FLORIDA STATE UNIVERSITY

COLLEGE OF ARTS AND SCIENCES

REDUCED ORDER MODELING USING THE WAVELET-GALERKIN APPROXIMATION OF DIFFERENTIAL EQUATIONS

By

DAVID WITMAN

A Thesis submitted to the
Department of Scientific Computing
in partial fulfillment of the
requirements for the degree of
Master of Science

Degree Awarded:
Fall Semester, 2013

David Witman defended this thesis on October 30, 2013.
The members of the supervisory committee were:

Janet Peterson

Professor Directing Thesis

Max Gunzburger

Committee Member

Ming Ye

Committee Member

The Graduate School has verified and approved the above-named committee members, and certifies that the thesis has been approved in accordance with university requirements.

I would like to dedicate this work to the memory of David S. Prenatt and Rodney H. Witman for their incredible influence in my life and academic studies.

ACKNOWLEDGMENTS

I would like to thank and acknowledge the extraordinary guidance of my adviser, Dr. Janet Peterson, for her unwavering support throughout the writing of this thesis. I would also like to thank Pratt & Whitney for allowing me the opportunity to work this past summer in industrial research position while continuing my research for this thesis. Finally I would like to thank my parents for encouraging me to pursue all my academic aspirations.

TABLE OF CONTENTS

List of Tables vii

List of Figures viii

Abstract x

1 Introduction **1**

2 Wavelets and the Scaling Function **3**

2.1 The Haar Scaling and Wavelet Functions 5

2.1.1 Approximation Example 8

2.2 The Daubechies Scaling Function 10

2.3 Generating the Scaling and Wavelet Functions 12

2.4 Multi-resolution 13

3 The Galerkin Finite Element Method **17**

3.1 Preliminary Definitions 17

3.2 Poisson's Equation 18

3.2.1 The Weak Formulation 19

3.2.2 Using Piecewise Linear Polynomial Basis Functions 21

3.2.3 Error Estimate 22

3.2.4 Quadrature Rules 24

3.2.5 The FEM Algorithm 25

3.2.6 Results 26

3.3 The One Dimensional Heat Equation 27

3.3.1 Weak Formulation 27

3.3.2 FEM Algorithm for Time Dependent Problems 28

3.3.3 Numerical Results 30

4 Reduced Order Modeling **33**

4.1 Generating Reduced Basis Vectors using FEM 34

4.2 Solution Approximation using Reduced Basis 39

5 The Wavelet-Galerkin Method **44**

5.1 Weak Formulation 45

5.2 Calculating the Connection Coefficients 46

5.3 Handling Dirichlet Boundary Conditions 52

5.4 Solving for the Wavelet-Galerkin Approximation 54

5.5 Numerical Results 58

6 Reduced Order Model of the Wavelet-Galerkin Method **65**
6.1 Parameter Selection, Snapshot Generation and Reduced Basis Construction 66
6.2 Construction of the Reduced Wavelet-Galerkin Solution 70
6.3 Results 73
6.4 Conclusions and Future Work 76

