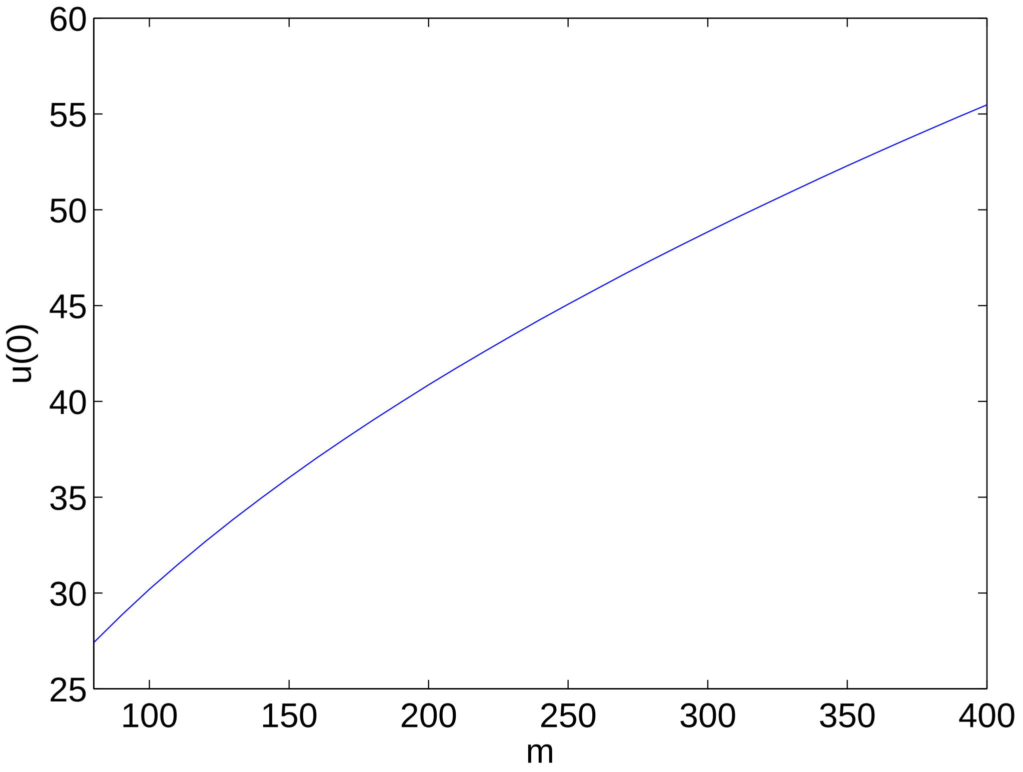
We find that as we increase m, our values for $u(0)$ appear to increase without bound. Our algorithm displays the results as a graph of $u(0)$ vs m, and also also returns the numerical values

Algorithm 3 mGNGA

```
Require: Specific $s^0$ of interest and corresponding coefficient vector $a^0$ of $u^0$
  $U = \emptyset$
  $mVals = \emptyset$
  $a = a^0$
  for $m = mLow$ to $mHigh$ do
      Run Newton's Method using $a$ as an initial guess
      $a = a_{sol}$, returned by Newton's Method
      Append $u(0)$ to vector $U$
      Append $m$ to vector $mVals$
  end for
  Return $mVals$ and $U$
```

for the x- and y- axis in a table. In Figure 9 below are the results for running the mGNGA at $s = 0$ when m varies from 80 to 400 in increments of 10. The corresponding table of m and $u(0)$ values are found in Table 1 as well.

Figure 9: Plot of m vs. u(0) for s = 0



Now that we have seen what happens to $u(0)$ at a point to the left of the vertical asymptote, is it

Table 1: m Values with their corresponding u(0) values

m	u(0)	m	u(0)	m	u(0)	m	u(0)
80	27.412628	180	39.011129	280	47.377946	380	54.225137
90	28.843549	190	39.947113	290	48.117270	390	54.852645
100	30.190262	200	40.856577	300	48.842817	400	55.471447
110	31.465416	210	41.741546	310	49.555291	410	56.081877
120	32.678808	220	42.603800	320	50.255339	·	·
130	33.838210	230	43.444920	330	50.943558	·	·
140	34.949907	240	44.266310	340	51.620498	·	·
150	36.019064	250	45.069230	350	52.286669	·	·
160	37.049986	260	45.854810	360	52.942545	490	60.698022
170	38.046309	270	46.624073	370	53.588564	500	61.245262

worth running the mGNGA to see what happens to the right of it. Our solution diverges at $s = s^1_*$, but to the right of this asymptote we should see our solutions converging for some finite m. This illustrates that our numerical approximation agrees with the theory that solutions do exist to the right of the asymptote.

By inspecting Figure 8, we can select a point that is clearly to the right of the asymptote, even if we are unsure of the exact s value of the vertical asymptote at s^1_*. We choose $s = .45$, and run the mGNGA at this point to observe what happens. According to the mGNGA, our value of $u(0)$ converges extremely quickly to a constant value. This convergent behavior is exactly what we would expect when we are looking to the right of the asymptote.

We already know from [2] that this first asymptote is located at exactly $s = \frac{1}{4}$, which we denote s^1_*. Another research report, [3], informs us that the second asymptote is located at $s = 2.25$. We denote this value by s^2_*. We are unsure, however, where the asymptotes of other branches are found. Our goal is to use our own algorithms and data to give us reasonable approximations of s^1_* and s^2_*. Once we are satisfied with our techniques for finding the first two known asymptotes, we can apply our methods to approximate the unknown asymptote of the third branch. We now turn to calculating good approximations for s^1_*.

9 Numerical Approximations of s^1_*

We can see from our bifurcation diagram that the first asymptote must lie somewhere between 0.2 and 0.3. By zooming in on our graph, we can further guess that the value of s^1_* is between 0.24 and 0.29. We are interested in finding better approximations than this, however, through more careful quantitative analysis.

9.1 Fitting Inflection Points

By observing Figure 8, we notice that for every value of m, the first branch changes curvature before crossing s^1_* and beginning to show spurious solutions. While in theory the branch should diverge to infinity and contain no inflection point, it eventually deviates from what it should truly look like, since m is finite. This causes the curvature of the branch to change sign, indicating an inflection point. We seek to obtain a sequence $\{s_i^1\}_{i=1}^m$ where s_i^1 is an approximation of the inflection point for every value m. Our goal is to see if this sequence converges to s_*^1, i.e., if $\lim_{i\to\infty} s_i^1 = s_*^1$.

We approximate the inflection points in two different ways. For our first method, we find the steepest slope of each branch. We take the two points whose discrete slope $\frac{\Delta u(0)}{\Delta s}$ is greater in absolute value than the slope of other pairs of points which lie in the general region of the asymptote. We then compute the midpoint between this pair of points as an approximation to the inflection point. A plot of these steepest slope points for each value m is displayed in Figure 10.

A more rigorous method for finding the inflection points involves fitting the points of our bifurcation diagram to a cubic function. To do this we fix a value m, and step through the points of one branch, taking them in groups of four. We fit a cubic function to each group of four consecutive points $\{s_a, s_b, s_c, s_d\}$, and then compute where the second derivative of the function is zero to find the inflection point. When we find group of points whose cubic fit has an inflection point located between s_b and s_c, we know this is the best approximation of the inflection point for the whole branch with specified m value. We then use the same method to find the inflection point for the first branch plotted for a higher m value. By varying over a number of m values, we obtain our sequence $\{s_i\}_{i=1}^m$. The plot of the sequence for this second method is seen in Figure 11.

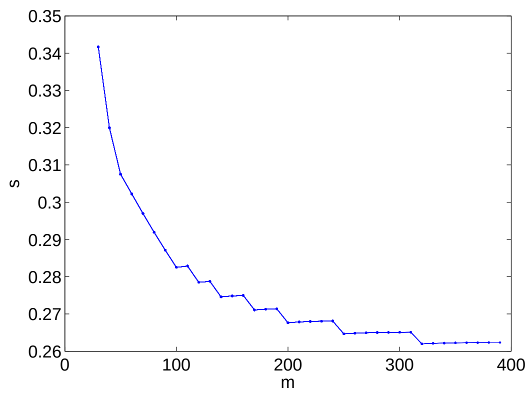


Figure 10: Sequence of inflection points approximated by midpoint of the two points where $\frac{\Delta u(0)}{\Delta s}$ is maximized in a region to the right of the asymptote $s = s_*^1$. The number of modes m ranges from 30 to 390 in steps of 10, with n once again defined as $4m + 1$.

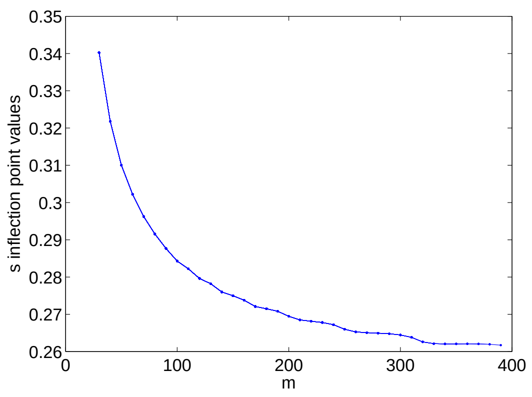


Figure 11: Sequence of inflection points found using cubic fit on a group of four consecutive points from our bifurcation diagram, where the inflection point of the fit is located between the middle two of the four points.

We notice that our first method returns much poorer results than our method using cubic fits. Our first method looks at only two points at a time, and simply takes the midpoint of the steepest region, rather than finding a more exact location of the inflection point within the interval. Our cubic fit method takes four points at a time, and finds the exact place where the second derivative of the fitted function is zero. This results in smoother data.

In both cases, the graph of our sequence approaches a horizontal asymptote as m gets large, and does appear to converge to a certain s, which we take to be s_*^1. Qualitatively, this convergent sequence helps confirm our conjecture that there is vertical asymptote approximated by the location of the inflection points of the plots for the first branch. Because the inflection point always occurs to the right of the asymptote, this method is able to at least give us an upper bound approximation for s_*^1. When we run our code for m ranging from 30 to 390 in steps of 10, we obtain a sequence whose last value is equal to $s = 0.261731$. Running our code for the single value $m = 500$ and $n = 4m + 1$, we obtain an upper bound of 0.2590 for s_*^1.

We attempted to fit the points of these two curves to exponential functions to find the value they converge to. Unfortunately we were unable to obtain a fitted function that converges to a better estimate of s_*^1 than the upper bound we found. The limitation in this method of finding a convergent sequence lies in the fact that the theoretical branch would not really exhibit an inflection point, since it would simply diverge to infinity. Our sequence values are therefore values that hint at the shortcomings of the discretization used in the GNGA, since they point to where our bifurcation diagram begins to deviate from the true branch. While it is helpful to obtain an upper bound for s_*^1 using this method, we are interested in obtaining a more accurate approximation of this value.

9.2 Return to mGNGA

The most successful method we use to approximate s_*^1 comes about by returning to the mGNGA. This time, we use the tGNGA to follow a branch into a narrow window between $s = .24$ and $s = .28$. This is sufficient to give us a fair initial guess a_{guess} for Newton's Method to use. We then loop through values of s between 0.24 and 0.28, with the small increment of 0.001, calling mGNGA on each s. By printing tables and plotting m vs $u(0)$ for each iteration of the loop, we are able to carefully inspect our data to see when $u(0)$ converges, which indicates an s to the right of the asymptote, and where it diverges, which indicates an s to the left of the asymptote. Figure 12 shows the results of running the mGNGA for $s \in \{.24, .25, .26, .27\}$. The data we obtain from this experiment, coupled with the plot of m versus $u(0)$ branches for differing s values, convince us that s_*^1 lies somewhere between $s = 0.24$ and $s = 0.26$.

9.3 Curve-Fitting the Data

The mGNGA does more than provide us with a good window for where s_*^1 lies. Not only can we tell which s values are converging, but we know what value of $u(0)$ they converge to. When we run the mGNGA on values of s from .28 to .26, we obtain a series of $(s, u(0))$ that are known to be to the right of s_*^1. We use these points to reconstruct the first branch of the bifurcation diagram to be void of spurious solutions. Our next step is to try to recreate the actual branch using a statistical best fit. Once we have a fitted function, we compute the asymptote where it diverges to infinity.

To do this, we need to utilize the upper bound, for which we trust the values of $u(0)$, and ignore values beyond that are not converging.

