THE PENNSYLVANIA STATE UNIVERSITY
SCHREYER HONORS COLLEGE

DEPARTMENT OF MATHEMATICS

USING THE FINITE ELEMENT METHOD FOR SOLVING THE SCHRÖDINGER EQUATION

IGNACIO SOFO
Spring 2011

A thesis
submitted in partial fulfillment
of the requirements
for baccalaureate degrees
in Mathematics and Philosophy
with honors in Mathematics

Reviewed and approved* by the following:

Victor Nistor
Professor of Mathematics
Thesis Supervisor

Svetlana Katok
Professor of Mathematics
Honors Adviser

*Signatures are on file in the Schreyer Honors College.

Abstract

The Finite Elements Method was first presented in an address to the American Mathematical Society by Richard L. Courant. The method was presented as part of a two-page appendix where he showed how piecewise-linear approximations on a set of elements could be used to solve partial differential equations[6]. The purpose of this paper is to look at solutions of a 1-dimensional Schrödinger equation using the Finite Element Method. The method simplifies the differential equation into an eigenvalue problem where numerical techniques can be used to acquire a solution. In particular, this paper will be focusing on the two main sources of error that accompany this numerical method: the size of the elements, and the overall size of our mesh.

Contents

Chapter 1

Introduction

The Finite Element Method (FEM) serves as a way to approximate solutions to differential equations. The idea behind the method is to use basis functions to transform the differential equation into a matrix problem. This process of discretization occurs on a domain which is then cut up into a finite number of elements [5] . In this paper we will be looking at the one-dimensional case, thus our elements are merely segments of a line. However, this method can be extended to higher dimensions where the elements take the form of triangles or other two dimensional shapes and the basis functions are accordingly extended to this space.

We will be applying this to a special equation which is known as the Schrödinger equation. This equation was formulated by Austrian physicist Erwin Schrödinger in an attempt to find a wave equation for the electron. By using this equation he was treating the hydrogen atom's electron as a wave that moved trough a potential [7]. We will notice that the application of the Finite Element Method to this equation results in an eigenvalue problem. Before we delve into these types of problems, let us consider a 1D Poisson

equation,

$$\begin{aligned} -u''(x) &= f(x) \;\; x \in (0,1) \\ u(0) &= 0 \\ u'(1) &= 0 \end{aligned} \tag{1.0.1}$$

In this problem, we are using a mixture of Dirichlet and Nuemann boundary conditions. We can now consider a function v(x) s.t. $v(0) = 0$. With this we can now transform equation (1.0.1) by defining

$$< f, v > \;\; := \int_0^1 f(x)v(x)dx$$

By using integration by parts as well as our boundary conditions, we get

$$\begin{aligned} \int_0^1 f(x)v(x)dx &= -\int_0^1 u''(x)v(x)dx \\ &= \int_0^1 u'(x)v'(x)dx - \Big[u'(x)v'(x)\Big]_0^1 \\ &= \int_0^1 u'(x)v'(x)dx \\ &:= \alpha(u,v). \end{aligned}$$

In this manner, (1.0.1) can be transformed into what we will call the weak form:

$$u(x) \in \mathcal{V} \;\; \alpha(u,v) =< f, v > \;\; \forall v \in \mathcal{V} \tag{1.0.2}$$

Where we take $\mathcal{V}$ to be:

$$\mathcal{V} = \{v \in C([0,1]) \mid \int |v'|^2 \, dx < \infty \text{ and } v(0) = 0\}$$

In order to show the equivalence of (1.0.1) and (1.0.2), we need to show that the equivalence also works in the other direction.

"$\Leftarrow$" We know

$$< f, v >= \alpha(u, v) \quad \forall v \in \mathcal{V}$$

$$= \int_0^1 u'(x)v'(x)dx = \Big[u'(x)v(x)\Big]_0^1 - \int_0^1 u''(x)v(x)dx$$

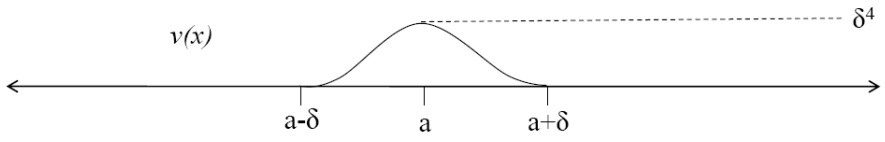
if we further assume that $v(1) = 0$, then we end up with the following

$$< f, v >= -\int_0^1 u''(x)v(x)dx =< -u'', v >$$

$$< u'' + f, v >= 0$$

Suppose there exists a function $h(x) = u''(x) + f(x) \neq 0$. There exists a point a such that $h(a) \neq 0$. Without loss of generality, and due to the assumption that $h(x)$ is continuous, we can say that $h(x)$ is positive on some interval δ around a. Recall that in our assumption, (1.0.2), we assumed $v \in \mathcal{V}$. Pick the following $v(x)$:

$$v(x) = \begin{cases} (x - (a+\delta))^2(x - (a-\delta))^2 & a - \delta \leq x \leq a + \delta \\ 0 & otherwise \end{cases}$$



However, we then arrive at the following contradiction

$$< u'' + f, v >= \int_0^1 (u''(x) + f(x))v(x)dx > 0$$

We thus have the first part of (1.0.1). All we have left to show is that $u(0) = 0$ and $u'(1) = 0$. The first part is a consequence of the fact that $u(x) \in \mathcal{V}$. To show that $u'(1) = 0$ we can go back to the weak form

$$\begin{aligned} < f, v >&= \Big[u'(x)v(x)\Big]_0^1 - \int_0^1 u''(x)v(x)dx \\ &= u'(1)v(1) + \int_0^1 (-u''(x))v(x)dx \\ &= u'(1)v(1) + < f, v > \end{aligned}$$

Thus $u'(1)v(1)$ must be 0. Since our restriction on v is merely that it must belong to $\mathcal{V}$ then $u'(1) = 0$. This concludes our proof and we can now say that (1.0.1) is equivalent to (1.0.2) for all $u \in \mathcal{V}$.