Bibliography 79

Biographical Sketch 81

LIST OF TABLES

2.1 The Daubecheis scaling function coefficients(c_k) 11

3.1 One dimensional Poisson equation solved with piecewise linear basis functions 26

3.2 Finite element convergence rates using piecewise-linear basis functions 30

4.1 Example 1: FEM Reduced order approximation error while increasing the number of basis functions 42

4.2 Example 2: FEM Reduced order approximation error while increasing the number of basis functions 42

5.1 Connection Coefficients for $D6$ 52

5.2 Connection Coefficients for $D8$ 53

5.3 Connection Coefficients for $D10$ 54

5.4 Example 1: Euclidean error and the condition numbers for scaling functions: $D6$ through $D20$ 64

5.5 Example 2: Euclidean error and the condition numbers for scaling functions: $D6$ through $D20$ 64

5.6 Example 2: wavelet-Galerkin convergence rates using scaling function $D10$ 64

6.1 Example 1: Euclidean error comparison between wavelet-Galerkin and finite element reduced order models 76

6.2 Example 2: Euclidean error comparison between wavelet-Galerkin and finite element reduced order models 76

LIST OF FIGURES

2.1 Haar scaling function 6

2.2 Haar wavelet 7

2.3 Haar scaling function translates, of $\phi(x)$, $\phi(x+1)$ and $\phi(x-1)$ 7

2.4 Haar scaling functions and the scaled translates, of $\phi(2x)$: $\phi(2x+1)$ and $\phi(2x-1)$. 8

2.5 Haar basis function approximation at varying resolutions with the wavelet approximation superimposed 9

2.6 Approximation to $D4$ scaling function using dyadic points: $\frac{1}{2}$, $\frac{1}{4}$, $\frac{1}{8}$, $\frac{1}{16}$ 14

2.7 Daubechies Wavelet and Scaling functions $D4$ to $D18$ 16

3.1 Piecewise linear basis functions on $[0,1]$ 22

3.2 FEM solution to our modified one-dimensional heat equation vs. actual solution at 5 time instances 31

4.1 Actual solution at $t = 0, 0.025, 0.05, 0.075, 0.1$ 35

4.2 Sample FEM snapshots with $t = 0, 0.025, 0.050, 0.075, 0.100$ 36

4.3 Singular values of FEM snapshot matrix 37

4.4 First four basis vectors calculated using SVD 38

4.5 Reduced order model approximation to the finite element solution with one basis function for five time instances 40

4.6 Reduced order model approximation to the finite element solution with two basis functions for five time instances 41

4.7 Reduced order model approximation to the finite element solution with three basis functions for five time instances 41

5.1 $D6$ Scaling basis functions with $m = 0$ 47

5.2 Example 1 Wavelet-Galerkin solution compared to actual solution at varying resolutions at 5 time instances 60

5.3 Example 1 Wavelet-Galerkin solution compared to actual solution at varying resolutions at 5 time instances 61

5.4 Example 1 Wavelet-Galerkin solution compared to actual solution at different ordered scaling functions (plots on the right are zoomed in) 62

5.5 Example 2 Wavelet-Galerkin solution compared to actual solution at different ordered scaling functions (plots on the right are zoomed in) 63

6.1 Sample FEM snapshots with $t = 0, 0.025, 0.050, 0.075, 0.100$ 67

6.2 Singular Values of the snapshot sets 68

6.3 First four basis vectors of the wavelet-Glerkin solution approximation to the one dimensional modified heat equation 69

6.4 Five time instances of the reduced order solution compared to the wavelet-Galerkin solution using only a single basis function 74

6.5 Example 1: Five time instances of the reduced order solution compared to the wavelet-Galerkin solution using two and three basis functions 75

6.6 Example 2: Five time instances of the reduced order solution compared to the wavelet-Galerkin solution using two and three basis functions 78

ABSTRACT

Over the past few decades an increased interest in reduced order modeling approaches has led to its application in areas such as real time simulations and parameter studies among many others. In the context of this work reduced order modeling seeks to solve differential equations using substantially fewer degrees of freedom compared to a standard approach like the finite element method. The finite element method is a Galerkin method which typically uses piecewise polynomial functions to approximate the solution of a differential equation. Wavelet functions have recently become a relevant topic in the area of computational science due to their attractive properties including differentiability and multi-resolution. This research seeks to combine a wavelet-Galerkin method with a reduced order approach to approximate the solution to a differential equation with a given set of parameters. This work will focus on showing that using a reduced order approach in a wavelet-Galerkin setting is a viable option in determining a reduced order solution to a differential equation.