We begin by exporting our believed data, namely the s values and their corresponding $u(0)$ values greater than .264 to a `.dat` file, for $m = 380$ and $n = 4m + 1$ with an s range from 1 to .264 to be read into a a *Mathematica* notebook. In this *Mathematica* notebook, we run non-linear regressions on our data, in an attempt to find a more precise estimate to the vertical asymptote. On the direction of our mentor, Dr. Swift, we tried to fit the data to the functional form

$$f(s) = \frac{c \mid 1 - s \mid^{\alpha}}{\mid s - s_*^1 \mid^{\beta}},$$

with s the variable and c, α, β and s_*^1 unknown, utilizing *Mathematica*'s built in `FindFit` function. To begin our fit, we crafted initial guesses, through trial and error, which would converge to a

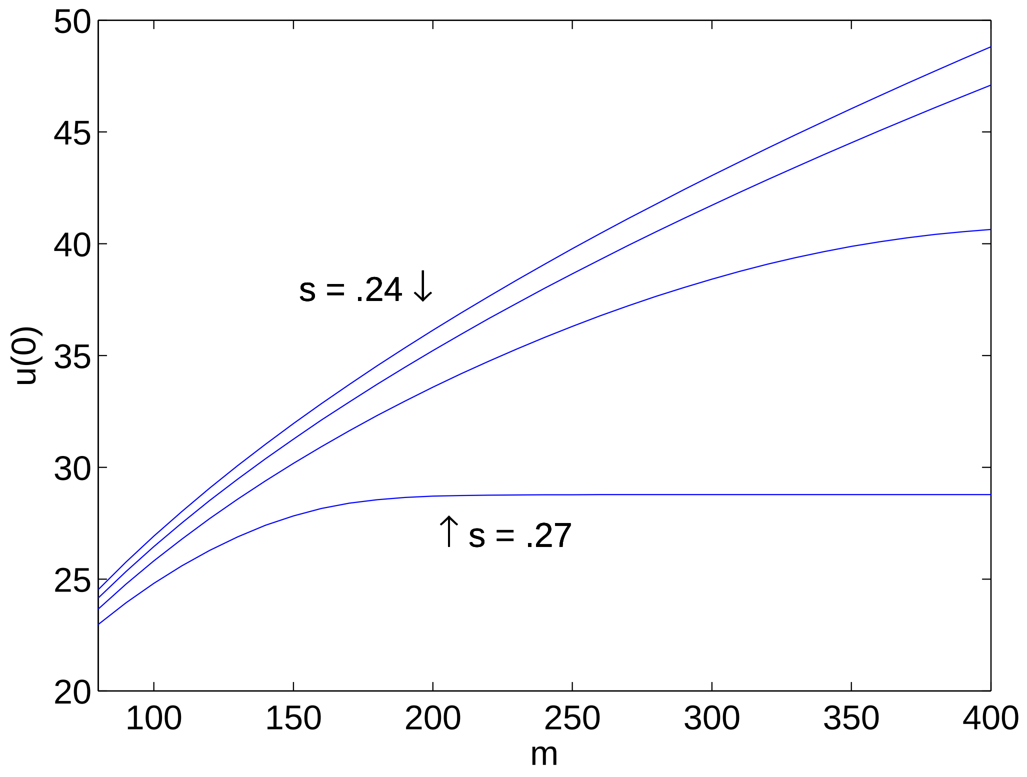


Figure 12: Plot of m versus u(0), for four fixed values s. The first two branches diverge to infinity, whereas $s = .26$ and $s = .27$ approach a horizontal asymptote, indicating points to the right of s_*^1.

reasonable value in a reasonable number of iterations. The fit, as returned by *Mathematica*, is

$$f(s) = \frac{4.77831 \mid 1 - s \mid^{21626}}{\mid s - 0.252215 \mid^{0.46135}}.$$

This is a relatively good fit, as seen in Figure 13. The estimate of s_*^1 returned is 0.252215, something we know is close to the asymptotic value, but a better estimate can be reached. But, we are including the points near 1, which for very low m is close to the value it approaches as $m \to \infty$. A refinement to our data must be made.

We proceeded to remove the data points greater than .5, leaving only the portion of our data that we believe is true, and yet is closer to the asymptote than 1. This yields the fit

$$f(s) = \frac{.391778 \mid 1 - s \mid^{0.0245204}}{\mid s - .249963 \mid^{0.51194}}.$$

In this fit, seen in Figure 14, the s_*^1 value is equal to .249963, a value within .00004 of the theoretical asymptote. We consider this a good approximation.

We pursued another means of estimating this asymptote, namely to curve-fitting the L_2 norm bifurcation diagram. Note, this norm actually goes to 0 at s_*^1 as $m \to \infty$. To do this, we wish to fit

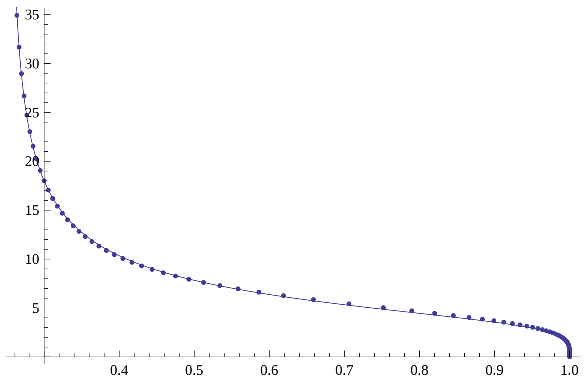


Figure 13: Curve fit to $u(0)$ schematic function data points using *Mathematica*, utilizing the full data set, points in the range $(.264, 1]$.

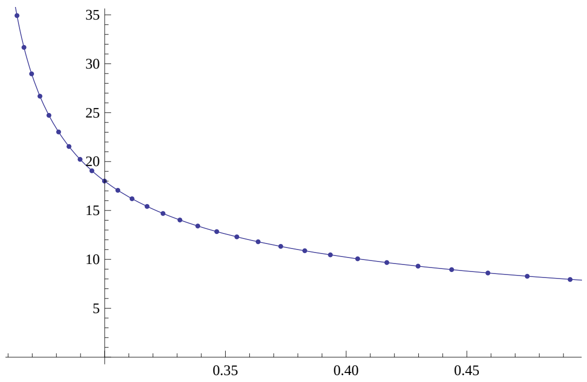


Figure 14: Curve fit to $u(0)$ schematic function data points using *Mathematica*, utilizing the points falling within the range $(.264, .5)$

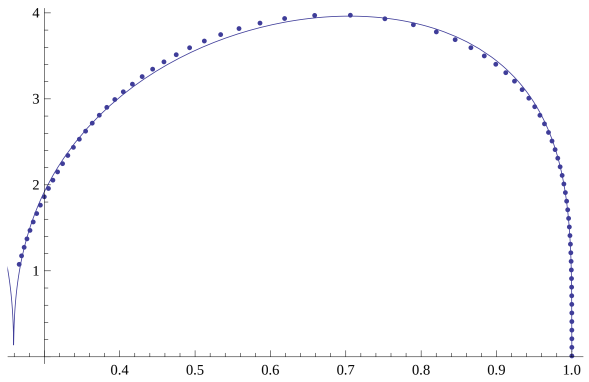


Figure 15: Curve fit to L_2 norm data points using *Mathematica*, utilizing the full data set, points in the range $(.264, 1]$.

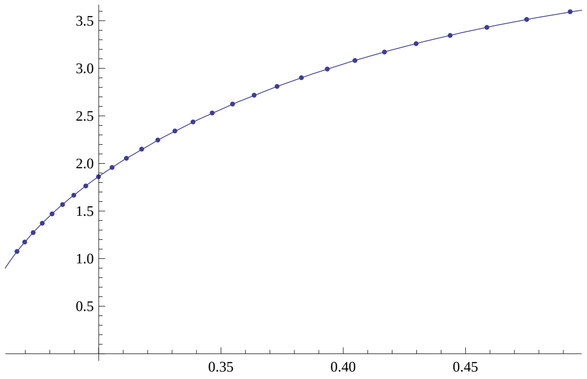


Figure 16: Curve fit to L_2 norm data points using *Mathematica*, utilizing the points falling within the range $(.264, .5)$

the data to a different curve, namely,

$$f(s) = c \mid 1 - s \mid^{\alpha} \mid s - s_*^1 \mid^{\beta} .$$

Once again, we must find a good initial guesses for the constants c, α, β and s_*^1. Then, we proceed to utilize *Mathematica*'s `FindFit` function.

$$f(s) = 7.57468 \mid 1 - s \mid^{0.264385} \mid s - 0.257837 \mid^{0.403305} .$$

This fits the data somewhat passably, as seen in Figure 15 but we can refine this estimate to one much better. Let us again remove the points to the right of .5, and do another fit. This yields

$$f(s) = 10.4742 \mid 1 - s \mid^{0.49972} \mid s - 0.250224 \mid^{0.516107},$$

as seen in Figure 16. This is also a good fit, falling within .0002 from the theoretical asymptote.

We now feel confident that our methods can produce a good estimate of the vertical asymptote, using both the $u(0)$ and L_2 norms. Now, we shall apply these methods to the second branch, and beyond.

10 Branches Beyond the First

Once our experimentation with the first branch was completed, we endeavored to see if our methods could generalize to the second and third branches. Each new branch should correspond to a new asymptote at a different value of s, and we would like to be able to find this value for any bifurcation branch.

10.1 Second Branch and s_*^2

Figures 17 and 18 display our findings for the location of s_*^2, the asymptote that the second branch of our bifurcation diagram approaches.

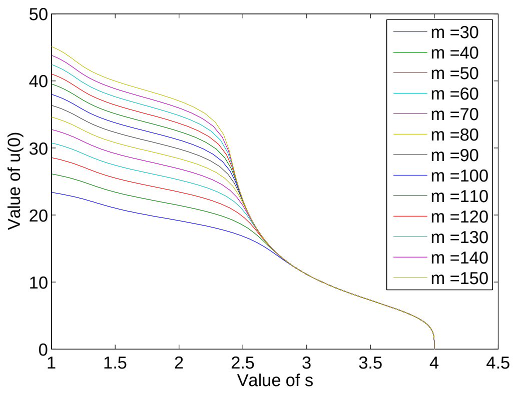


Figure 17: Second branch, bifurcating at $s = 4$ off of the trivial branch, with m ranging from 30 to 150 in steps of 10.

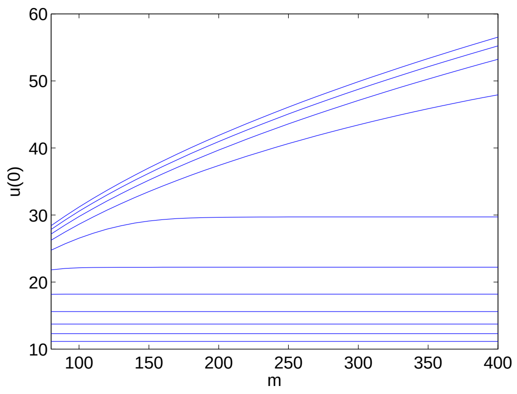


Figure 18: Plot of m versus u(0) for various fixed values of s from 2 to 3, with step size between s values of .1, and with m ranging from 80 to 400.

As with the first branch, we can estimate the location of the second vertical asymptote, and refine the estimate using the mGNGA. Based on our observation, we can see the value is between 2 and 3. Figure 18 shows the output of the mGNGA when run with a starting value of $s = 2$. The first branch can be seen to clearly diverge, but as s increases, the branches diverge more slowly, and even begin converge, as in the case of the last several s values, 2.8, 2.9 and 3.0. Our numerical experimentation shows that the threshold where convergence appears to turn to divergence is between 2.1 and 2.3.

Running our code to find inflection points of the the branches for various m returns similar results. We obtain an upper bound of 2.3156 for s_*^2 when we call our inflection point-finding code when $m = 500$. As before, we can generate a plot of what appears to be a convergent sequence, but we are unable to obtain good results from a best fit of the resulting data.

Best-fitting our bifurcation diagram is once again a more successful approach. Once we have obtained a good enough window for s_*^2, we compute a list of $(s, u(0))$, points to the right of the asymptote, as well as points from our L_2 bifurcation diagram, $(s, \|u\|_2)$. Once again, we use $m =$

400 and $n = 4m + 1$ for our calculations. Exporting this data to $Mathematica$ and running nonlinear regressions, we obtain the following results.

Using data points for the whole branch from $s = 2.32$ to $s = 4$, our best-fit function for the $u(0)$ norm is

$$f(s) = \frac{9.56872 \mid 4 - s \mid^{0.226521}}{\mid s - 2.23349 \mid^{0.572927}}.$$

When we fit only data in the interval $(2.32, 3.5)$, our function gives a slightly better approximation:

$$f(s) = \frac{9.52642 \mid 4 - s \mid^{0.193849}}{\mid s - 2.24287 \mid^{0.565529}}.$$

The plots both of these functions can be found in Figures 19 and 20.

The resulting functions for our L_2 norm bifurcations diagram are

$$f(s) = 2.84122 \mid 4 - s \mid^{0.256051} \mid s - 2.25498 \mid^{0.535318},$$

for the whole branch. For the interval $(2.32, 3.5)$, we get

$$f(s) = 2.87045 \mid 4 - s \mid^{0.293633} \mid s - 2.24498 \mid^{0.575691} .$$

The plots corresponding to these best-fit approximations are found in Figures 21 and 22.

From this data, our estimate of the vertical asymptote is around 2.23349 to 2.25498 for the location of the second asymptote in our bifurcation diagram. We can see that our numerical approximations are once again consistent with previous research done. According to [3], the exact value of s_*^2 is 2.25.

10.2 The Third Branch and s_*^3

Although we can compare our numerical calculations of s_*^1 and s_*^2 to already known values, we are unsure of the exact location of the third asymptote, s_*^3. Noticing the pattern of exact values of $s_*^1 = \frac{1}{4}\lambda_1 = \frac{1^2}{2^2}\lambda_1$, $s_*^2 = \frac{9}{16}\lambda_2 = \frac{3^2}{4^2}\lambda_2$, we make the following conjecture:

Conjecture 2. *The third branch's asymptote will appear at* $\frac{5^2}{6^2}\lambda_3 = \frac{25}{36}\lambda_3 = 6.25.$

We are unable to provide a reference to a proof of this closed form result, but because our methods return good approximations for s_*^1 and s_*^2, we are confident that we can come up with a reasonable numerical approximation for s_*^3 as well.

Note, these data sets were generated with $m = 1000$ and $n = 8m + 1$.

Using the full data set, from 6.4 to 9, the fit obtained is

$$f(s) = \frac{14.1645 \mid 9 - s \mid^{0.240559}}{\mid s - 6.24443 \mid^{0.560835}}.$$

Refining the data set, removing those values close to 9, namely points in the interval $(6.4, 8)$ we get the fit

$$f(s) = \frac{14.4007 \mid 9 - s \mid^{0.217368}}{\mid s - 6.2338 \mid^{0.582463}}.$$

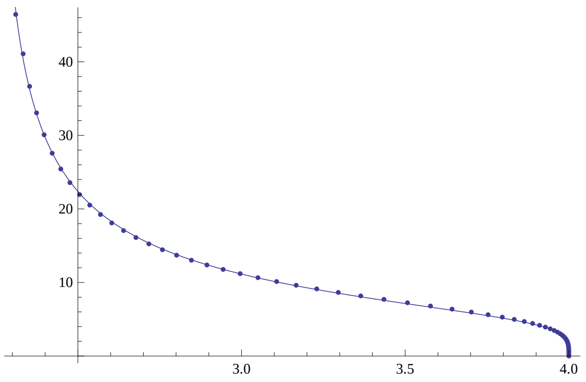


Figure 19: Curve fit to $u(0)$ schematic function data points using *Mathematica*, utilizing the full data set, points in the range $(2.32, 4]$.