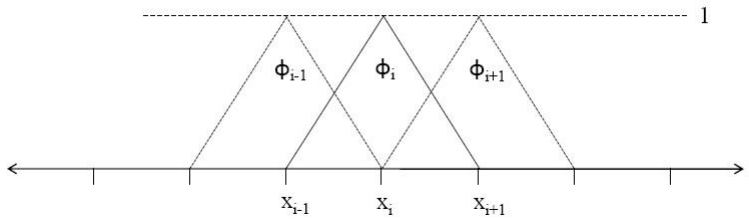
Chapter 2

Finite Elements

Still looking at the problem defined in (1.0.1), we will now define a mesh we will be working on. We take nodes x_i for $i = 0, 1, ...n, n+1$ and each element will have width h such that $h_i = x_i - x_{i-1}$ for $i = 1, ...n + 1$.

At this point we will be defining basis functions that will serve to construct our solution. For the purpose of numerical integration we will be using piecewise-linear functions ϕ. We will define these functions as follows:

$$\phi_i(x) = \begin{cases} \frac{x - x_{i-1}}{x_i - x_{i-1}} & x \in (x_{i-1}, x_i) \\ \frac{x_{i+1} - x}{x_{i+1} - x_i} & x \in (x_i, x_{i+1}) \quad for\ i = 1, ..., n \\ 0 & otherwise \end{cases}$$



We can now define a subspace $\mathcal{S} \subset \mathcal{V}$ where

$$\mathcal{S}_n = \{f : [0,1] \to \mathbb{R} | f \text{ is linear on } [\frac{i}{n}, \frac{i+1}{n}] \text{for } i = 1, ..., n-1 \text{and continuous}\}$$

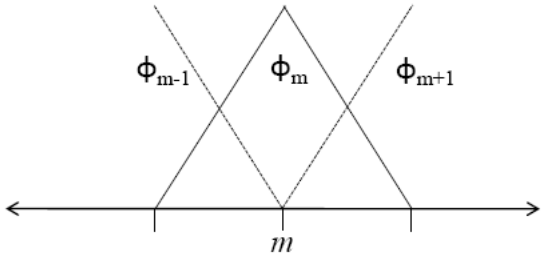
Note that this is the space of piecewise linear functions on a mesh with n nodes. More importantly, we can now use our defined $\phi' s$ as a basis of the space $\mathcal{S}_n$.

Definition 1 *We call a set of elements $e_1, e_2, \ldots, e_n$ a **basis** of A, if two things hold:*

1. ***Linear Independence*** *If $c_1 e_1 + c_2 e_2 + \ldots + c_n e_n = 0$, then $c_0 = c_1 = \ldots = c_n = 0$*
2. ***Span*** *$\forall a \in A$, $a = \sum_{i=1}^{n} c_i e_i$*

In order to show that these basis functions that we have defined are a basis of $\mathcal{S}_n$, we will need to show the following two things:

1. Linear independence of $\phi' s$. If $\sum_{i=1}^{n} c_i \phi_i = 0$ then $c_i = 0$ for $i = 1, ..., n$. Let us look at one specific node in our mesh:



The only ϕ that attains a value other than 0 at this node m is ϕ_m. Since the sum must be 0 at all points on our mesh, we know that c_m must be zero. We can apply this logic to all points on our mesh to show that $c_i = 0$ for $i = 1, ..., n$.

2. Next, we need to prove that for every $f \in \mathcal{S}_n$, then there exist $\phi_1, \phi_2, ..., \phi_n$ such that $f = \sum_{i=1}^{n} c_i \phi_i$. For this we will again use the fact we used in proving linearity. For any $x_m \in [x_1, x_n]$ we have:

$$f(x_m) = \sum_{i=1}^{n} c_i \phi_i(x_m)$$

Since for $i \neq m$, $\phi_i = 0$ we can simplify this to

$$f(x_m) = c_m \phi_m(x_m) = c_m$$

Thus we can construct a linear combination of our $\phi' s$ to build any function in $\mathcal{S}_n$.

We can now reconsider(1.0.2) with $u_s \in \mathcal{S}$ such that we have

$$\alpha(u_s, v_s) =< f, v_s > \text{for all } v \in \mathcal{S} \tag{2.0.1}$$

then we can say $u_s = \sum_{i=1}^{n} c_i \phi_i$. Thus making (1.0.2) into

$$\alpha(\sum_{i=1}^{n} c_i \phi_i, v) =< f, v > \forall v \in \mathcal{S} \tag{2.0.2}$$

We can now pick $v = \phi_j$ for $j = 1, ..., n$

$$\alpha(\sum_{i=1}^{n} c_i \phi_i, \phi_j) =< f, \phi_j > \text{for } j = 1, ..., n \tag{2.0.3}$$

$$\sum_{i=1}^{n} c_i \alpha(\phi_i, \phi_j) =< f, \phi_j > \text{for } j = 1, ..., n \tag{2.0.4}$$

The expression in (2.0.4) becomes the system of equations which we have to

solve to obtain f

$$Ac_i = f_i \tag{2.0.5}$$

Where $A = \alpha(\phi_i, \phi_j)$ and $f_i =< f, \phi_j >$.

Note that the solution to this differential equation then boils down to solving a linear system of equations. This fact helps guarantee a unique solution to our problem.