CHAPTER 1

INTRODUCTION

With the increased need for real time simulations and large parameter studies, research into the ability to quickly generate an approximate solution to a complex system of partial differential equations defined in terms of a set of parameters has been a hot topic in recent years. The hardware available today allows for large numerical simulations to be performed in an instant but it is still un-realistic to solve a large number of complex partial differential equations in a reasonable amount of time. Thus the ability to build reduced order models of differential equations has become a go-to method in order to generate realizations of these equations with a low computational cost. Reduced order modeling aims to take a very high-dimensional system and reduces it to only depend on a few degrees of freedom so that a solution to a differential equation with unknown input parameters can be generated quickly. This technique has been used extensively in standard Galerkin finite element methods to solve differential equations from a number of different applications [9][15]. One reduced order modeling approach, that has been thoroughly studied in a finite element setting is the proper orthogonal decomposition which aims to extract the dominant characteristics of a solution and then use those characteristics to reconstruct a solution with a given set of parameters.

Within the past two decades, wavelets have also played a huge role in the computational science field from topics ranging from image compression to signal processing and voice recognition among many others. Wavelets are functions with very interesting and attractive features for computational scientists including multi-resolution, differentiability and orthogonality. For example the JPEG image compression standard relies on wavelets to compress the data contained within a photo such that one can get the full effect of an image but also detect the complexities and details of the image. The properties of wavelets present a great opportunity for a confluence of large and small scale information to be combined together within the same data set. One of the pre-eminent wavelet families is known as the Daubechies family of orthogonal wavelets. The Daubechies family of wavelets were initially introduced by Ingrid Daubechies in a series of lectures aimed at presenting a number of the applications that wavelets were being used in at the time[6]. One such

application of wavelets is using wavelet scaling functions as bases in a standard Galerkin method. This wavelet-Galerkin approach to solving differential equations has been widely documented in the literature[13][8][14][3]. Wavelet scaling functions offer a number of promising capabilities that can be used to expand and enhance the currently available sets of basis functions. Applications in approximating discontinuous functions as well as representing multi-scaled problems are among the reasons for pursuing a wavelet-Galerkin method.

This research aims to demonstrate the ability to couple a wavelet-Galerkin method with reduced order modeling to approximate the solution to a partial differential equation and compare it the the finite element based reduced order model. Explicitly we wish to show only the viability of applying a reduced order approach to an existing wavelet-Galerkin method. We note that this is not an exhaustive study designed to resolve all of the problems inherent in the wavelet-Galerkin method. The prototype equation that we will use throughout the course of this paper is the one dimensional heat equation.

The first chapter will explain some of the useful properties and background of the Daubechies family of wavelets, including an approximation example using a simple scaling and wavelet function. Then a review of the finite element method will be presented with two examples to show how a finite element solution to the differential equation can be constructed. Using this finite element example we will present a method for generating a reduced order model of the standard finite element method using the proper orthogonal decomposition. In chapter four we will present the wavelet-Galerkin method used in this work, including a thorough investigation of how to generate the inner products occurring in the weak formulation. Finally all the concepts will come together as we demonstrate the feasibility of a wavelet-based Galerkin method coupled with a reduced order model using the same example that was used in the finite element reduced order chapter.

CHAPTER 2

WAVELETS AND THE SCALING FUNCTION

In order to formulate a wavelet-Galerkin approximation to a boundary or initial value problem, we must first introduce the concept of a wavelet. By definition, wavelets are functions that are non-zero over a given domain and tend to have oscillatory properties while the function has zero mean. Wavelets are most well known for their uses in image compression; more specifically the JPEG filetype [19]. When compressing images it is necessary to be able to represent large and small scale features so that one can not only get the overall view of the image but also the smaller details. Using wavelets as a basis for the Galerkin method is attractive because of their abilities to represent a wide range of spatial and temporal scales due to their multiresolution properties. Also we note that unlike Fourier bases, but similar to piecewise polynomial bases, wavelets do not need to be defined over an infinite domain as they have compact support meaning the function values are zero outside of a compact set [17]. Additionally, wavelets are smoother than piecewise polynomial bases but also orthogonal like a Fourier basis; so in terms of properties, wavelets bridge some of the gaps between commonly used Galerkin bases. Wavelets are defined in terms of families depending on how they are constructed. That being said, in this work we will focus on the Daubechies family of orthogonal wavelets.