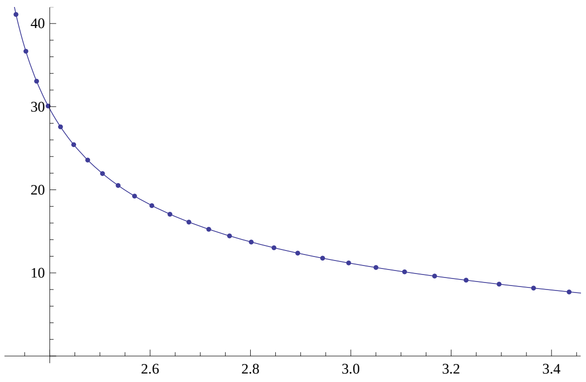


Figure 20: Curve fit to $u(0)$ schematic function data points using *Mathematica*, utilizing the points falling within the range $(2.32, 3.5)$

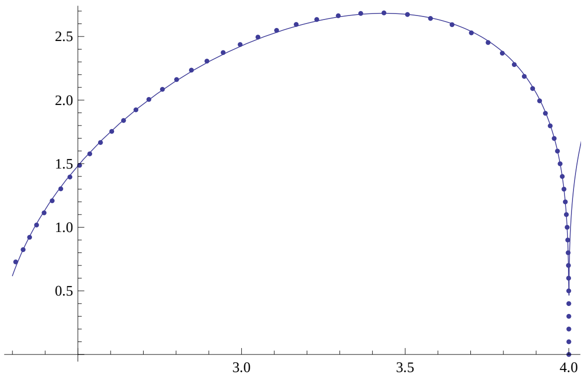


Figure 21: Curve fit to L_2 norm data points using *Mathematica*, utilizing the full data set, points in the range $(2.32, 4]$.

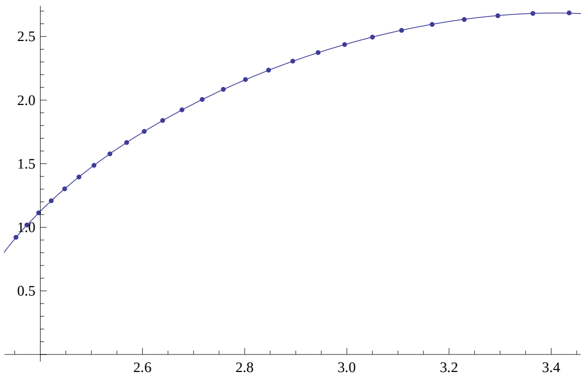


Figure 22: Curve fit to L_2 norm data points using *Mathematica*, utilizing the points falling within the range $(2.32, 3.5)$

Using the L_2 norm bifurcation diagram, we get two fits, and more estimates of the vertical asymptote. Using the full data set, we get the following fit.

$$f(s) = 1.55979 \mid 9 - s \mid^{0.252971} \mid s - 6.25363 \mid^{0.574459} .$$

Restricting the data to the interval $(6.4, 8)$ we get the fit

$$f(s) = 1.51871 \mid 9 - s \mid^{0.275706} \mid s - 6.22931 \mid^{0.610095} .$$

These estimates of the third asymptote, $s_*^3 \in [6.22931, 6.25363]$, are quite close to our conjectured value, which leads us to trust our results.

11 Final Results and Conclusion

By focusing on one branch of our bifurcation diagram and varying the number of eigenfunctions, by varying the number of modes m in our basis, we are able to better understand and interpret the

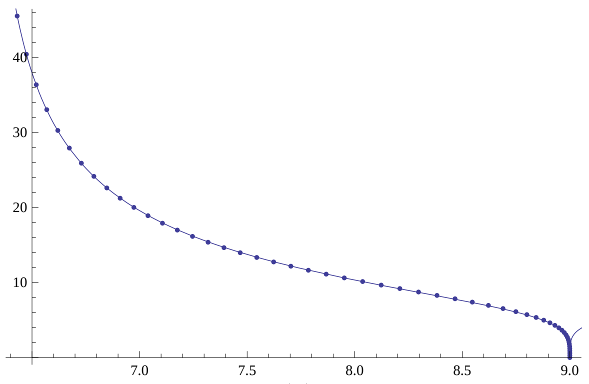


Figure 23: Curve fit to $u(0)$ schematic function data points using *Mathematica*, utilizing the full data set, points in the range $(6.4, 9]$.

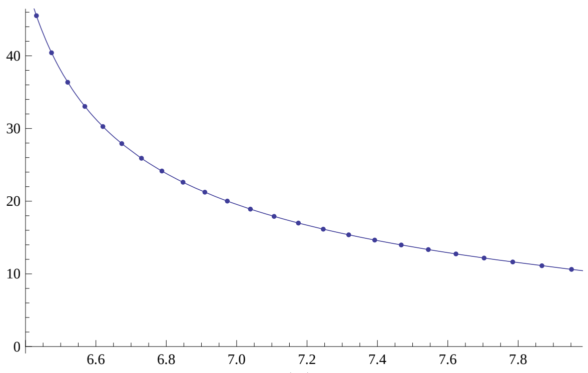


Figure 24: Curve fit to $u(0)$ schematic function data points using *Mathematica*, utilizing the points falling within the range $(6.4, 8)$

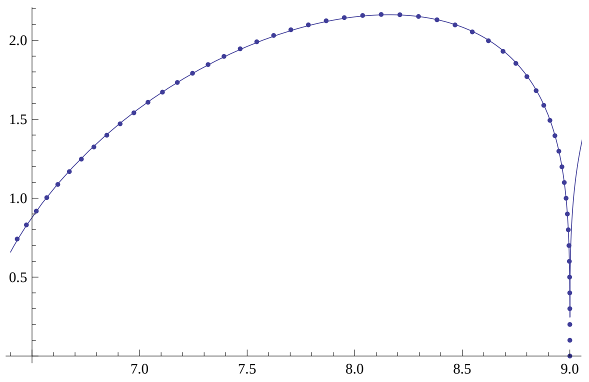


Figure 25: Curve fit to L_2 norm data points using *Mathematica*, utilizing the full data set, points in the range $(6.4, 9]$.

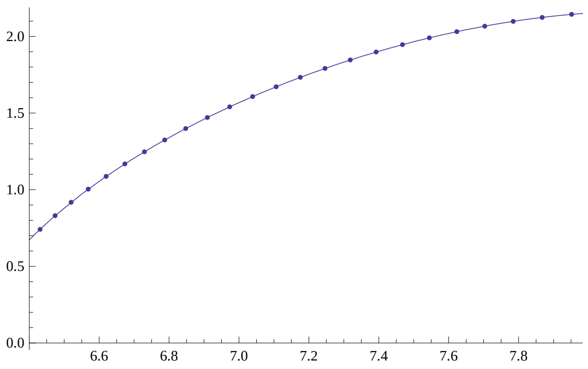


Figure 26: Curve fit to L_2 norm data points using *Mathematica*, utilizing the points falling within the range $(6.4, 8)$

numerical solutions we obtain for our ODE using the tGNGA. Not only are we able to explain why spurious solutions occur, but we have also successfully determined a method for deciding which parts of our numerical solutions are real, and which are simply mistakes due to truncation.

We can now return to our initial results to produce a better bifurcation diagram. Recall that in Figure 7 we observed spurious solutions for the first three bifurcation branches corresponding to (5). While we would have expected each branch to diverge to infinity at a different positive s value, what we obtained was a plot of three branches which crossed $s = 0$ and displayed solutions that we knew did not really exist for the PDE (1). Namely, our algorithm finds solutions to the discrete problem, that do exist, but these solutions are not solutions to the continuous problem. With our new understanding of spurious solutions, we can construct a diagram that more accurately simulates the behavior we would expect based on the previous theoretical research that has been done for the sphere with radial symmetry. By increasing our number of modes to $m = 400$, we plot a new bifurcation diagram which displays three branches which appear to increase to infinity at vertical asymptotes (see Figure 27).

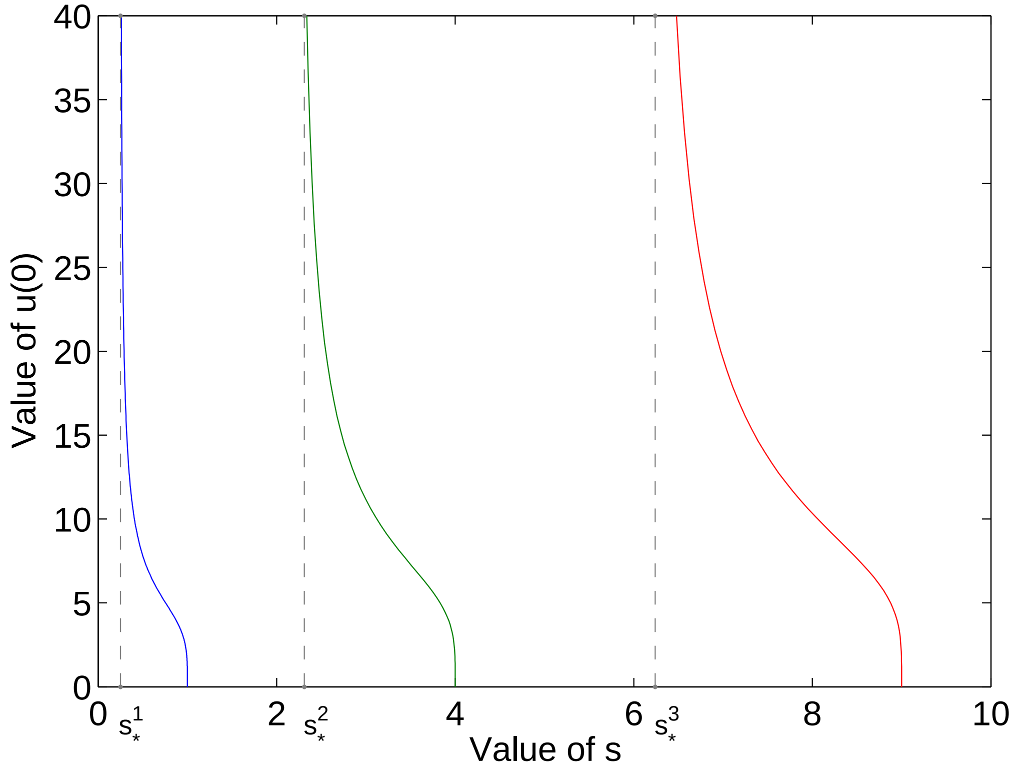


Figure 27: Bifurcation diagram for (5) with $f(u) = su + u^5$ for the first three branches. Here, m is increased to 400 to give more accurate results and simulate the true asymptotic behavior the theoretical branches would. The first and second asymptotes are plotted at $s = .25$ and $s = 2.25$, the exact location of s_*^1 and s_*^2 according to [2] [3]. The third asymptote is plotted at 6.24029, an approximation to s_*^3 using our numerical methods, calculated by taking the average of the four values found in from our statistical best-fits.

While our initial numerical solutions are puzzling at first glance, this final diagram demonstrates that we are still able to obtain correct results and reasonable approximations using our numerical methods. By perfecting these methods we can understand where spurious results exist and how we can fix them. The new ability we have in distinguishing between real and spurious solutions near asymptotes makes our GNGA algorithms more useful and powerful in computing numerical solutions to various ODE and PDE.

12 Future Research Possibilities

During our work we encountered further interesting research questions that we did not have time to fully address or answer. Such questions are good potential research topics for future REU programs or other similar research projects at the undergraduate or graduate level. Two such open questions are stated below.

Open Question 1. *Prove (or disprove) the closed form result of the location of the third asymptote. Is it really located at exactly* $\frac{5^2}{6^2}\lambda_3 = \frac{25}{36}\lambda_3 = 6.25$*? Can this be generalized to find the exact* *location of s_*^k for any branch number k?*

Open Question 2. *Perform further research into the inexact numerical integration in our calculations for the ball. Can we predict how our results will change when we use different m and n values? What is the best m to n ratio for integrating in different situations? What can we say about numerical integration in different regions Ω? For what regions is it exact?*

13 Epilogue: Solutions on the Ball

While most of our paper is focused upon the Bifurcation Diagram, our algorithm also returns discrete numerical solutions, that for a given s a well chosen m value will yield a good approximation of the continuous solution. The following figures were generated with $m = 100$ and $n = 4m + 1$.

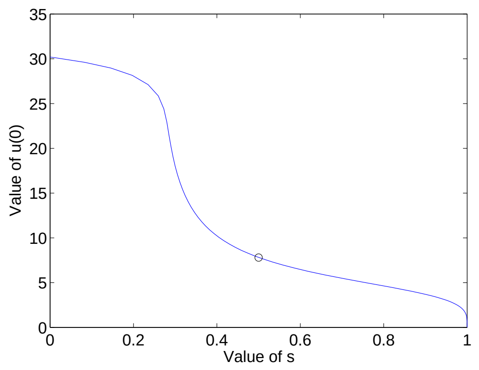


Figure 28: Bifurcation Digram, schematic function $u(0)$ plotted against bifurcation parameter s. The dot signifies the plotted solution to the right.