Chapter 3

Introduction of Eigenvalue Problem

We can repeat the process above but this time using $f = \lambda u$. When doing this, we start with

$$\begin{aligned} -u'' &= \lambda u && (3.0.1)\\ -\int_0^1 u''(x)v(x)dx &= \lambda \int_0^1 u(x)v(x)dx && (3.0.2)\\ \int_0^1 u'(x)v'(x)dx - \Big[u'(x)v'(x)\Big]_0^1 &= \lambda \int_0^1 u(x)v(x)dx && (3.0.3)\\ \alpha(u,v) &= \lambda < u, v > && (3.0.4) \end{aligned}$$

Here, we can see the relation between this last equation and what we had in (1.0.2). As we did with the simple 1D poisson equation we will now use $u_s = \sum_{i=1}^{n} c_i\phi_i$ as well as pick $v = \phi_j$ for $j = 1, ..., n$.

$$\begin{aligned} \alpha(\sum_{i=1}^{n} c_i\phi_i, \phi_j) &= \lambda < \sum_{i=1}^{n} c_i\phi_i, \phi_j > \text{for } j = 1, ..., n && (3.0.5)\\ \sum_{i=1}^{n} c_i\alpha(\phi_i, \phi_j) &= \lambda \sum_{i=1}^{n} c_i < \phi_i, \phi_j > \text{for } j = 1, ..., n && (3.0.6) \end{aligned}$$

Define: $A = \alpha(\phi_i, \phi_j)$, $M =< \phi_i, \phi_j >$. We get the following eigenvalue problem:

$$Ac = \lambda Mc \tag{3.0.7}$$

3.1 Schödinger Equation

When considering the Schrödinger equation we have to bring in one more term into consideration: the potential. In the remainder of this paper we will be using the potential x^2 instead of the usual $\frac{1}{x}$ and thus we will in a sense be dealing with the Schrödinger equation for a harmonic oscillator[4]. Our equation is of the form:

$$-u''(x) + x^2 u(x) = \lambda u(x) \tag{3.1.1}$$

We know one of the solutions to (3.1.1) is of the form $ce^{-\alpha x^2}$such that

$$\begin{gathered}
-(ce^{-\alpha x^2})'' + x^2 ce^{-\alpha x^2} = \lambda ce^{-\alpha x^2} \\
-(ce^{-\alpha x^2}(-2\alpha x))' + x^2 ce^{-\alpha x^2} = \lambda ce^{-\alpha x^2} \\
2\alpha c(e^{-\alpha x^2} + xce^{-\alpha x^2}(-\alpha 2x)) + x^2 ce^{-\alpha x^2} = \lambda ce^{-\alpha x^2} \\
2\alpha ce^{-\alpha x^2} - 4\alpha^2 x^2 ce^{-\alpha x^2} + x^2 ce^{-\alpha x^2} = \lambda ce^{-\alpha x^2} \\
2\alpha - 4\alpha^2 x^2 + x^2 = \lambda \\
\alpha = \tfrac{1}{2} \; and \; \lambda = 1
\end{gathered}$$

This forms the lowest energy solution for equation (3.1.1). Higher eigenvalue solutions will be of the form $ce^{-\alpha x^2} H_n(x)$.Where $H_n(x)'s$ are the Hermite polynomials [3].

In order to work with this problem we will change the above procedure

to look as follows:

$$\begin{aligned}
-u''(x) + x^2u(x) &= \lambda u(x) \\
-\int_0^1 u''(x)v(x)dx + \int_0^1 x^2u(x)v(x)dx &= \lambda \int_0^1 u(x)v(x)dx \\
\int_0^1 u'(x)v'(x)dx - \Big[u'(x)v(x)\Big]_0^1 + \int_0^1 x^2u(x)v(x)dx &= \lambda \int_0^1 u(x)v(x)dx \\
\alpha(u,v) + < x^2u, v > &= \lambda < u, v >
\end{aligned}$$

Pick $u = \sum c_i\phi_i$ and $v = \phi_j \; for \forall j = 1, ..., n$

$$\sum c_i\alpha(\phi_i, \phi_j) + \sum c_i < x^2\phi_i, \phi_j >= \lambda \sum c_i < \phi_i, \phi_j > for \forall j$$

This will then simplify into the following eigenvalue problem

$$(A + P)c = \lambda Mc$$

where $P = \sum_{i=1}^{n} < x^2\phi_i, \phi_j >$for each j.

3.2 Computing A, P, and M

These three matrices will be tri-diagonal due to the nature of the basis functions ϕ that we have chosen. We will begin by finding A (also called the stiffness matrix[1]), recall that we had defined it as

$$A = \alpha(\phi_i, \phi_j) \text{ for } i, j = 1, ..., n$$

Note: for a mesh with n+2 nodes we will have n $\phi's$ and thus our matrices will be n x n.

We can understand each matrix by understanding how they each break

down.

$$A = \begin{bmatrix} \int_{x_0}^{x_{n+1}} \phi_1'(x)\phi_1'(x)dx & \int_{x_0}^{x_{n+1}} \phi_1'(x)\phi_2'(x)dx & \cdots & \int_{x_0}^{x_{n+1}} \phi_1'(x)\phi_n'(x)dx \\ \int_{x_0}^{x_{n+1}} \phi_2'(x)\phi_1'(x)dx & \int_{x_0}^{x_{n+1}} \phi_2'(x)\phi_2'(x)dx & & \\ \vdots & & \ddots & \\ \int_{x_0}^{x_{n+1}} \phi_n'(x)\phi_1'(x)dx & & & \int_{x_0}^{x_{n+1}} \phi_n'(x)\phi_n'(x)dx \end{bmatrix}$$

By our definition we can simplify these integrals. We know that ϕ_k is only non-zero on the interval (x_{k-1}, x_{k+1}). Therefore, when we have $\int_{x_0}^{x_{n+1}} \phi_i(x)\phi_j(x)dx$ the only i and j for which this will be non-zero are given by $|i-j| \leq 1$. This results in the following tri-diagonal symmetric matrix (symmetry is trivially achieved by commutativity).

$$A = \begin{bmatrix} \int_{x_0}^{x_{n+1}} \phi_1'\phi_1'dx & \int_{x_0}^{x_{n+1}} \phi_1'\phi_2'dx & 0 & \cdots & 0 \\ \int_{x_0}^{x_{n+1}} \phi_2'\phi_1'dx & \int_{x_0}^{x_{n+1}} \phi_2'\phi_2'dx & \int_{x_0}^{x_{n+1}} \phi_2'\phi_3'dx & 0 & \\ 0 & \int_{x_0}^{x_{n+1}} \phi_3'\phi_2'dx & \int_{x_0}^{x_{n+1}} \phi_3'\phi_3'dx & \ddots & 0 \\ \vdots & 0 & \ddots & \ddots & \int_{x_0}^{x_{n+1}} \phi_{n-1}'\phi_n'dx \\ 0 & & 0 & \int_{x_0}^{x_{n+1}} \phi_n'\phi_{n-1}'dx & \int_{x_0}^{x_{n+1}} \phi_n'\phi_n'dx \end{bmatrix}$$