One major difficulty with wavelets, in general, is the ability to represent them in a typical functional form. All the wavelets that will be used in this work are represented by an underlying scaling function which will be defined by a recurrence relation. This means that the functional representation of the equation will appear on either side of the equation. Equation 2.1 presents the form in which a general scaling function is defined in terms of coefficients, a_k.

$$\phi(x) = \sum_{k=-\infty}^{\infty} a_k \phi(2x - k) \tag{2.1}$$

where ϕ represents the scaling function and k is an integer value. This equation is typically referred to as the dilation equation. We will require that:

$$\int_{-\infty}^{\infty} \phi(x) = 1 \tag{2.2}$$

and all the translates of $\phi(x)$, denoted by $\phi(x-k)$, be orthonormal to $\phi(x)$. For example, we must be able to say $\int_{-\infty}^{\infty} \phi(x)\phi(x-k) = 0$ for all $k \neq 0$ and $\int_{-\infty}^{\infty} \phi^2(x) = 1$.

Using the dilation equation we can then define the wavelet relation which is dependent on the scaling functions and the reverse set of coefficients (a_{1-k}). The wavelet function can be constructed using the following relation.

$$w(x) = \sum_{k=-\infty}^{\infty} (-1)^k a_{1-k} \phi(2x-k) \tag{2.3}$$

where $w(x)$ is the wavelet function. In general, this work will refer to ϕ as the scaling function and w as the wavelet function. Knowledge of the wavelet function is beneficial, but not necessary to this research as we will only use scaling functions as our Galerkin bases when we implement the wavelet-Galerkin method.

Extending the concept of the dilation equation, we can write a relation that allows one to shift and scale the scaling function. Equation 2.4 includes the shift (k) and scaling (m) terms used in the dilation equation to adjust the positioning and size of our scaling function. In this work we will often refer to the term $\phi_{m,k}$ which can be represented by equation 2.4.

$$\phi_{m,k}(x) = 2^{\frac{m}{2}} \phi(2^m x - k) \tag{2.4}$$

where $2^{\frac{m}{2}}$ is a scaling factor that allows the orthonormality condition to be satisfied.

One nice characteristic of the scaling functions is that for a single function at a fixed level of resolution, m, all translates of the original function are orthonormal to the original and each other. To see this, we recall the orthonormality property by stating that at a fixed resolution, $m = q$, all translates, $\phi_{q,k}$ are orthonormal. We can show this from equation 2.4 by using a change of variables. By setting $u = 2^m x - k$ we can then write the orthonormality condition in terms of u.

$$\begin{aligned} 2^m \int_{-\infty}^{\infty} \phi_{m,k}\phi_{m,l} &= 2^m \int_{-\infty}^{\infty} \phi(2^m x - k)\phi(2^m x - l)dx \\ &= 2^m \frac{1}{2^m} \int_{-\infty}^{\infty} \phi(u)\phi(u-s)du \end{aligned} \tag{2.5}$$

where $s = k - l$ as a shift factor. Using this equation we can see how $\phi_{m,k}$ is still an orthonormal function after including the resolution term, m.

This concept of orthonormality is not guaranteed for all basis functions, like piecewise linear basis functions, that will be examined in the finite element section. But this concept of orthonormal functions is inherent in the Daubechies family of scaling functions. So in general, we can define the concept of orthonormality by the equation:

$$\int_{-\infty}^{\infty} \phi_{m,k}\phi_{m,j}dx = \delta_{j,k} \tag{2.6}$$

where $\delta_{j,k}$ is the Kronecker delta function such that the integral evaluates to 1 if $j = k$ and 0 otherwise. The ability to define a function as orthonormal will be very useful when we formulate our wavelet-Galerkin method and will allow us to make some simplifying assumptions, reducing the complexity of the problem we must solve.