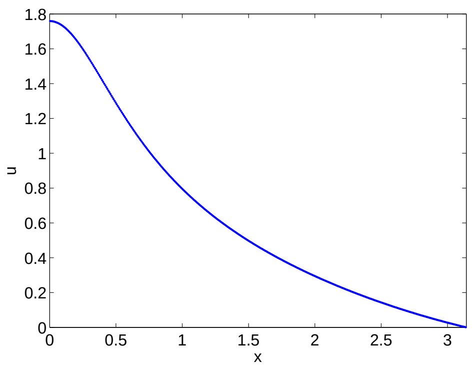


Figure 29: Solution: Plot of u vs x, this solution is located at the dot on the bifurcation diagram in Figure 28.

14 Epilogue 2: The PDE on the Shell

14.1 New Region

It is an interesting result that although no solutions to $\Delta u + su + u^5 = 0$ exist on the ball when $s < s_*^1$, this is not the case when our region is a shell in $\mathbb{R}^3$. In other words, if we take ball of radius r and remove from inside of it a smaller ball of radius r_0, we obtain a region, the shell, or Annulus, for which solutions exist for this PDE at all values of s. In this section we address the difference between solutions on each of these two regions, and use the GNGA to plot solutions and bifurcation diagrams for (5) when $f(u) = su + u^5$, Ω is a shell in $\mathbb{R}^3$, and $u = 0$ on $\partial\Omega$.

14.2 Boundary Conditions

The main difference between the solutions of these two regions lies in the difference between their boundary conditions. Consider when Ω is a ball in three-space of radius r_1. In this case, boundary conditions require that $u(r_1) = 0$ for every solution u to the differential equation. These boundary conditions need to hold for our eigenfunctions, in other words, $\psi_k(r_1) = \frac{\sin(kr_1)}{\sqrt{2\pi} r_1} = 0$. Note that the eigenfunctions, and therefore solutions u, are not defined when $r = 0$.

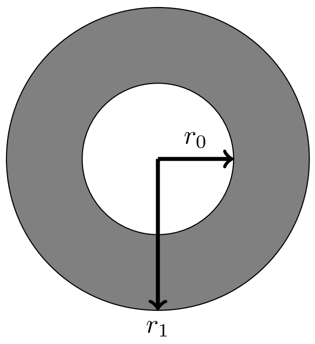


Figure 30: 2D representation of a shell, or annulus. Region has both an inner and outer radius, r_0 and $r_1 = r_0 + \pi$, respectively.

When we consider Ω to be a shell in three-space, however, the boundary conditions require that $u = 0$ both at an outer radius r_1 and an inner radius r_0. In this paper we will choose our shell where $r_1 = r_0 + \pi$, which will help simplify our calculations while still giving us the results we desire. In order to fulfill the boundary conditions, our eigenfunctions need to be different than they were for the ball, which significantly changes solutions to the PDE. These changes are reflected in solution plots and bifurcation diagrams.

14.3 Eigenfunctions for the Shell, Dirchlet Boundary Conditions

Although our boundary conditions are slightly different for the shell, we are still seeking solutions to the Spherical Bessel Equation, as derived in the Appendix:

$$u'' + \frac{2}{r}u' + \lambda u + u^5 = 0. \tag{8}$$

Our eigenvalues are still given by $\lambda_k = k^2$, and since the equation we want to solve is the same as before, we make use once again of the spherical Bessel functions. Namely,

$$\begin{aligned} j_0(x) &= \frac{\sin(x)}{x} \\ y_0(x) &= -\frac{\cos(x)}{x} \end{aligned} \tag{9}$$

Here we note the first difference in the form our eigenfunctions take. Previously we did not use the spherical Bessel functions of the second kind $y_0(r) = -\frac{cos(r)}{r}$, because this function's limiting behavior as $r \to 0$ would create problems for our Generalized Fourier series expansion. Since $lim_{r\to\infty} \frac{cos(r)}{r} = \infty$, if we used y_0 in our eigenfunctions, the principal norm we use in our calculations, $u(0)$, would be undefined. The asymptotic behavior our solutions would exhibit at $r = 0$ would be less that ideal.

We do not have this same problem on the shell, however, since r never equals zero on an annulus. We can make use of the spherical Bessel functions of the second kind in our series expansion without worrying about divergent behavior at any r in our domain.

The general form of our eigenfunctions is therefore

$$u(r) = c_1\frac{\sin(kr)}{kr} - c_2\frac{\cos(kr)}{kr}. \tag{10}$$

The second difference in our eigenfunctions lies of course in the new boundary conditions they must fulfill:

$$u(r_0) = 0 \text{ and } u(r_0 + \pi) = 0, r_0 > 0. \tag{11}$$

As mentioned before, r_0 is the inner radius and $r_0 + \pi$ is the outer radius.

We know that $u(r)$ satisfies the ODE for any c_1 and c_2, but only for proper values of k will this be an eigenfunction. With that in mind, let us find some of those proper values of k such that $u(r)$ is an eigenfunction. Note, utilizing trigonometric identities,

$$u(r) = c_1\frac{\sin(kr)}{kr} - c_2\frac{\cos(kr)}{kr}$$

$$= A\,\frac{\sin(k(r-B))}{kr}$$

Thus,

$$\begin{aligned} u(r_0) = 0 &\implies A\sin(k(r_0 - B)) = 0 \\ u(r_0+\pi) = 0 &\implies A\sin(k(r_0 + \pi - B)) = 0 \end{aligned}$$

Note that A can be any constant. Recall that r_0, our initial radius is given. We can now solve for k, and our introduced constant B. So,

$$\begin{aligned} \sin(k(r_0 - B)) = 0 &\implies k(r_0 - B) = 0 \\ \sin(k(r_0 + \pi - B)) = 0 &\implies k(r_0 + \pi - B) = m\pi \end{aligned}$$

These equations are satisfied when $B = r_0$, and $k \in \mathbb{Z}^+$. Therefore, our normalized eigenfunctions are as follows:

$$\psi_k(r) = \frac{\sin(k(r - r_0)}{\sqrt{2}\pi r} \text{ for } k \in \mathbb{Z}^+.$$

with eigenvalues $\lambda = k^2$.

Observe that zero Dirichlet boundary conditions hold for any positive value of r_0:

$$\psi_k(r_0) = \frac{\sin(k(r_0 - r_0))}{\sqrt{2}\pi r_0} = \frac{\sin(0)}{\sqrt{2}\pi r_0} = 0,$$

and

$$\psi_k(r_1) = \frac{\sin(k(r_1 - r_0))}{\sqrt{2}\pi r_1} = \frac{\sin(k(r_0 + \pi - r_0))}{\sqrt{2}\pi r_1} = \frac{\sin(k\pi)}{\sqrt{2}\pi r_1} = 0.$$

Since r_0 is required to be a positive value for the region to be a shell, we are not concerned that $\psi_k(r)$ is not defined when $r = 0$. Our solutions u therefore no longer exhibit the asymptotic behavior they would for the ball as $r \to \infty$.

14.4 Solutions and Bifurcation Diagrams

We must make a few small changes to the initial conditions in our code in order to plot solutions and bifurcation diagrams for the shell. While the eigenvalues have not changed, we must include code for our new eigenfunctions, and set an initial value for r_0, the inner radius. Once these changes are made, we run the tGNGA to plot a branch of the bifurcation diagram, and also plot a solution corresponding to a specific value of s on that branch. The following figures display the results for the first branch of solutions to the PDE with $r_0 = \pi$ and $r_1 = 2\pi$. We set our lower bound for the plot to be at $s = 0$, at which point we plot the solution diagram.

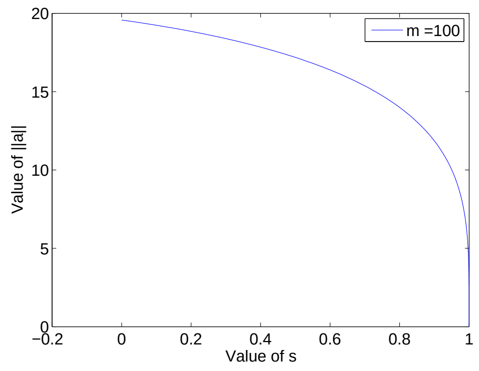


Figure 31: Bifurcation Digram

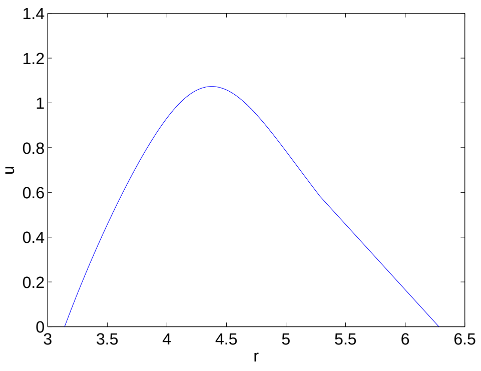


Figure 32: Solution: u vs x

Note that the bifurcation diagram doesn't show any signs that our solutions diverge at some asymptote. We have full confidence that solutions on the branch all the way to $s = 0$ are not spurious. Consider also the plot of the solution at $s = 0$. We can see in the diagram that $u = 0$ at $r = \pi$ as well as at $r = 2\pi$, which is consistent with our boundary conditions.

15 Epilogue 3: Basins of Attraction

15.1 Motivation

Occasionally when solving equations with Newton's Method, it is helpful and interesting to plot a diagram of the basins of attraction. We do this by taking a two-by-two grid of points, separated by a step size, and letting each grid point represent an initial guess for the root of a specific function. When we run Newton's method on one of these points, we can discover which root of the equation of interest each initial guess converges to, or if it converges at all. To visualize this convergence, we assign a color to each root, and a shade to depict how many iterations the point takes to converge to the root. We then plot the grid points and their corresponding color. For example, a dark red point and a light red point both indicate initial guesses that converge to the same root, but the darker shading indicates that it took more iterations in order to converge.

A basin of attraction in this sense is a set of points in our resulting diagram that all converge to the same root of the function in question. Namely, all of the points which are the same color, shading aside. These diagrams are often fractals and are pleasing to the eye as well as enlightening as to the behavior of the function.

15.2 Examples

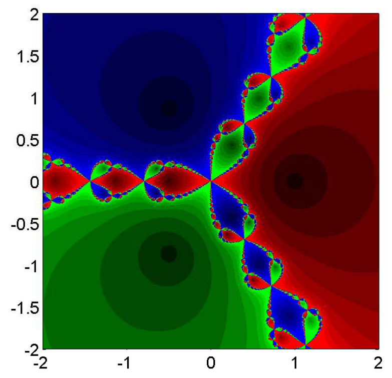
It is easy to plot basins of attraction for simple complex functions such as $g(z) = z^3 - 1$. In this case we simply choose a section of the complex plane for our initial guesses, and run the sequence

$$x_{n+1} = x_n + \frac{g(x_n)}{g'(x_n)},$$

until the initial guess converges to one of three roots.

When we run Newton's Method on the points in the complex plane from -5 to 5 on each axis, we obtain Figure 33.

Figure 33: Basins of Attraction for $g(z) = z^3 - 1$



As we can see, there are three distinct roots that each point might possibly converge to. We can do the math to find that these roots are $1 + 0i$, $-\frac{1}{2} + \frac{\sqrt{3}}{2}i$, and $-\frac{1}{2} - \frac{\sqrt{3}}{2}i$, namely three roots of unity. Notice that there is a relatively large area surrounding each root where points converge to the root closest to them. We also note a fractal boundary between roots, where it is harder to tell exactly where each point will converge, given an initial guess.

15.3 GNGA Basins

Plotting basins of attractions becomes a little more tricky for functionals, because we are not dealing with a 2-dimensional plane but rather an infinite dimensional space such as H. Nevertheless, the

GNGA is still a Newton algorithm and it is possible plot basin of attraction fractals of our functional J by taking a two-dimensional cross section of H and carefully choosing our initial guesses. Before plotting a fractal for the PDE we are studying, we must first choose the region on which we seek solutions, such as the interval or ball, which boundary conditions, Dirichlet or Neumann, what non-linear function we use for $f(u)$. We must also select which cross section of our infinite dimensional space we are going to plot, namely which two basis vectors we shall use as axes.

The following diagrams show the basins of attraction for our functional J when $f(u) = su+u^3$, where $s = \frac{1}{2}$. We choose to solve the PDE on the interval $I = [0, \pi]$ with Dirichlet boundary conditions. We plot the cross section of H spanned by the first two eigenfunctions, ψ_1 and ψ_2. Recall that our solutions u have the form

$$u = \sum_{k=1}^{m} \psi_k a_k$$

where a_k represent the coefficients of a generalized Fourier series expansion. In theory each initial guess for a root of the functional J would be made up of an infinite number of a coefficients, but in practice we know our code approximates a solution using m coefficients. Here we only assign values to the first two coefficients, a_1 and a_2, and let the rest of the $m-2$ coefficients in our initial guess be equal to zero. This allows us to represent each initial guess by a point in a two dimensional plane. We then loop through the pairs of a_1 and a_2 values, calling the GNGA on each pair to find which root of J they converge to. We expect each initial guess to converge to one of 5 roots. The trivial solution at $s = \frac{1}{2}$ is the first, and the other four roots are the positive and negative solutions found at $s = \frac{1}{2}$ on the first and second branches of our bifurcation diagram.