We can go even further to simplify this matrix. We will again use the very specific areas where our basis functions are defined to change the bounds of our integrals. Since our basis functions are defined piece-wise, it will be easier for computational purposes to integrate over each individual element.

$$A = \begin{bmatrix} \int_{x_0}^{x_1} (\phi_1')^2 dx + \int_{x_1}^{x_2} (\phi_1')^2 dx & \int_{x_1}^{x_2} \phi_1' \phi_2' dx & \cdots & 0 \\ \int_{x_1}^{x_2} \phi_2' \phi_1' dx & \int_{x_1}^{x_2} (\phi_2')^2 dx + \int_{x_2}^{x_3} (\phi_2')^2 dx & \ddots & 0 \\ \vdots & \ddots & \ddots & \int_{x_{n-1}}^{x_n} \phi_{n-1}' \phi_n' dx \\ 0 & 0 & \int_{x_{n-1}}^{x_n} \phi_n' \phi_{n-1}' dx & \int_{x_{n-1}}^{x_n} (\phi_n')^2 dx + \int_{x_n}^{x_{n+1}} (\phi_n')^2 dx \end{bmatrix}$$

Our choice of linear basis functions makes these integrals trivial. We essentially have two cases:

1. The integrals on the diagonal:

$$\int_{x_{k-1}}^{x_k} (\phi_k')^2 dx + \int_{x_k}^{x_{k+1}} (\phi_k')^2 dx = \int_{x_{k-1}}^{x_k} (1)^2 dx + \int_{x_k}^{x_{k+1}} (-1)^2 dx$$

   Recall that when we defined our mesh, we defined $h_k = x_k - x_{k-1}$. We are then left with:

$$\int_{x_{k-1}}^{x_k} 1 dx + \int_{x_k}^{x_{k+1}} 1 dx = h_k + h_{k+1}$$

2. The integrals above and below the diagonal:

$$\begin{aligned} \int_{x_k}^{x_{k+1}} \phi_k' \phi_{k+1}' dx &= \int_{x_k}^{x_{k+1}} (-1)(1) dx \\ &= -(h_{k+1}) \end{aligned}$$

This allows us to simplify even further our matrix:

$$A = \begin{bmatrix} h_1 + h_2 & -h_2 & \cdots & 0 \\ -h_2 & h_2 + h_3 & \ddots & 0 \\ \vdots & \ddots & \ddots & -h_n \\ 0 & 0 & -h_n & h_n + h_{n+1} \end{bmatrix}$$

When finding A we were merely integrating the constants -1 and 1. Computing P and M will be more complex. In the case of M (also called the mass matrix [1]), we will be looking at integrals of the form $\int_{x_0}^{x_{n+1}} \phi_i \phi_j dx$. For our potential matrix P, the terms will be of the form $\int_{x_0}^{x_{n+1}} x^2 \phi_j \phi_i dx$. After using simplifications such as those used to find A, we obtain the following:

$$M = \begin{bmatrix} \int_{x_0}^{x_1} (\phi_1)^2 dx + \int_{x_1}^{x_2} (\phi_1)^2 dx & \int_{x_1}^{x_2} \phi_1 \phi_2 dx & \cdots & 0 \\ \int_{x_1}^{x_2} \phi_2 \phi_1 dx & \int_{x_1}^{x_2} (\phi_2)^2 dx + \int_{x_2}^{x_3} (\phi_2)^2 dx & \ddots & 0 \\ \vdots & \ddots & \ddots & \int_{x_{n-1}}^{x_n} \phi_{n-1} \phi_n dx \\ 0 & 0 & \int_{x_{n-1}}^{x_n} \phi_n \phi_{n-1} dx & \int_{x_{n-1}}^{x_n} (\phi_n)^2 dx + \int_{x_n}^{x_{n+1}} (\phi_n)^2 dx \end{bmatrix}$$

$$P = \begin{bmatrix} \int_{x_0}^{x_1} x^2 (\phi_1)^2 + \int_{x_1}^{x_2} x^2 (\phi_1)^2 & \int_{x_1}^{x_2} x^2 \phi_1 \phi_2 & \cdots & 0 \\ \int_{x_1}^{x_2} x^2 \phi_2 \phi_1 & \int_{x_1}^{x_2} x^2 (\phi_2)^2 + \int_{x_2}^{x_3} x^2 (\phi_2)^2 & \ddots & 0 \\ \vdots & \ddots & \ddots & \int_{x_{n-1}}^{x_n} x^2 \phi_{n-1} \phi_n \\ 0 & 0 & \int_{x_{n-1}}^{x_n} x^2 \phi_n \phi_{n-1} & \int_{x_{n-1}}^{x_n} x^2 (\phi_n)^2 + \int_{x_n}^{x_{n+1}} x^2 (\phi_n)^2 \end{bmatrix}$$

Chapter 4

Numerical Methods

We will now be exploring some of the numerical methods we will use to set up and solve the eigenvalue problems we attain from applying the FEM to the Schrödinger equation. We will begin by looking at the way we set up matrices P and M. When computing the integrals in M and P it's important to notice what we are integrating. For M we are multiplying our two ϕ's which are linear on the elements. For P we are adding the x^2 term. This means that the we will be integrating a polynomial of at most degree 4.