In this chapter we will first present a simple wavelet and corresponding scaling function, called the Haar wavelet which will be used to approximate a simple function. Then the Daubechies family of wavelets will be introduced including the coefficient values used to solve the dilation equation. Following this introduction an example Daubechies scaling function will be resolved using the dilation equation and the Cascade algorithm. Using the scaling function and our knowledge of the Daubechies coefficients, we should then be able to construct and present the Daubechies family of scaling and wavelet functions. Finally the concept of multi-resolution will be presented along with its uses and benefits.

2.1 The Haar Scaling and Wavelet Functions

One of the most basic wavelets is the Haar wavelet[10], developed by Alfréd Haar in 1910 and is considered the simplest Daubechies wavelet. The Haar scaling function is literally a square wave, constructed such that it has a constant value of 1 over a compact domain and exactly zero everywhere else. The general equation for the Haar scaling function is given in equation 2.7

$$\phi(x) = \begin{cases} 1 & \text{for } 0 \leq x \leq 1 \\ 0 & \text{elsewhere} \end{cases} \tag{2.7}$$

Figure 2.1 shows the shape of the Haar function. To show that equation 2.7 is a solution to the dilation equation 2.1, we can set $a_0 = a_1 = 1$ and all remaining coefficients to zero, to get the equation:

$$\phi(x) = (1)\phi(2x) + (1)\phi(2x-1) \tag{2.8}$$

Because $\phi(2x)$ is only one over the interval $[0, \frac{1}{2}]$ and $\phi(2x-1)$ is only one over the interval $[\frac{1}{2}, 1]$, clearly this is a solution to this equation. Then after resolving the scaling function we can construct the Haar wavelet according to equation 2.3.

$$w(x) = (-1)^0\phi(2x) + (-1)^1\phi(2x-1) \tag{2.9}$$

Using this equation and the Daubechies coefficients the Haar wavelet can be resolved to create a square wave shown in Figure 2.2. From the definition of the Haar scaling function and Figure 2.1,

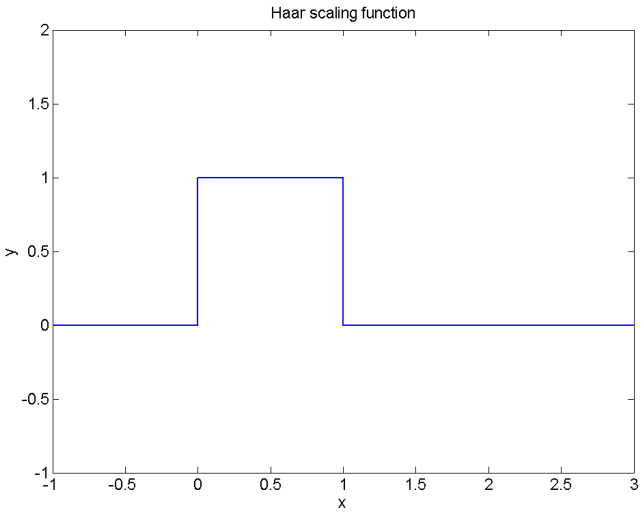


Figure 2.1: Haar scaling function

we clearly see that the integration of the function over its entire domain is one, as expected.

Now, we want to be able to use the scaling function to approximate real-valued functions; to accomplish this, we must define it as a basis function. Thus we need to be able to use the scaled and translated scaling functions within our analyses. Figures 2.3 and 2.4 depict what the translated and scaled Haar functions would look like were they needed to form a basis.