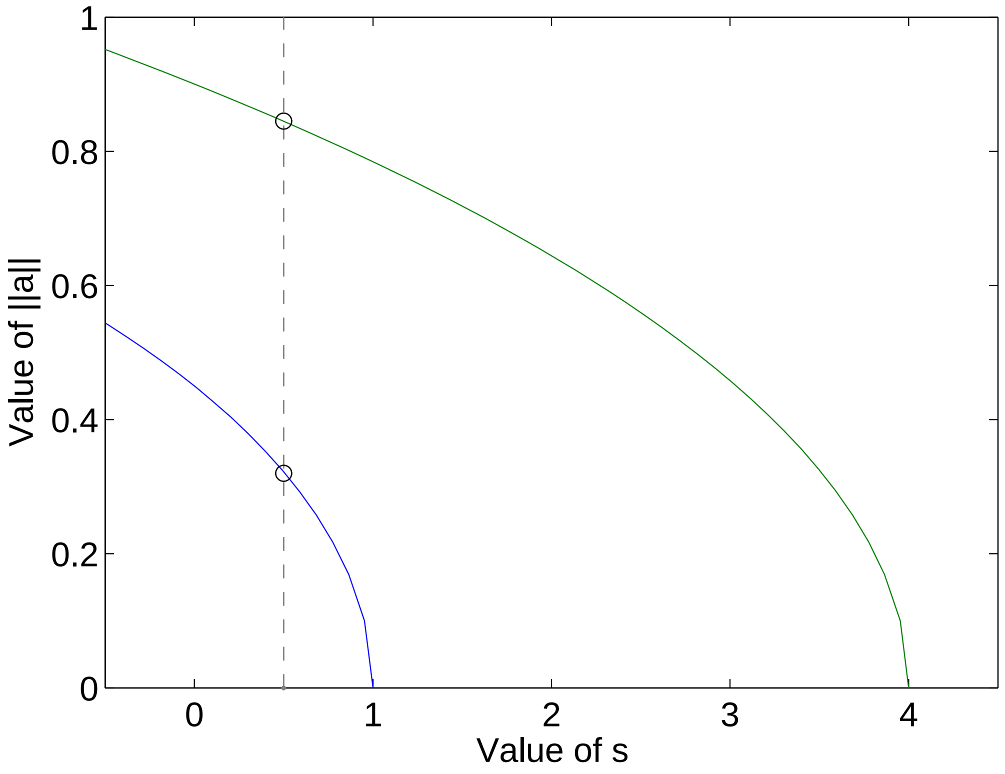


Figure 34: Bifurcation Diagram, Dirichlet boundary conditions, $f(u) = su + u^3$ with points of interest circled.

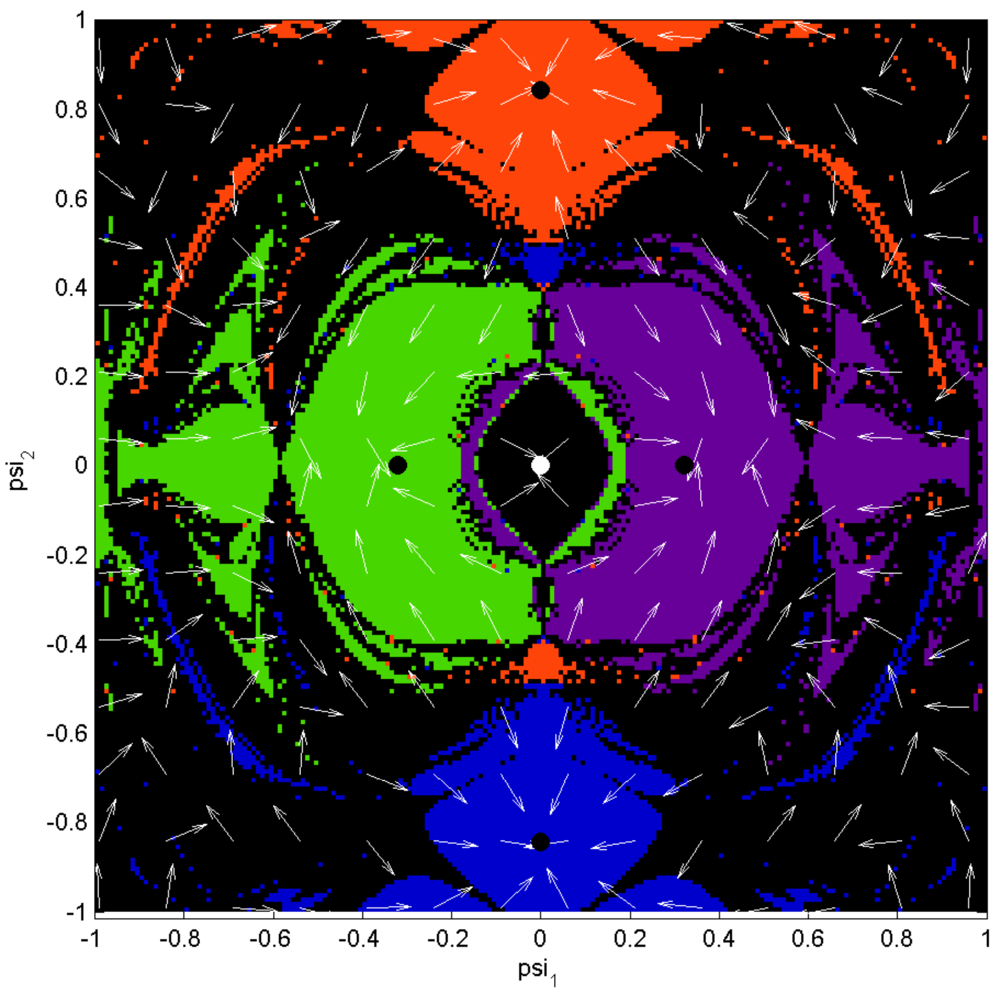


Figure 35: GNGA Basins of Attraction for J, $s = \frac{1}{2}$, with different colors representing the different roots that the points converge to. We have also included points where our roots occur. The black dots represent the roots, with the white dot representing 0. Note that the newton delta here is 1. Also included are the search direction vectors of Newton's Method for a chosen number of initial guesses, for illustrative purposes. The direction vectors show the first two coefficients of the newton step χ. The Newton step shows that Newton's sequence is updating an initial guess in the direction of a specific root.

We can also take different cross sections of our infinite dimensional space, ψ_1 and ψ_3, ψ_2 and ψ_3, and beyond. We also can look into different regions, such as the ball and others. Looking at different cross sections helps one visualize some of the more complex concepts in our infinite dimensional space. Observe, in Figures 36 and 37 the ψ_1, ψ_3 cross section of the ball.

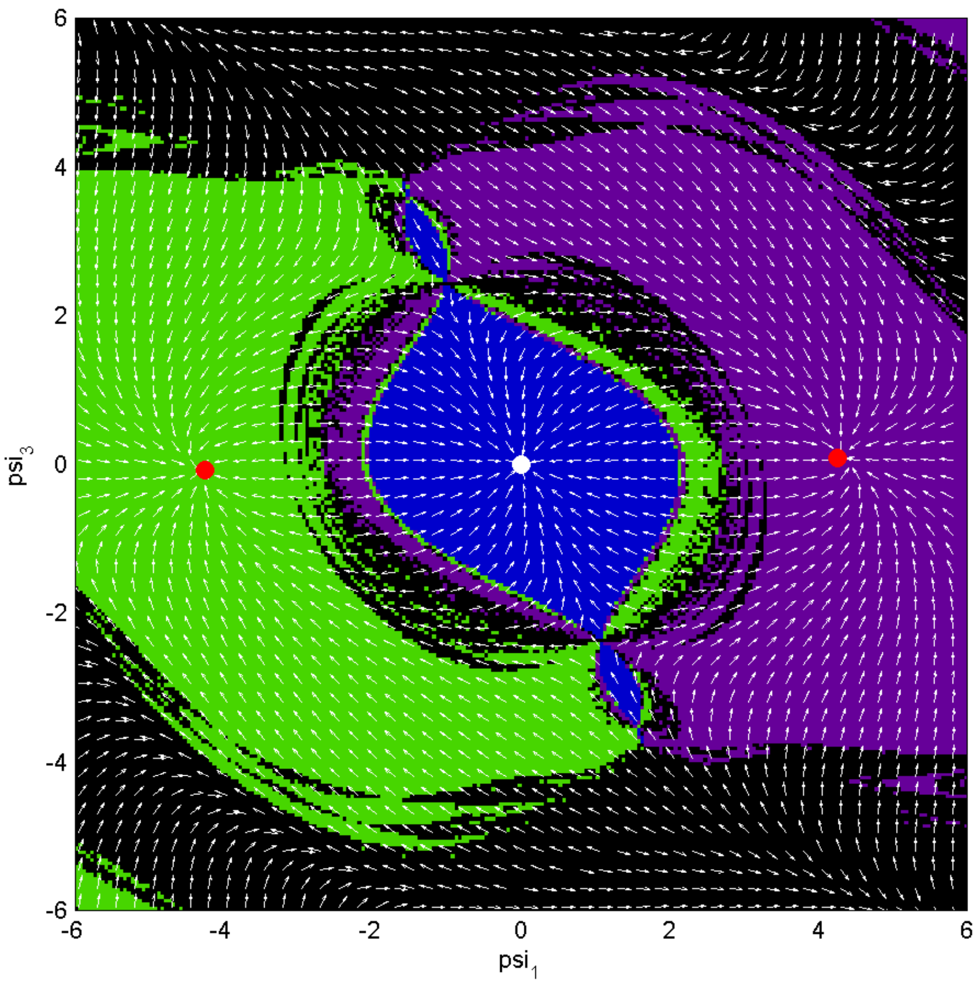


Figure 36: Cross Section: ψ_1 and ψ_3, Ball at $s = \frac{1}{2}$, Subcritical Exponent Case, $f(u) = su + u^3$

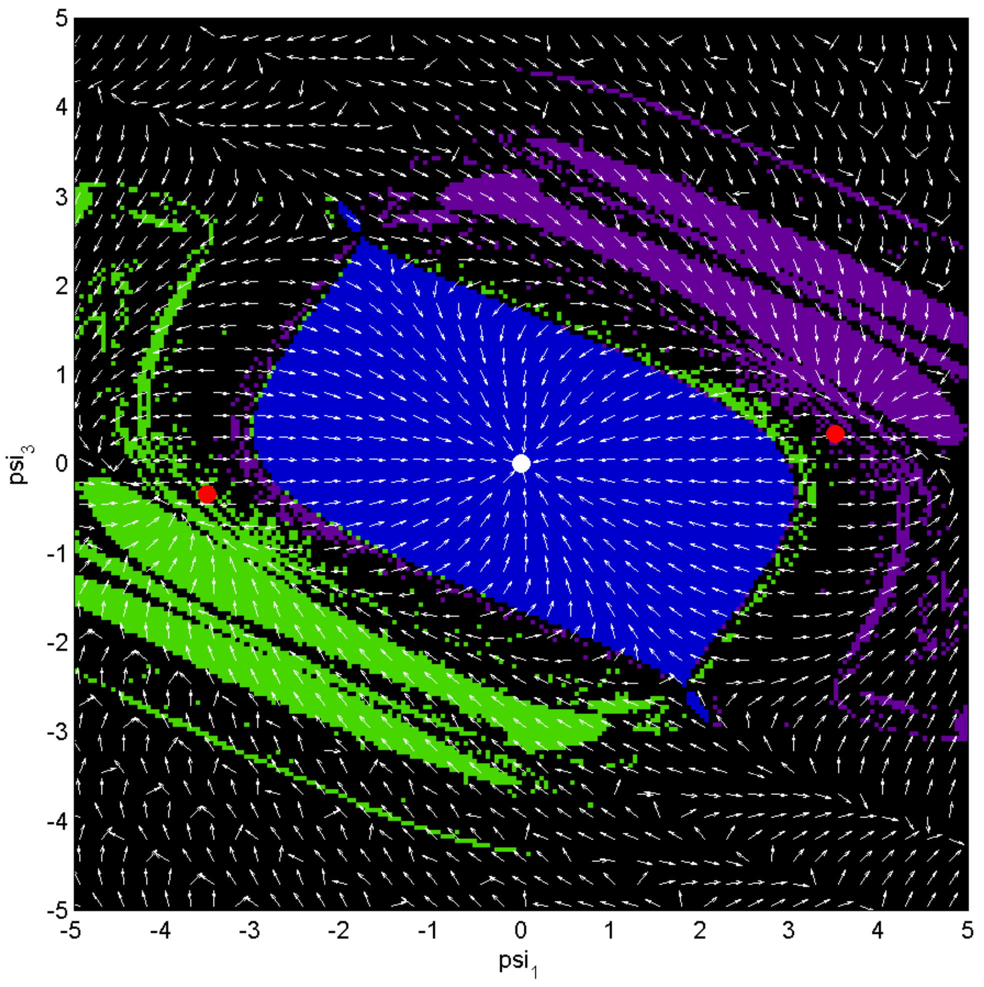


Figure 37: Cross Section: ψ_1 and ψ_3, Ball at $s = \frac{1}{2}$, Critical Exponent Case, $f(u) = su + u^5$

References

[1] Ham Brezis and Louis Nirenberg. Positive solutions of nonlinear elliptic equations involving critical sobolev exponents. *Communications on Pure and Applied Mathematics*, 36(4):437–477, 1983.

[2] C.J. Budd and A.R. Humphries. Weak finite dimensional approximations of semi-linear elliptic pdes with near critical exponents.

[3] Michael Grigsby, Cathy Ho, Hernan Oscco Adriana Melgoza, Michelle Craddock, and Maria Mercedes Franco. On solutions to a nonlinear elliptic equation.

[4] John M. Neuberger, Nándor Sieben, and James W. Swift. Automated bifurcation analysis for nonlinear elliptic partial difference equations on graphs. *Internat. J. Bifur. Chaos Appl. Sci. Engrg.*, 19(8):2531–2556, 2009.

[5] John M. Neuberger, Nándor Sieben, and James W. Swift. Automated bifurcation analysis for nonlinear elliptic partial difference equations on graphs. *Internat. J. Bifur. Chaos Appl. Sci. Engrg.*, 19(8):2531–2556, 2009.

[6] Paul H. Rabinowitz. *Minimax methods in critical point theory with applications to differential equations*, volume 65 of *CBMS Regional Conference Series in Mathematics*. Published for the Conference Board of the Mathematical Sciences, Washington, DC, 1986.