4.1 Gaussian Quadrature

Unlike other rules for numerical integration, Gaussian Elimination chooses optimal evaluation points. Rather than equally spaced points, such as those used in the Trapezoidal Rules, we can determine the best points and weights for each quadrature rule [2]. These are methods that work off of approximating integration as follows:

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n} w_i f(x_i) \tag{4.1.1}$$

The different methods are ways of arriving at values for w_i and x_i that will best approximate the integral. What is important to note about this numerical method, is that if we use a n-point Gaussian Quadrature rule, integration will be accurate for polynomials of degree up to $2n-1$[2]. With that in mind we can show how to come up with the values w_i and x_i for a polynomial $f(x)$ of degree 3. From our discussion of accuracy, we know that we can achieve an ideal solution by using a 2-point rule. Therefore:

$$\int_{-1}^{1} f(x)dx = w_1 f(x_1) + w_2 f(x_2) \tag{4.1.2}$$

Where we have $f(x)$ as the polynomial:

$$f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 \tag{4.1.3}$$

Integrating this $f(x)$ we would get something of the form:

$$\int a_0 + a_1 x + a_2 x^2 + a_3 x^3 dx = a_0 \int 1dx + a_1 \int x dx + a_2 \int x^2 dx + a_3 x^3 dx \tag{4.1.4}$$

Since we know that we will have exact solutions for any polynomial of degree less than 3, we can find w_1, w_2, x_1, x_2 by looking at all these lower degrees:

$$f(x) = 1 : w_1 \cdot 1 + w_2 \cdot 1 = \int_{-1}^{1} 1dx = 2 \tag{4.1.5}$$

$$f(x) = x : w_1 x_1 + w_2 x_2 = \int_{-1}^{1} x dx = 0 \tag{4.1.6}$$

$$f(x) = x^2 : w_1 x_1^2 + w_2 x_2^2 = \int_{-1}^{1} x^2 dx = \frac{2}{3} \tag{4.1.7}$$

$$f(x) = x^3 : w_1 x_1^3 + w_2 x_2^3 = \int_{-1}^{1} x^3 dx = 0 \tag{4.1.8}$$

With these four equations and four unknowns, we can come up with the following values:

$$w_1 = 1, \quad w_2 = 1, \quad x_1 = -\frac{\sqrt{3}}{3}, \quad x_2 = \frac{\sqrt{3}}{3} \tag{4.1.9}$$

By using this we can come up with higher order rules to calculate integrals for higher degree polynomials. For our purposes using a 5-point rule will be more than enough since it will allow us to integrate polynomials up to degree 9.

Therefore our integrals from matrix M become:

$$\int_{x_1}^{x_2} \phi_2(x)\phi_1(x)dx = \frac{x_2 - x_1}{2} \sum_{i=1}^{5} w_i \phi_2(\frac{x_2 - x_1}{2} x_i + \frac{x_1 + x_2}{2}) \phi_1(\frac{x_2 - x_1}{2} x_i + \frac{x_1 + x_2}{2}) \tag{4.1.10}$$

Notice that here we are using extra constants being multiplied outside the summation as well as within the basis functions. This is due to the fact that Gaussian Quadrature rules are set for the interval (-1.1), and if we wish to use other intervals of integration we merely have to create a shift such as the one seen in equation (4.1).

Here, x_i and w_i are given by:

	x_i	w_i
1	$-\frac{1}{3}\sqrt{5 + 2\sqrt{10/7}}$	$\frac{322-13\sqrt{70}}{900}$
2	$-\frac{1}{3}\sqrt{5 - 2\sqrt{10/7}}$	$\frac{322+13\sqrt{70}}{900}$
3	0	$\frac{128}{225}$
4	$\frac{1}{3}\sqrt{5 - 2\sqrt{10/7}}$	$\frac{322+13\sqrt{70}}{900}$
5	$\frac{1}{3}\sqrt{5 + 2\sqrt{10/7}}$	$\frac{322-13\sqrt{70}}{900}$

Using this 5-point Quadrature rule allows us to then calculate the integrals necessary to construct M and P without any error. Note that (4.1)

would be the formula for computing entry $m_{2,1}$ of matrix M.

4.2 Inverse Power Method

Once we have the matrices constructed, we are left with the following eigenvalue problem:

$$M^{-1}(A+P)c = \lambda c \tag{4.2.1}$$

$$B = M^{-1}(A+P) \tag{4.2.2}$$

$$Bc = \lambda c \tag{4.2.3}$$

In order to solve this problem we will be using a modification of the power method. The power method is an iterative method that works by repeatedly applying B in order to get the dominant eigenvalue. In our case we are looking for the least dominant eigenvalue, thus we will be using a modification of the technique called the inverse power method [2]. What the inverse power method does is instead of using (4.2.3) we can use:

$$(B - qI)^{-1}c = \tilde{\lambda}c \tag{4.2.4}$$

Here instead of converging to the largest eigenvalue we will be converging to the eigenvalue closest to q. By setting $q = 0$ we can thus use this method to find the smallest eigenvalue of our problem (4.2.3).

There is one issue that we must deal with before we move on to looking at the error that comes from applying the FEM. The fact that the power method (and the inverse power method) are iterative methods means that in the input we have to pass in some signification to stop. In our implementation we are using two such checks. The first is a maximum number of iterations, and the second is a tolerance of how fast our eigenvalue is converging. If our iterations converge to a solution with error under the tolerance, then it

would output this solution; otherwise, an error message would be displayed. In our implementation of the inverse power method we are using the following values:

1. tolerance=.00000000000001
2. maximum number of iterations=100

The error message assures that the solutions being achieved are not being achieved due to the iterations exceeding 100.

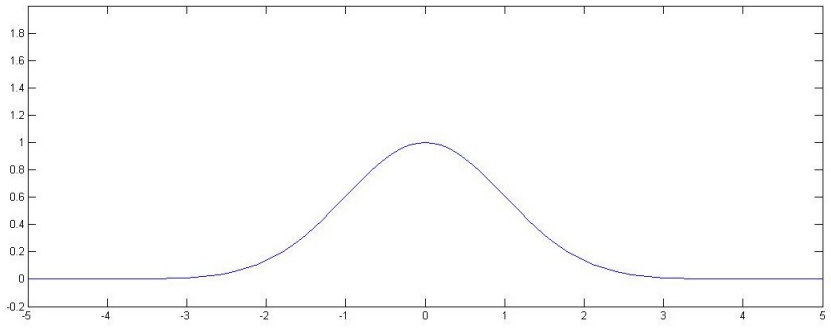
Chapter 5

Sources of Error

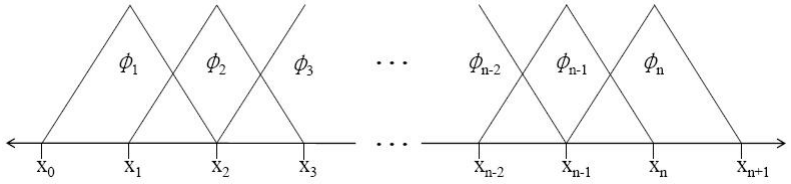
When looking at the solution attained by the finite element method, we have to understand that it is an approximation and thus will have some inherent error involved. If we ignore any error we might incur from the Inverse Power Method or using Gaussian Quadrature, we will be looking at two sources of error. The aim of this project is to look at the interaction between these two sources and by finding a relation between them, be able to manipulate them accordingly.