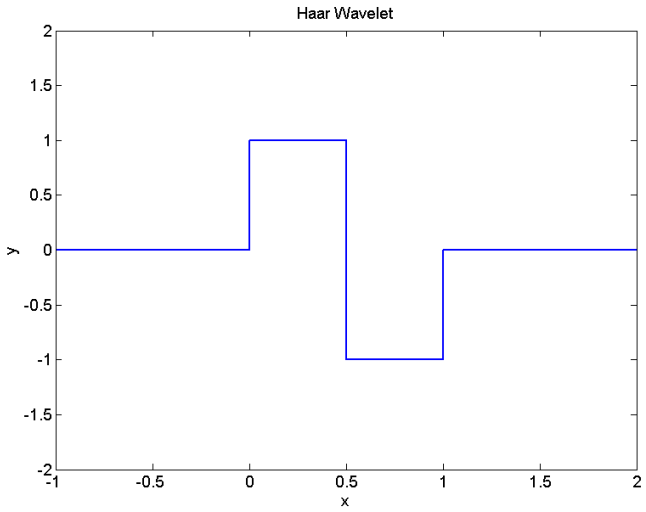


Figure 2.2: Haar wavelet

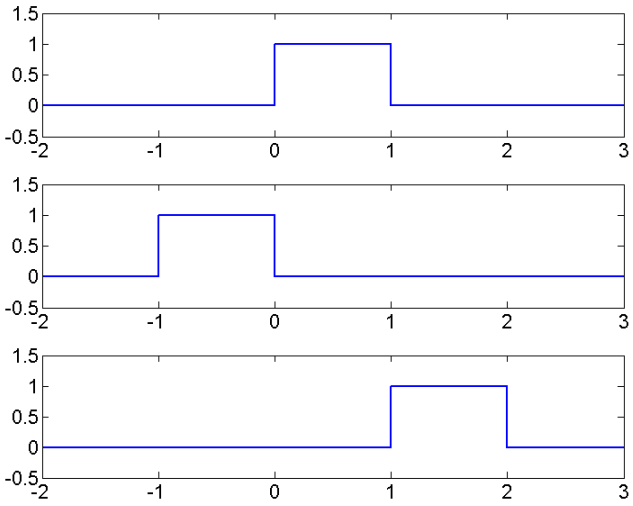


Figure 2.3: Haar scaling function translates, of $\phi(x)$, $\phi(x+1)$ and $\phi(x-1)$

One may notice that when we scale the Haar scaling function, the orthonormal relation may appear to be contradicted. This is remedied by adding a scaling factor, $2^{\frac{m}{2}}$, to the function $\phi(2^m x - k)$, from equation 2.4.

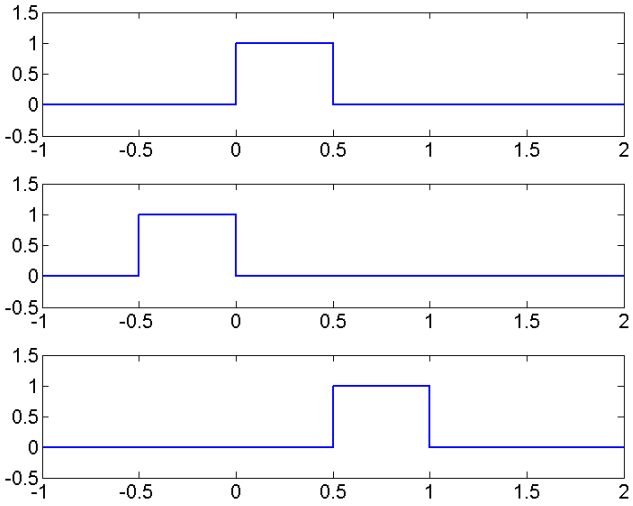


Figure 2.4: Haar scaling functions and the scaled translates, of $\phi(2x)$: $\phi(2x+1)$ and $\phi(2x-1)$

2.1.1 Approximation Example

In order to better understand how a set of basis functions can be used to approximate the value of a function we can use a simple basis function like the Haar to illustrate this method. When working through this example, it can be useful to envision the Haar wavelet as a piecewise constant function, often used to demonstrate standard Galerkin methods like the finite element method.

The method for determining the Haar scaling function approximation is straightforward: discretize the domain of the function then determine the scaling coefficients by computing the average values of the function within each discretized region. Using these scaling function coefficients we can then reconstruct the approximation as a linear combination of the scaling coefficients with the Haar basis functions. Figure 2.5 shows four different approximations of the function $u(x) = -x(x-1)$ using the Haar basis functions.