Appendix

A Proof of the Convergence of Newton's Method

Definition 1. *If $g : \mathbb{R} \to \mathbb{R}$ satisfies $g(p) = p$, then we call p a $fixedpoint$.*

Theorem 1. *If $g : [a, b] \to^c [a, b]$ then g has a fixed point.*

Proof. Let $h(x) = x - g(x)$. Also, assume $g(a) \neq a$ and $g(b) \neq b$. Now, since the range of g is $[a, b]$, we have that $g(a) > a$ and $g(b) < b$. This tells us that $h(a) < 0$ and $h(b) > 0$, and since h is continuous, being the difference of two continuous functions, by the $Intermediate\,Value\,Theorem$ there exists a point $p \in [a, b]$ such that $h(p) = 0$. Thus, $h(p) = 0 = p - g(p)$ and we have shown that $\exists\, p \in [a, b]$ such that $g(p) = p$. □

Theorem 2. *If also, $\exists\, \sigma \in (0, 1)$ with $|g'(x)| \leq \sigma$ on $[a, b]$. Then:*

1. *The fixed point on the interval is unique.*
2. *The fixed point iteration, $p_{k+1} = p_k - \frac{f(p_k)}{f'(p_k)}$, converges to p.*

Proof. Let p, q be fixed points over $[a, b]$ Now, the slope of the secant between p and q is $|\frac{g(p)-g(q)}{p-q}|$. However, since p and q are fixed points, we have that the slope is $|\frac{p-q}{p-q}| = 1 \nleq 1$. Thus contradicting our assumption that $\sigma < 1$. Therefore, there can only be one fixed point over$[a, b]$.

Next, let $p_0 \in [a, b]$, and $p_{k+1} = g(p_k)$. By the $Mean\,Value\,Theorem$, there exists a value $c \in [a, b]$ such that for some $r, s \in [a, b]$, $|\frac{g(r)-g(s)}{r-s}| = |g'(c)|$. Using this, it can be seen that:

$$\begin{aligned}
|p - p_1| &= |g(p) - g(p_0)| \\
&= |g'(c_0)(p - p_0)| \\
&= |g'(c_0)||p - p_0| \\
&\leq \sigma|p - p_0| \\
&< |p - p_0|
\end{aligned}$$

Thus, $|p - p_1| < |p - p_0|$, and p_1 is no futher from p than p_0. To show convergence of fixed point iteration, we must show $\lim_{n\to\infty} |p - p_k| = 0$. Observe that $|p - p_k| \leq \sigma^k|p - p_0|$ for k = 1. Now, suppose $|p - p_{k-1}| \leq \sigma^{k-1}|p - p_0|$. From earlier, we know that:

$$|p - p_k| \leq \sigma|p - p_{k-1}|$$

and,

$$|p - p_{k-1}| \leq \sigma^{k-1}|p - p_0|$$

Thus,

$$|p - p_k| \leq \sigma^k |p - p_0|$$

Since $0 < \sigma < 1$, we have that as $n \to \infty$, $\sigma^n \to 0$. Thus, $0 \leq lim_{k\to\infty}|p - p_k| \leq \lim_{k\to\infty} \sigma^k |p - p_0| = 0$. Because $|p - p_0| \to 0|$, it follows that $p_k \to p$. Thus, we have shown that $g(p_{k-1}) = p_k \to p$. □

Theorem 3. *If $f : \mathbb{R} \to \mathbb{R}$ is continuous and twice differentiable, and $f(p) = 0$, $f'(p) \neq 0$, then $\exists$ $\delta > 0$ such that if $p_0 \in B_\delta(p)$ where B is a ball with radius δ, then Newton's Method converges.*

Proof. Let us again define $p_{k+1} = g(p_k) = p_k + \frac{f(p_k)}{f'(p_k)}$. Thus g is a generating function for the sequence P_k with $p_k \to p$ as $k \to \infty$ by $\mathbf{Theorem 2}$.

Now, if $s \in B_\delta(p)$, then similarly we have that $s \in [p - delta, p + \delta]$. Since $f'(p) \neq 0$, and $g(p) = p$, we have that g is continuous and small near p. Thus, $\exists\ \delta \geq 0$ such that $|g'(x)| \leq \frac{1}{2} \leq \sigma < 1$ on $(p - \frac{\delta}{2}, p + \frac{\delta}{2})$. By the $Mean Value Theorem$, we have that:

$$\begin{aligned} |g'(x)| &= |\frac{g(p+\frac{\delta}{2}) - g(p-\frac{\delta}{2})}{(p+\frac{\delta}{2}) - (p-\frac{\delta}{2})}| \\ &= |\frac{g(p+\frac{\delta}{2}) - g(p-\frac{\delta}{2})}{2\frac{\delta}{2}}| \end{aligned}$$

$$\begin{aligned} \text{So,} \quad & |\delta g'(x)| = |g(p+\tfrac{\delta}{2}) - g(p-\tfrac{\delta}{2})| \\ \text{Therefore,} \quad & \delta\sigma \leq |g(p+\tfrac{\delta}{2}) - g(p-\tfrac{\delta}{2})| \end{aligned}$$

Since we have that $\delta \leq \frac{1}{2}$ and $\sigma \leq 1$, this tells us that initial guesses of points within δ of the fixed point will converge to p. □

B Eigenfunctions for the Ball

Here we detail the calculations necessary for finding and normalizing the eigenfunctions of the Laplacian in spherical coordinates.

B.1 Spherical Bessel Functions

Our research is greatly aided by previous work done with the Spherical Bessel Differential Equation, shown below:

$$\frac{d^2R(r)}{dr^2} + \frac{2}{r}\frac{dR(r)}{dr} + [k^2r^2 - n(n+1)]R(r) = 0 \tag{12}$$

We note that for $n = 0$ and $\lambda = k^2$, this is the same equation as the ODE we are concerned with solving. It is helpful, therefore, to use the already known solutions to the Spherical Bessel Differential Equation. We use the spherical bessel function for $n = 0$, which is simply the sinc function: $j_o(kr) = \frac{\sin(kr)}{kr}$. Note that k here symbolizes the k^{th} zero of this spherical bessel function. (mention why we don't use spherical bessel functions of the second kind?)

B.2 Eigenvalues and Eigenvectors

As previously mentioned, $\lambda = k^2$. Thus we know that the eigenvalues of our ODE are simply the squares of the zeros of the spherical bessel function. For zero Dirichlet boundary conditions, each value k is simply a positive integer (show?). The corresponding eigenfunctions are given by the equations $j_o(kr)$. We choose to normalize these eigenfunctions for ease of computations later.

Let $\psi_k(r) = \alpha\frac{\sin(kr)}{kr}$. If we take the norm of $\psi_k(r)$ and set it equal to one, we can solve for α to find our normalized eigenfunctions.

We first find the norm of a general function $f(r, \phi, \theta)$. Since we are dealing with only radial symmetry, we can actually treat this function as being only in terms of r. The norm, as we know, is the integral over the region in question of the function squared (we know, but should we explain why, in this case, we use this norm?). Since we are dealing in spherical coordinates, this integral becomes

$$||f(r)||^2 = \int_0^\pi \int_0^{2\pi} \int_0^\pi f^2(r) r^2 \sin(\phi) d\phi d\theta dr. \tag{13}$$

This integral is greatly simplified by the fact that our function f is only in terms of r. It is not hard to show that this simplifies down to

$$||f(r)||^2 = 4\pi \int_0^\pi f^2(r) r^2 dr.$$

We are now ready to normalize our eigenfunctions. Let $f(r) = \psi_k(r) = \alpha\frac{\sin(kr)}{kr}$. We plug this into the integral above and perform the following calculations:

$$\begin{aligned}
||\psi_k(r)||^2 &= 4\pi \int_0^\pi \alpha^2 \frac{\sin^2(kr)}{k^2 r^2} r^2 dr \\
&= 4\pi\alpha^2 \int_0^\pi \frac{\sin^2(kr)}{k^2} dr \\
&= 4\pi\alpha^2 \left[\frac{r}{2k^2} - \frac{1}{4k^3}\sin(2kr)\right]_0^\pi \\
&= 4\pi\alpha^2 \left[\frac{\pi}{2k^2}\right] \\
&= \frac{2\pi^2\alpha^2}{k^2}
\end{aligned}$$

Setting this equal to one, we solve for α.

$$\frac{2\pi^2\alpha^2}{k^2} = 1 \Rightarrow \alpha^2 = \frac{k^2}{2\pi^2} \Rightarrow \alpha = \frac{k}{\sqrt{2}\pi}.$$

Thus we conclude that are normalized eigenfunctions are given by the equation

$$\psi_k(r) = \alpha\frac{\sin(kr)}{kr} = \frac{k}{\sqrt{2}\pi}\frac{\sin(kr)}{kr} = \frac{\sin(kr)}{\sqrt{2}\pi r}. \tag{14}$$

B.3 Check

We check our work by seeing if the following equation holds:

$$\frac{1}{r^2}\frac{d}{dr}\left(r^2\frac{d}{dr}\frac{\sin(kr)}{\sqrt{2}\pi r}\right) = -k^2\psi_k(r).$$

Starting with the left, and working from the inside out, we reach our desired answer.

$$\begin{aligned}
\frac{1}{r^2}\frac{d}{dr}\left(r^2\frac{d}{dr}\frac{\sin(kr)}{\sqrt{2}\pi r}\right) &= \frac{1}{r^2}\frac{d}{dr}\left(r^2\frac{1}{\sqrt{2}\pi}\frac{rk\cos(kr)-\sin(kr)}{r^2}\right) \\
&= \frac{1}{r^2}\frac{d}{dr}\left(\frac{1}{\sqrt{2}\pi}\left(rk\cos(kr)-\sin(kr)\right)\right) \\
&= \frac{1}{r^2}\frac{1}{\sqrt{2}\pi}\frac{d}{dr}\left(rk\cos(kr)-\sin(kr)\right) \\
&= \frac{1}{r^2}\frac{1}{\sqrt{2}\pi}\big(-rk^2\sin(kr)+k\cos(kr)-k\cos(kr)\big) \\
&= \frac{1}{r^2}\frac{1}{\sqrt{2}\pi}\big(-rk^2\sin(kr)\big) = \frac{-k^2\sin(kr)}{\sqrt{2}\pi r} \\
&= -k^2\psi_k(r).
\end{aligned}$$

C Norms and Schematic Functions

In the course of this paper, we utilize various norms and other schematic functions to help visualize our infinite dimensional space (although estimated by a finite basis) in two dimensional space. The calculations of the following norms of $u(r)$ are as follows:

Assume,

$$u(r) = \sum_{k=1}^{\infty} a_k \psi_k(r)$$

where $\psi_k(r) = \frac{\sin(kr)}{\sqrt{2\pi} r}$, the basis functions for our problem. There are three norms that we are concerned with. $\|u\|_2^2$, the L_2 norm, $\|u\|_H^2$, the H_2 norm and $u(0)$. The tree norms are calculated slightly differently.

The L_2 norm is

$$\|u\|_2^2 = \int_\Omega u^2$$

The H norm is

$$\|u\|_H^2 = \int_\Omega \nabla u \nabla u$$

The $u(0)$ norm is

$$\lim_{r \to 0} u(r)$$

Now, these norms can certainly be simplified, and that is what is discussed in the following sections.

C.1 The L_2 norm

Now, let's calculate $\|u\|_2^2 = \int_\Omega u^2$

$$\begin{aligned}
\|u\|_2^2 &= \int_\Omega u^2 \\
&= \int_\Omega \left(\sum_{k=1}^{\infty} a_k \psi_k\right)\left(\sum_{l=1}^{\infty} a_l \psi_l\right) \\
&= 4\pi \sum_{l,k=1}^{\infty} a_k a_l \int_0^\pi \frac{1}{r^2} \psi_k \psi_l \\
&= 4\pi \sum_{l,k=1}^{\infty} a_k a_l \delta_{k,l} \\
&= 4\pi \sum_{k=1}^{\infty} a_k^2
\end{aligned}$$

With $\delta_{k,l}$ denoting the Kronecker delta. Finally,

C.2 The H_2 norm

Next, let us now calculate $\|u\|_H^2 = \int_\Omega \nabla u \nabla u$. Note, $\Delta\psi = \lambda\psi$

$$
\begin{aligned}
\|u\|_H^2 &= \int_\Omega \nabla u \nabla u \\
&= \int_\Omega u \Delta u && \textit{(Green's Second Identity)} \\
&= \int_\Omega \left(\sum_{l=1}^\infty a_l \psi_l\right)\left(\sum_{k=1}^\infty a_k k^2 \psi_k\right) \\
&= 4\pi \sum_{k,l=1}^\infty k^2 a_k a_l \int_0^\pi r^2 \psi_l \psi_k \\
&= 4\pi \sum_{k,l=1}^\infty k^2 a_k a_l \delta_{k,l} \\
&= 4\pi \sum_{k=1}^\infty k^2 a_k^2
\end{aligned}
$$

C.3 The $u(0)$ Schematic Function

Let us now calculate $u(0)$. Since $\frac{\sin(kr)}{\sqrt{2\pi}r}$ is undefined at $r = 0$, we must take the limit of this sum. Namely,

$$
\begin{aligned}
\lim_{r\to 0} \sum_{k=1}^\infty a_k \psi_k(r) &= \sum_{k=1}^\infty a_k \lim_{r\to 0} \frac{\sin(kr)}{\sqrt{2\pi}r} \\
&= \sum_{k=1}^\infty a_k \lim_{r\to 0} \frac{k\cos(kr)}{\sqrt{2\pi}} && \textit{(L'Hopital's Rule)} \\
&= \sum_{k=1}^\infty a_k \frac{k}{\sqrt{2\pi}} \\
&= \frac{1}{\sqrt{2\pi}} \sum_{k=1}^\infty a_k k
\end{aligned}
$$

D Sine and Cosine on the Interval using Midpoint and Trapezoid Rule

Theorem 1 (Exact Numerical Approximation of Sine and Cosine using Midpoint and Trapezoid Rule)**.** *Assume $k, \in \mathbb{Z}^+$ and $n \in \mathbb{Z}^+$.*

The Midpoint and Trapezoid rule give the exact answer for $\int_0^1 \cos(k\pi x)dx$ if and only if k is odd, or k is even and $n \nmid \frac{k}{2}$, and the exact answer for $\int_0^1 \sin(k\pi x)dx$ if and only if k is even.