5.1 R-Size of Mesh

First, in our implementation of the FEM we chose to have n basis functions for $n+2$ nodes. We made this choice because we knew we had Dirichlet boundary conditions at x_0 and x_{n+1}. In other words we were restricting our solution to be 0 at the endpoints. In the case of the ground state solution to the Schrödinger equation, recall that we are looking at the solution of the form $e^{-\frac{1}{2}x^2}$:



Because the solution tends to 0 as we go to $+\infty$ and $-\infty$, we can use basis that are fixed at 0 at the endpoints:



Where we decide to place our x_0 and x_{n+1} will be the first major source of error we will incur. Due to the fact that our solution is symmetric across the y-axis, we will be using:

$$|x_0| = |x_{n+1}| := R \tag{5.1.1}$$

This R then defines the first source of error and thus the first input into our algorithm when implementing the FEM.

5.2 h-Size of Elements

The second source we will be looking at is the size of our elements. Much as in the case of Riemann integrals, the exact solution would be the result of taking the limit as our element size went to zero (and as $R \to \infty$). Recall

that we defined h as:

$$h_i = x_i - x_{i-1} \text{ for } i = 1, \ldots, n+1 \tag{5.2.1}$$

The importance of this source of error as it relates to the solution we are analyzing has to do with uniformity. We have seen in the implementation of the FEM that error from our h's can be minimized by increasing the number of nodes (and thus decreasing the h) in regions where there is more activity. In other words, if we have a solution that is fairly uniform in some areas and contains more activity in others, we can decrease our error by putting more nodes in the area of higher activity.

In the case of our problem, we can look ahead and intuit that the use of more nodes near 0 will create a more exact approximation to our solution.

Chapter 6

Implementation

With these two sources of error in mind, we can now begin to look at the way they affect the solution of our problem. In order to look at this, we will be performing a set number of iterations of certain refinements of our mesh. Here, the word refinement is used to refer to two things:

1. splitting existing elements into two elements, and thus increasing the number of nodes. In other words we are affecting h.

2. increasing the size of our mesh by a fixed amount on the left and the right. This is equivalent to increasing the size of R.

In order to understand how these two types of refinements affect the way our problem converges to the known solution, we will be using the following algorithm:

Input: R,h
mesh=[-R:h:R] {From the given R and h we will get our mesh starting at -R and going to R with elements of size h}
for $t = 1$ to 5 **do**
  Form matrix A

Form matrix M

Form matrix P

Get smallest eigenvalue to the problem $M^{-1}(A+P)c = \lambda c$

Refine our mesh

end for

Recall that we will be applying this implementation of the Finite Element Method to the Schrödinger equation with potential x^2 that looks like:

$$-u''(x) + x^2 u(x) = \lambda u(x) \tag{6.0.1}$$

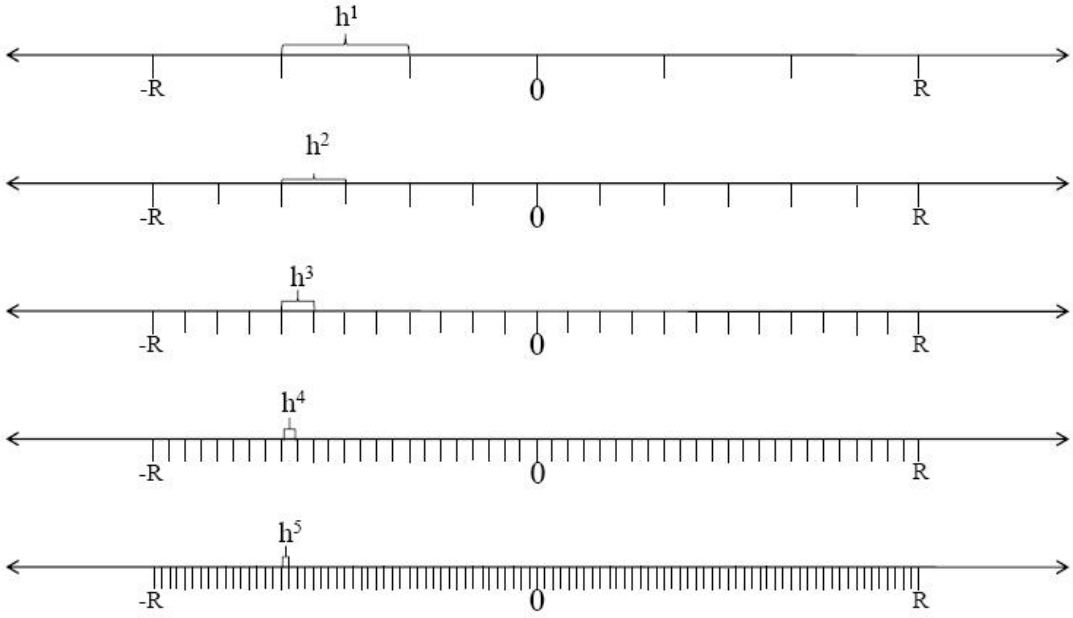
So that we are looking for the lowest energy solution which is:

$$u(x) = e^{\frac{1}{2}x^2} \tag{6.0.2}$$

$$\lambda = 1 \tag{6.0.3}$$

6.1 Refining h

The first computations we are going to run will be using a refinement where we will only be refining h. We will pick a fixed R and conduct 5 refinements of our mesh. Take an initial $h = h^{(1)}$. The iterations will look as follows:



Note that:

$$h_n^{(i)} = \frac{h_n^{(i-1)}}{2} \tag{6.1.1}$$

We have picked this factor of $\frac{1}{2}$ as the change in our h.

Note: The example shown above for the iterations contains a uniform mesh (all h_n's are equal). This will not always be the case. The idea behind this kind of refinement is used in the general case where the h_n's are not equal.

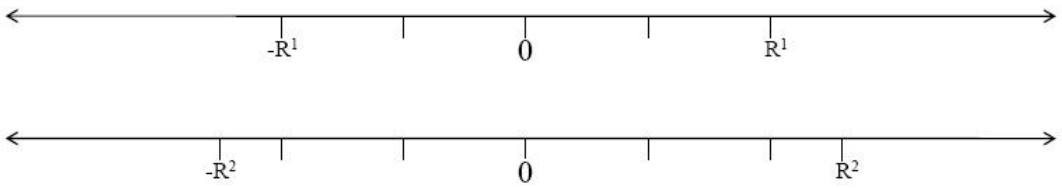
6.2 Refining R

When we talk about refining R we are referring to a process where our h will stay fixed and we will expand R as our iterations go along. Take an initial $R = R^1$. In the refinement of h we picked a factor of $\frac{1}{2}$ for the change of h. Here we are going to have to choose how the R will change. In this case we

will be using the following equation:

$$R^n = R^1 + \frac{(n-1) * R^1}{4} \tag{6.2.1}$$

Thus, one iteration of this refinement will look as follows:



6.3 Results Comparing Two Refinements

We will now look at the results we get from running our program using these two iterations. The way we will be comparing the results will be based off of the quotient between the eigenvalues during each successive iteration. From each computation we run, we will be attaining the following values: $\lambda_1^{(1)}, \lambda_1^{(2)}, \lambda_1^{(3)}, \lambda_1^{(4)}, \lambda_1^{(5)}$ (because we are only looking at the ground state solution, we will only be looking at the lowest eigenvalue and thus we can ignore the 1 that appears in the subscript of λ). In order to see the convergence of these eigenvalues towards our solutions we can look at the following formula:

$$c^n = \frac{|\lambda^{(n+1)} - \lambda^{(n)}|}{|\lambda^{(n)} - \lambda^{(n-1)}|} \tag{6.3.1}$$

Considering we will be using 5 iterations of each refinement, we will have 3 of these values to look at.

First, let us look at the λ's from running our refinements on a mesh with $R^1 = 2$ and $h^{(1)} = .5$

	Refine R	Refine h
$\lambda^{(1)}$	1.089705387815853	1.089705387815853
$\lambda^{(2)}$	1.024460781878389	1.078657867350158
$\lambda^{(3)}$	1.016126447838206	1.075859139122253
$\lambda^{(4)}$	1.015576343305613	1.075156783263255
$\lambda^{(5)}$	1.015559418937207	1.074981021366152

Table 6.1: Mesh with $R^1 = 2$ and $h^{(1)} = .5$

With these λ's we can now look at our convergence rates using (6.3.1) :

	Refine R	Refine h
c^1	0.127739817268125	0.253335419164453
c^2	0.066004617758455	0.250955363223620
c^3	0.030765731607488	0.250246217572456

Table 6.2: Rates of Convergence for $R^1 = 2$ and $h^{(1)} = .5$

Notice here that after 5 iterations, our solution where we refined R is actually closer to the exact solution of 1. However, the rates of convergence tell us that refinement of h is actually converging faster than our R. By keeping our R constant at 2, we are assuring that we will be far from our exact solution, however this h refinement still converges at an even pace. Let us look at another case with a larger R.

	Refine R	Refine h
$\lambda^{(1)}$	1.015559418937207	1.015559418937207
$\lambda^{(2)}$	1.015559247652806	1.003902650356347
$\lambda^{(3)}$	1.015559243285470	1.000977205107389
$\lambda^{(4)}$	1.015559243133214	1.000245088762323
$\lambda^{(5)}$	1.015559243126635	1.000062011687892

Table 6.3: Mesh with $R^1 = 4$ and $h^{(1)} = .5$

With these λ's we can now look at our convergence rates using (6.3.1) :

	Refine R	Refine h
c^1	0.025497566285531	0.250965370777134
c^2	0.034862545776977	0.250258091593456
c^3	0.043205357429321	0.250065547183918

Table 6.4: Rates of Convergence for $R^1 = 4$ and $h^{(1)} = .5$

This second run of calculations gives a bit of a different picture. By increasing R to 4, we are now creating a situation where our rate of convergence is still better with refinements of h. However, we now notice that this increase of R also allows these refinements of h to be better after 5 iterations than our refinements of R.

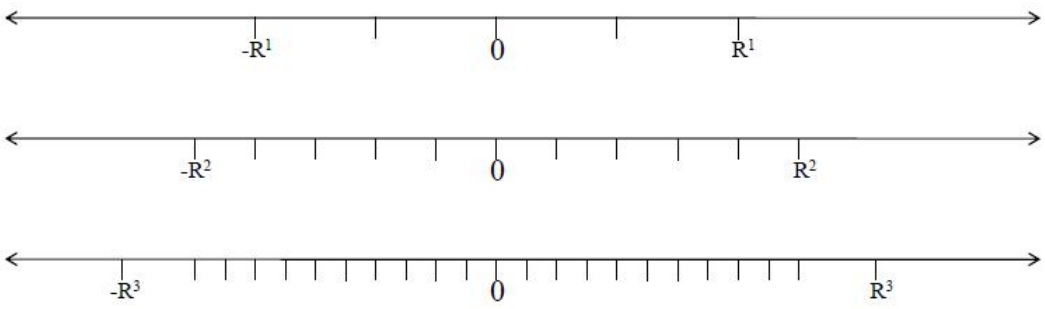
Here we see the dilemma of these two types of refinements. On the one hand, we need R to be large enough so that the solution we are approximating is the solution we want. On the other hand, we need our h's to be as small as possible in order for the approximation to be more accurate. This inspires a third type of refinement which is a combination of the two things we see above.

6.4 Optimal Refinement

Recall that it was mentioned earlier that it would be beneficial to have more nodes around areas of high activity in our solution. This, combined with the complication we saw above, brings us to an optimal refinement. At each iteration we will be doing two things:

1. Refine all elements. Grab every element and split it in two.
2. Add a large element to our ends and thus increase R. What is added to our R is the same as in the refinement of R.