Proof. Let us begin with the Trapezoid rule. Let us denote the Trapezoid rule for our integral $\int_0^1 e^{i\pi kx}dx$ as

$$(C + iS)_{n,k}^T.$$

We shall denote the midpoint rule similarly.

By the definition of the trapezoid rule, we expand $(C+iS)^T_{n,k}$ as follows.

$$(C+iS)^T_{n,k} = \frac{1}{n}\left(\frac{1}{2}e^0 + \left[\sum_{j=1}^{n-1}\left(e^{\frac{i\pi k}{n}}\right)^j\right] + \frac{1}{2}e^{i\pi k}\right) \tag{15}$$

The Midpoint rule is denoted

$$(C+iS)^M_{n,k} = \frac{1}{n}\sum_{j=0}^{n-1}\left(e^{ik\pi}\right)^{\frac{j+\frac{1}{2}}{n}} \tag{16}$$

For ease of notation, let us define $\omega = e^{\frac{i\pi k}{n}}$. This means that

$$\omega = e^{\frac{i\pi k}{n}} = \cos(k\pi x) + i\cdot\sin(k\pi x)$$

We can now rewrite (15) above as

$$(C+iS)^T_{n,k} = \frac{1}{n}\left(\frac{1}{2}e^0 + \left[\sum_{j=1}^{n-1}(\omega)^j\right] + \frac{1}{2}e^{i\pi k}\right)$$

and write (16) similarly.

To make any progress on the theorem, we must break into cases. Namely, k odd and k even.

Case 1

Now, let us first suppose k is odd. It may be helpful to note that $\omega^{\frac{1}{2}} = e^{\frac{i\pi k}{2n}}$ Now, we can compute the exact integral for ω, namely,

$$\int_0^1 \omega dx = \int_0^1 e^{i\pi kx}dx = \frac{2i}{\pi k}$$

for odd k. Expanding (15), noticing the first and last terms cancel, leaves us with

$$\begin{aligned}
\frac{1}{n}\sum_{j=1}^{n-1}\left(e^{\frac{i\pi k}{n}}\right)^j &= \frac{1}{n}\left(\frac{\omega-\omega^n}{1-\omega}\right)\\
&= \frac{1}{n}\left(\frac{\omega+1}{1-\omega}\right) = \frac{1}{n}\frac{(\omega+1)(1-\bar{\omega})}{1-\omega-\bar{\omega}+1}\\
&= \frac{1}{n}\frac{\left(e^{\frac{i\pi k}{n}}+1\right)\left(1-e^{\frac{-i\pi k}{n}}\right)}{2\left(1-\cos(\frac{k\pi}{n})\right)}\\
&= \frac{1}{n}\frac{2\cdot i\cdot\sin(\frac{k\pi}{n})}{2\left(1-\cos(\frac{k\pi}{n})\right)}\\
&= \frac{1}{n}\frac{i\cdot\sin(\frac{k\pi}{n})}{\left(1-\cos(\frac{k\pi}{n})\right)}
\end{aligned}$$

This number is purely imaginary. Clearly, this demonstrates that the cosine term is exact (namely, 0), and the sine, or rather the imaginary term in our expansion is not exact, though it approaches the exact answer as $n \to \infty$.

For the midpoint rule, and k odd, one would do the following:

$$\begin{aligned}
(C+iS)^M_{n,k} &= \frac{1}{n}\sum_{j=0}^{\infty}\left(e^{ik\pi}\right)^{\frac{j+\frac{1}{2}}{n}} \\
&= \frac{1}{n}\sum_{j=0}^{\infty}\left(\omega^{\frac{1}{2}}\omega^{j}\right) \\
&= \frac{1}{n}\left(\frac{\omega^{\frac{1}{2}}-\omega^{\frac{1}{2}}\omega^{n}}{1-\omega}\right) \\
&= \frac{1}{n}\cdot\frac{2\omega^{\frac{1}{2}}}{1-\omega} \\
&= \frac{2}{n}\cdot\frac{\omega^{\frac{1}{2}}\left(1-\bar{\omega}\right)}{\left(1-\omega\right)\left(1-\bar{\omega}\right)} \\
&= \frac{2}{n}\cdot\frac{\omega^{\frac{1}{2}}-\omega^{\frac{-1}{2}}}{1-\omega-\bar{\omega}+1} \\
&= \frac{2}{n}\frac{e^{\frac{ik\pi}{2n}}-e^{-\frac{ik\pi}{2n}}}{2-2\cos\left(\frac{k\pi}{n}\right)} \\
&= \frac{2}{n}\frac{2i\cdot\sin\left(\frac{k\pi}{2n}\right)}{2\left(1-\cos\left(\frac{k\pi}{n}\right)\right)}
\end{aligned}$$

So, finally, for the midpoint rule,

$$(C+iS)^M_{n,k} = \frac{2}{n}\frac{i\cdot\sin\left(\frac{k\pi}{2n}\right)}{1-\cos\left(\frac{k\pi}{n}\right)}$$

Thus, $C^M_{k,n}=0$ and $S^M_{k,n}=\frac{2}{n}\frac{\sin\left(\frac{k\pi}{2n}\right)}{1-\cos\left(\frac{k\pi}{n}\right)}$ So, we can conclude that Cosine is exact for any odd k, and we can say that Sine is not exact for any odd k. This simply means that, while $\frac{2}{n}\frac{\sin\left(\frac{k\pi}{2n}\right)}{1-\cos\left(\frac{k\pi}{n}\right)} \to \frac{2}{k\pi}$ as $n \to \infty$, which can be verified by a computer algebra system, or other for any finite value of n, $\frac{2}{n}\frac{\sin\left(\frac{k\pi}{2n}\right)}{1-\cos\left(\frac{k\pi}{n}\right)} \neq \frac{2}{k\pi}$

Case 2

Let us suppose k is even. To prove this, we must use exhaustive cases. Let us again denote the midpoint rule for ω as $(C+iS)^M_{k,n}$ and the trapezoid rule as $(C+iS)^T_{k,n}$.

This leaves us with two cases. The case when $n \mid \frac{k}{2}$ and when $n \nmid \frac{k}{2}$

To make this easy, let us compute the case where the function we are computing the integral of is

$$\omega_1 = e^{\frac{2i\pi k}{n}} = \cos(2k\pi x) + i\cdot\sin(2k\pi x).$$

Let us first look at the Trapezoid rule for $\int_0^1 \omega_1$. Now,

$$(C+iS)^T_{k,n} = \frac{1}{n}\sum_{j=0}^{n-1}\omega_1^j$$

and, in turn the midpoint rule is just shifted by an angle, namely ω. Thus,

$$(C+iS)^M_{k,n} = \frac{1}{n}\sum_{j=0}^{n-1}\omega_1^j\omega.$$

I will cover the case of the Midpoint rule, due to the fact that the trapezoid rule is sufficiently similar.

In the case of the Midpoint rule,

$$(C+iS)^M_{k,n} = \frac{1}{n}\sum_{j=0}^{n-1}\omega_1^j\omega = \frac{\omega}{n}\sum_{j=0}^{n-1}\omega_1^j.$$

Now, we break into two sub cases.

Let us now cover the midpoint rule when k is odd.

Case 1.a: $n \mid k$

This implies $\frac{k}{n} = d \in \mathbb{Z}^+$, which in turn implies that $\omega_1 = e^{2i\pi d}$. Thus, when we put this term into the sum,

$$\begin{aligned}\frac{\omega}{n}\sum_{j=0}^{n-1}\omega_1^j &= \frac{\omega}{n}\sum_{j=0}^{n-1}\left(e^{2i\pi d}\right)^j \\ &= \frac{\omega}{n}\sum_{j=0}^{n-1}1^j \\ &= \frac{\omega}{n}n = \omega = e^{\frac{i\pi k}{n}} = e^{i\pi d} = (-1)^d\end{aligned}$$

Recall, that we can integrate this function (ω_1) exactly, and the result is 0. Therefore, in this case, the Sine term is exact, but the Cosine term is not.

Case 1.b: $n \nmid k$

This implies,

$$\sum_{j=0}^{n-1}(e^{\frac{2i\pi k}{n}})^j = \frac{(e^{\frac{2i\pi k}{n}})^n - 1}{e^{\frac{2i\pi k}{n}} - 1} = 0$$

Therefore, we can conclude that both sine and cosine are exact in this case.

Now, we observe that for even k, one can rewrite the ω integral $\int_0^1 e^{\frac{i\pi k}{n}}dx$ as $\int_0^1 e^{\frac{2i\pi\frac{k}{2}}{n}}dx$ Through this result, we can conclude that $(S)^{M,T}_{k,n}$ for ω is exact for all even k, and $(C)^{M,T}_{k,n}$ is exact if and only if $n \nmid \frac{k}{2}$

Therefore, the Midpoint and Trapezoid rule give the exact answer for $\int_0^1 \cos(k\pi x)dx$ if and only if k is odd, or k is even and $n \nmid \frac{k}{2}$, and the exact answer for $\int_0^1 \sin(k\pi x)dx$ if and only if k is even.

□

E MATLAB Code

E.1 README

```
####### README File for MATLAB Code ##############
       Authors/Coders: Nate Veldt and Jake Clark
             Northern Arizona University REU 2012
###############################################

Overview of Files
*******************
main.m:
    %%Master Function%%
    - Initializes global variables for the suite of program
    - Calls other functions to follow branches and find bifurcations and ...
      solutions
    - Collects output from other functions and generates plots of the ...
      final results

RunTNGA.m:
    %%Branch Following Function%%
    - Called by main.m on with initial point and direction vector v to ...
      begin finding solutions and  bifurcations on a specific branch
    - Calls tGNGA.m for an initial guess pGuess
    - Computes new vector v and new guesses on a branch
    - Continues following a branch within a window of s values
    - Calls VectorSecant.m if a change in Morse Index is detected
    - Returns results to main.m

tGNGA.m:
    %%GNGA Constraint Function%%
    - Called by RunTGNGA
    - takes two solutions on a branch and finds a new solution p_new
    - returns p_new to RunTGNGA

VectorSecant.m:
    %%Bifurcation Point-Finding Function%%
    - Called by RunTGNGA when a change in morse index is found between ...
      two points on a  branch
    - Uses secant method to hone in on the exact location of the ...
      bifurcation point, where an    eigenvalue of the Hessian is zero

mGNGA:
    %%convergence/divergence of at a value s%%
    - Used principally for the critical exponent case and when the region ...
      is a ball
    - called by RunTGNGA when we are within a certain interval of s
    - Finds a solution to the discrete ODE using the GNGA for varying ...
      values of m
    - Plots u(0) versus m for one or several values of s, to show ...
      convergence or divergence
```

E.2 The MATLAB Main implementation

```
function main()
clc
close all
%screen size
scrsz = get(0,'ScreenSize');
set(0,'DefaultFigureWindowStyle','docked')
set(0,'DefaultTextFontSize',15)
set(0,'DefaultAxesFontSize',15)


%%%%%%  GLOBAL VARIABLES %%%%%%%%
%PDE initial condition variables
global B boundaryCond L tau dV exponent

%Boolean flags for algorithm-running options
global mGNGARun mGNGAlow mGNGAhigh plotHnorm  plotAnorm plotU0norm
global checkSolution plotMvsA plotMvsSteepD plotUinfinityNorm logPlots
global inflectionPlot plotAsymptotes sLowReassignV writeLaTeX
global writeAvals writeSM writeSU writesL2 writeInflection deleteB
mGNGARun = false;
mGNGAlow = .26;
mGNGAhigh = .275;
plotAsymptotes = false;
plotMvsSteepD = false;
sLowReassignV = true;
inflectionPlot = false;
writeLaTeX = true;
writeAvals = false;
writeSM = false;
writeSU = false;
writesL2 = false;
writeInflection = false;
deleteB = 0;
%%%% END GLOBAL VARIABLE SECTION %%%%%%%

folderName = 'runOutput';
rmdir(folderName,'s')
mkdir(folderName)

subFol = strcat(folderName,'/LaTeX/');
mkdir(subFol)

folderName2 = 'runData';
rmdir(folderName2,'s')
mkdir(folderName2)

fprintf('Welcome to the Main Method\n');

legendVec = {};
Ems = [];
Avals = [];
steepDVals = [];
```

```
inflecPts = [];
u0inf = [];

%{
    STEP ONE: INITIALIZE ALL VARIABLES
    This includes choosing: m, +/-u^3, +-u^5,(Neu or Dir conditions),
    L (length step), Δ, tolerance, etc.
%}
for m = 40
n = 2*m+1;

exponent = 3;
tau = 1;
L = pi;

%ASSIGNING BOUNDARY CONDITIONS:
dirichlet(m)
neumann(m)
dirichletBall(m)

a = zeros(m,1);
dx = L/n;
x = (((0:n-1) + 1/2)*dx)';
Δ = .1;
tol = 1e-9;
sLow = 0;
sHigh = 5;

dV = calcDV(x,dx);

%Initial v Value for the trivial branch parallel to s axis.
v = zeros(m+1,1);
B = calcB(n,m,x);

%Lower and higher bounds for eigenfind
sLowTriv = -2;
sTriv = -1;
sHighTriv = 5;%2;

%{
    STEP TWO: FIND WHERE MORSE INDEX CHANGES ON THE TRIVIAL BRANCH
    Use RunTGNGA
%}
v(1) = 1;
plotNum = 11;
[u0Vals,sVals, normAvals, u, BifPts, EigVecs,a]= ...
    RunTGNGA(x,true,m,n,.01,v,sTriv,a,sLowTriv,sHighTriv,tol);
PQueue = BifPts;
EigQueue = EigVecs;
BifurcationPtsList = PQueue;
fprintf('Done with trivial bifurcations \n\n');

counterB = 0;
while counterB < deleteB
       PQueue(:,1) = [];
```

```
            EigQueue(:,1) = [];
            counterB = counterB + 1;
    end
    %{
        STEP THREE: CALL THE TGNGA ON ALL BIFURCATION POINTS
        The tGNGA will continue to be called until there are no more points
        left to plot in the queue.
    %}
    while length(PQueue) > 0
        v = EigQueue(:,1);
        [u0Vals,sVals, normAvals, u, BifPts, EigVecs,uHnorms,a,uInf]= ...
            RunTGNGA(x,false,m,n,Δ,v,PQueue(m+1,1),PQueue(1:m,1),sLow,sHigh,tol);
        Ems = [Ems m]
        if inflectionPlot == true
            newS = [];
            newu0 = [];
            for i = 1:length(sVals)
                if (sVals(i) > .2 && sVals(i) < .36)
                    newS = [newS sVals(i)];
                    newu0 = [newu0 u0Vals(i)];
                end
            end
            [r,ru0] = inflectionFind(newS,newu0);
            inflecPts = [inflecPts r];
            u0inf = [u0inf ru0];

            [sMid, steepest] = steepestD(sVals, u0Vals,1);
            steepDVals = [steepDVals sMid];
            %Avals = [Avals a];
        end

        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        %{  BEGIN RESULTS PLOTTING SECTION   %}
        %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    ...

        %Adding new bifurcations branches to the queue to follow
        if BifPts ≠ false
            fprintf('Some new points to plot\n');
            PQueue = [PQueue BifPts];
            EigQueue = [EigQueue EigVecs];
            BifurcationPtsList = [BifurcationPtsList BifPts];
        end

        %Remove branch just plotted from the queue
        PQueue(:,1) = [];
        EigQueue(:,1) = [];
    end
    end
```