Thus, three iterations of this refinement will look as follows:



There are two important things to note here. First, the order in which we do the two parts are important. First we divide any element in two, and then we add the extra part on the ends. Second, notice that this will give us non-uniform h's. This is something different from what we did in the other two refinements.

We can now apply the inputs used in the last section to test this new optimal refinement. First, let's look at the second set of parameters we used:

	Refine R	Refine h	Optimal Refinement
$\lambda^{(1)}$	1.015559418937207	1.015559418937207	1.015559418937207
$\lambda^{(2)}$	1.015559247652806	1.003902650356347	1.003901956341820
$\lambda^{(3)}$	1.015559243285470	1.000977205107389	1.000976302135762
$\lambda^{(4)}$	1.015559243133214	1.000245088762323	1.000244127190784
$\lambda^{(5)}$	1.015559243126635	1.000062011687892	1.000061035039447

Table 6.5: Mesh with $R^1 = 4$ and $h^{(1)} = .5$

With these λ's we can now look at our convergence rates using (6.3.1) :

	Refine R	Refine h	Optimal Refinement
c^1	0.025497566285531	0.250965370777134	0.250968354572796
c^2	0.034862545776977	0.250258091593456	0.250260247250885
c^3	0.043205357429321	0.250065547183918	0.250066125032368

Table 6.6: Rates of Convergence for $R^1 = 4$ and $h^{(1)} = .5$

Here we notice that there isn't much difference between the refinement of h and our optimal refinement. In a sense, we can say that R^1 is large enough that it's expansion in the optimal refinement isn't helping our solution significantly. However, if we look at the case that $R^1 = 2$, we have the following situation:

	Refine R	Refine h	Optimal Refinement
$\lambda^{(1)}$	1.089705387815853	1.089705387815853	1.089705387815853
$\lambda^{(2)}$	1.024460781878389	1.078657867350158	1.013571164411471
$\lambda^{(3)}$	1.016126447838206	1.075859139122253	1.001778588307346
$\lambda^{(4)}$	1.015576343305613	1.075156783263255	1.000299229798247
$\lambda^{(5)}$	1.015559418937207	1.074981021366152	1.000067924070450

Table 6.7: Mesh with $R^1 = 2$ and $h^{(1)} = .5$

With these λ's we can now look at our convergence rates using (6.3.1) :

	Refine R	Refine h	Optimal Refinement
c^1	0.127739817268125	0.253335419164453	0.154891920831572
c^2	0.066004617758455	0.250955363223620	0.125448290181602
c^3	0.030765731607488	0.250246217572456	0.156355424580964

Table 6.8: Rates of Convergence for $R^1 = 2$ and $h^{(1)} = .5$

It is in this case that we can notice the power of our optimal refinement. The number of nodes in the refinement of h and our optimal refinement are about the same. However, the inclusion of some expansion of R makes this optimal refinement a much better way of approximating our solution.

Chapter 7

Conclusion and Possible Future Works

The Finite Element method in one dimension allows us to solve problems such as the Schrödinger equation by transforming them into simple eigenvalue matrix problems. The discretization process involves finding a piecewise linear basis on our elements to approximate our solution. From here we find that by using simple numerical methods we can build the matrices that make up the eigenvalue problem.

At this point, we began our investigation of the different sources of error. In this particular problem there are a couple of factors that play into their relation. We saw that at whatever value we decided to fix R we would be approximating a solution that is zero at $-R$ and R. This made it so that after fixing R whatever refinement we did with the h could only be as close to our real solution as this R allowed. The same could be said for h. Once we fixed an h and decided to increase R, we were still bound by how fine our h was.

The optimized solution proved a way to include more points in the relevant areas while increasing R and refining h. Most importantly, this was all done without drastically increasing the number of nodes in comparison

to our other refinements. This is important because the run time for our algorithm increases with an increase in the number of nodes in our mesh. Further research could be done in the analysis of run time in comparison with the error minimization. The optimal refinement proposed in here serves as a first step in the right direction since it converges towards the actual solution without having to double our nodes.

Bibliography

[1] Cosmin Anitescu. On the convergence and superconvergence of the generalized finite element methods, 2010. TY: GEN; M1: Syracuse University; M2: Ph.D.

[2] Richard L. Burden and J.Douglas Faires. *Numerical Analysis*. Brooks/Cole, ninth edition, 2011.

[3] Ricardo J. Cordero-Soto. Solvable time-dependent models in quantum mechanics, 2011. TY: GEN; M1: Applied Mathematics for the Life and Social Sciences; M2: Ph.D.

[4] C. Filgueiras, E.O. Silva, W. Oliveira, and F. Moraes. The effect of singular potentials on the harmonic oscillator. *Annals of Physics*, 325(11):2529 – 2541, 2010.

[5] A. Mitchell. An introduction to the mathematics of the finite element method, 1973.

[6] G. Pelosi. The finite-element method, part i: R. l. courant [historical corner]. *Antennas and Propagation Magazine, IEEE*, 49(2):180 –182, april 2007.

[7] E. Schrödinger. An undulatory theory of the mechanics of atoms and molecules. *Phys. Rev.*, 28(6):1049–1070, Dec 1926.

ACADEMIC VITA of Ignacio Sofo

Ignacio Sofo
2327 Setter Run Lane
State College, PA, 16801
nacho.sofo@gmail.com

Education: Bachelor of Science Degree in Mathematics, Penn State University, Spring 2011
Bachelor of Arts Degree in Philosophy, Penn State University, Spring 2011
Honors in Mathematics
Thesis Title: Using the Finite Element Method for Solving the Schrödinger Equation
Thesis Supervisor: Victor Nistor

Related Experience:

Participated in the Penn State-Göttingen International Summer School in Mathematics.
Faculty: Juan Gil, Thomas Krainer
Summer 2010

Awards:

Ray H. Dotterer Scholarship in Philosophy
Superior Academic Achievement recognition
Dean's List

Activities:

Editor for Penn State Student Handbook
Manager and practice player for Penn State Women's Soccer Team