E.3 The MATLAB RunTGNGA implementation

```
% RUNTGNGA
% Function uses Newton's Method to follow a branch of a
% bifurcation graph
%
% Inputs:  x - Interval/region partitioned into grid points
%          onTrivBranch - Boolean variable true if on trivial branch
%          m - number of modes
%          n - number of grod points
%          ∆ - Newton's Method Delta
%          v - direction vector
%          s - Starting point for calculations
%          a - coefficients of 'solution' u, initial guess
%          sLow - s Window Lower bound
%          sHigh - s Window upper bound
%          tol - Newton's Method Tolerance
% Outputs: u0Vals - u(0) values for our plots
%          sVals - list of values of S to plot on the bifurcation diagram
%          normAvals - list of the values of normA for the bifurcation plot
%          BifPts - a list of the pValues where MI changes
%          EigVecs - a corresponding list of eigenvectors corresponding ...
    to the
%                   change in MI at the BifPts.
% Algorithm Outline:
%         Called with initial point and direction vector v
%         to begin finding solutions and    bifurcations on a specific branch
%         Calls tGNGA.m for an initial guess pGuess
%         Computes new vector v and new guesses on a branch
%         Continues following a branch within a window of s values
%         Calls VectorSecant.m if a change in Morse Index is detected
%         Returns results

function [u0Vals,sVals, normAvals, uNew, BP, EV, uHnorms,a,uInf] =  ...
    RunTGNGA(x,onTrivBranch,m,n,∆,v,s,a,sLow,sHigh,tol)
global B mGNGARun mGNGAlow mGNGAhigh sLowReassignV

fprintf('Welcome to RunTGNGA\n');

xVal = [s];
yVal = [norm(a)];
u0Vals = [uNaught(a)];
uHnorms = [hNorm(a)];
uInf = [max(abs(B*a))];
pCurrent =[a;s];

if onTrivBranch == true
    OldMorse = 0;
else
    OldMorse = -100;
end

maxIts = 20;
callSecant = 0;
BifPts = zeros(m+1,1);
EigVecs = zeros(m+1,1);
column = 1;
```

```
returnBifs = false;
its = 0;
flag = 0;
flag2 = 0;

while (s < sHigh && s > sLow)
    its = its + 1;
    sOld = s;
    pGuess = pCurrent + (Δ * v);

    aOld = pGuess(1:m);

    [pNew, newMorse, Iters, NewtonFailed] = tGNGA(pGuess,v,m,tol,maxIts);

     if sLowReassignV == true
         if(pNew(m+1) < sLow)
             fprintf('Lower bound (sLow) reached.\n');
             pNew(m+1) = sLow;
             v = zeros(m+1,1);
             v(m+1,1) = 1;
             [pNew, newMorse, Iters, NewtonFailed] = ...
                 tGNGA(pNew,v,m,tol,maxIts);
             pNew;
         end
     end

    a = pNew(1:m,:);
    s = pNew(m+1,:);

    if mGNGARun == true
        if (mGNGAhigh > sOld && sOld > mGNGAlow && onTrivBranch == false ...
            && flag == 0)
            plotter = 3;
            fprintf('s is now in desired interval. It is time to run ...
                mGNGA.\n');
            [sVals, u0] = mGNGA(aOld,plotter);
            flag = 1;
            figure(137)
            plot(sVals,u0)
            title 'Try to fit something to this'
            xlabel('Convergent s values')
            ylabel('u(0)')
        end
    end

    %decreasingMItest(OldMorse,newMorse);
    if newMorse ≠ OldMorse
        if (its > 1 || onTrivBranch == true)
            fprintf('Morse index changed from %d to %d\n',OldMorse,newMorse);
            callSecant = 1;
            returnBifs = true;
        end
        OldMorse = newMorse;
    end

    pOld = pCurrent;
    pCurrent = pNew;
    v = (pCurrent - pOld)/(norm(pCurrent - pOld));

    %Calling the Secant Method, finding bifurcations
    if callSecant == 1
        hOld = calcH(m,pOld(m+1),B*pOld(1:m));
        hCur = calcH(m,pCurrent(m+1),B*pCurrent(1:m));

        [pStar, eVector] = VectorSecant(pOld,pCurrent,hOld,hCur,m,n,1e-9);

        eVector(m+1) = 0;
        BifPts(:,column) = pStar;
        EigVecs(:,column) = eVector';
        column = column + 1;
        callSecant = 0;
    end

    xVal = [xVal s];
    yVal = [yVal norm(a)];
    u0Vals = [u0Vals uNaught(a)];
    uHnorms = [uHnorms hNorm(a)];
    uInf = [uInf max(abs(B*a))];

end

%Vectors and Values to Return
sVals = xVal;
normAvals = yVal;
uNew = B*a;
if returnBifs == true
    BP = BifPts;
    EV = EigVecs;
else
    BP = false;
    EV = false;
end


end

%Various functions used in this file
function decreasingMItest(OldMorse,newMorse)
    if newMorse < OldMorse
        fprintf('Decreasing morse index.\n');
    end
end
```

E.4 The MATLAB tGNGA implementation

```
% TGNGA
% Function uses Newton's Method to compute a point on a branch of a
% bifurcation graph
%
% Inputs:  pGuess - the initial guess, not located on the branch
%          v - the direction vector we use to update pGuess
%          m - the number of rows and columns of the Hessian
%          B - the n x m matrix of eigenfunctions
%          lam - the vector of eigenvalues
%          gtol - the tolerance we use for calculating gTwid
%          maxIts - the maximum number of iterations we will calculate
% Outputs: pNew - the new point [a;s] on the branch that we found
%          oldMI - the signature of the Hessian before newton's method
%          newMI - the signature of the Hessian after our calculations
%          Iters - the number of iteratins Newton's Method takes to
%                  converge (or fail through reaching maxIts)
% Algorithm Outline:
%          While we haven't exceeded maxIts, we calculate h and g.
%          We append v and -a to form hTwid and gTwid.
%          We solve for Chi to get the Newton step, and with it we
%          update pGuess, u, and g. If norm(g) is within a certain
%          tolerance, we stop looping and return the latest pGuess as pNew.

function [ pNew, newMI, Iters, NewtonFailed ] = tGNGA(pGuess,v,m,gtol,maxIts)
global B dV lam

its = 0;
a = pGuess(1:m);
s = pGuess(1+m);
notDone = true;
failed = false;

while notDone
    its = its + 1;
    u = B*a;
    g = a.*(lam-s) - B' * (dV.*f(u));
    h = calcH(m,s,u);
    hTwid = [horzcat(h,-a);v'];
    invHTwid = pinv(hTwid);
    gTwid = [g;0];
    ChiTwid = invHTwid*gTwid;
    pGuess = pGuess - ChiTwid;
    a = pGuess(1:m); %update a
    s = pGuess(1+m); %update s

    if its > maxIts %test for too many iterations
        fprintf('\nDANGER tGNGA: exited early. Poor pNew returned.\n');
        notDone = false;
        failed = true;
    end

    if norm(gTwid) < gtol %test for convergence
        notDone = false;
    end
end

newMI = sig(h); % Calculate new Morse Index of h
```

```
NewtonFailed = failed;
pNew = pGuess; %Assign pGuess to pNew, since constraint is satisfied
Iters = its;

end
```

E.5 The MATLAB mGNGA implementation

```
% MGNGA
% Function runs the GNGA on a specific p point of our bifurcation diagram
% for varying values of m. We can run mGNGA for multiple values of s as
% well.
%
% Inputs:  OriginalA - initial guess a_0 found by the tGNGA in a certain
%                         window of the bifurcation diagram.
%          plotNum - used to number the plots in mGNGA
% Outputs: sVals - list of different values of s used
%          u0 - final u0 values corresponding to the sVals vector
% Algorithm Outline:
%          Finds a solution to the discrete ODE using the GNGA
%      for varying values of m
%          Plots u(0) versus m for one or several values of s,
%          to show convergence or divergence

function [sVals, u0] = mGNGA(OriginalA,plotNum)
global L mGNGAlow mGNGAhigh
error=1e-9;
maxIts = 40;

sVals = [];
u0 = []; %final u0 values for differing s values that converge (s < s*)

u0vals = []; %u0 values for differeing m values, which might diverge
mVals = [];
normAvals = [];
uInfvals = [];

step = .002;
for s = mGNGAlow:step:mGNGAhigh
    NewA = OriginalA;
    u0vals = [];
    mVals = [];
    normAvals = [];
    uInfvals = [];
    fprintf('s is now %f \n',s);
    for m = 20:10:400
        n = 3*m;
        its = 0;
        lam = ((1:m).^2 * pi^2/(L^2))';
        dx = L/n;
        x = (((0:n-1) + 1/2)*dx)';
        dV = calcDV(x,dx);

        BNEW = calcB(n,m,x);

        %Make a correct size
        a = zeros(m,1);
        if size(NewA) < m
            for ii = 1:size(NewA)
                a(ii) = NewA(ii);
            end
        else
            for jj = 1:m
                a(jj) = NewA(jj);
            end
        end

        u = BNEW*a;
        g = a.*(lam-s) - BNEW' * (dV.*f(u));
        h = zeros(m,m);

        while sqrt(dot(g,g)) > error
            its = its+1;
            h = calcHMGNGA(m,s,lam,BNEW,u,dV);
            chi = h\g;
            a = a - chi;
            u = BNEW*a;
            g = a.*(lam-s) - BNEW' * (dV.*f(u));

            if its>maxIts
                break
            end
        end
        if m > 10
            u0vals = [u0vals abs(uNaught(a))];
            mVals = [mVals m];
            normAvals = [normAvals norm(a)];
            uInfvals = [uInfvals max(abs(u))];
            fprintf('For m = %d, u(0) = %f \n',m,uNaught(a));
            fprintf('For m = %d, max(abs(u)) = %f \n',m,max(abs(u)));
        end
        NewA = a;
    end

    if m > 70
        figure(12)
        figN = plot(mVals, u0vals);
        set(gcf, 'PaperPosition', [0 0 7 5]); %Position plot at left hand ...
            corner with width 7 and height 5.
        set(gcf, 'PaperSize', [7 5]); %Set the paper to have width 7 and ...
            height 5.
        axis([80 400 0 50])
        %text(250,27,'\uparrow s = 2.7','HorizontalAlignment','right')
        %text(200,38,'s = 2.4 \downarrow','HorizontalAlignment','right')
        axis 'auto y'
        xlabel('m');
        ylabel('u(0)');
        hold on

        saveas(figN,'runOutput/plot_m_vs_u0.pdf')

        figure(13)
        figN2 = plot(mVals, uInfvals);
        set(gcf, 'PaperPosition', [0 0 7 5]); %Position plot at left hand ...
            corner with width 7 and height 5.
        set(gcf, 'PaperSize', [7 5]); %Set the paper to have width 7 and ...
            height 5.
        %axis([80 400 0 50])
        %axis 'auto y'
        xlabel('m');
        ylabel('max(abs(u))');
        hold on
        saveas(figN2,'runOutput/plot_m_vs_uInf.pdf');

    end

    fprintf('For s = %f, u(0) equals %f \n',s,uNaught(a));
    sVals = [sVals s];
    u0 = [u0 uNaught(a)];
end
fprintf('Exiting mGNGA\n');
end

%Custom H calculaton for mGNGA
function h = calcHMGNGA(m,s,lam,B,u,dV)
h = zeros(m,m);
for i = 1:m
    secondPart = [B'*(B(:,i).*(fPrime(u).*dV))];
    h(i,i:m) = ((lam(i) - s)*kronDel(i,m) - secondPart(i:m))';
end
h = h + h' - diag(diag(h));
end

%The kronecker Δ vector the upper triangular part of each row of the
%Hessian
function y2 = kronDel(i,m)
del = zeros(m+1-i,1);
del(1) = 1;
y2 = del;
end
```