From Figure 2.5 we can see that the function is approximated to the best ability by a piecewise constant function with the given resolution, shown in the red plot. That being said, there is still a lot of information missing from this approximation. To improve the approximation we can add the orthogonal complement, or the Haar wavelet, to our computation. Section 2.4 will discuss the concept of multi-resolution properties of wavelets but by using this example we can see how adding the wavelet to the scaling function approximation gives us added details of the

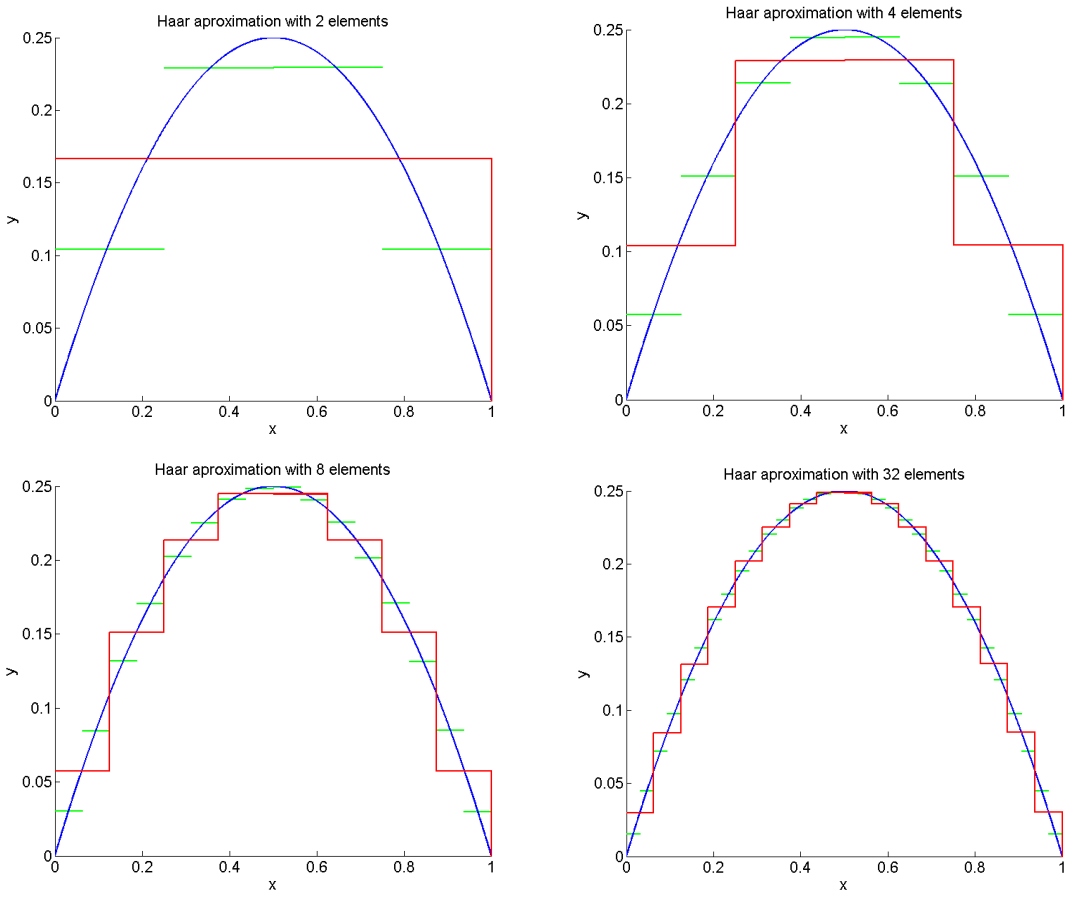


Figure 2.5: Haar basis function approximation at varying resolutions with the wavelet approximation superimposed

solution. Taking the definition of the Haar wavelet in Figure 2.2 we can superimpose the wavelet approximation to the function and see that we will end up with the scaling function approximation at one order higher resolution, this is the green plot in Figure 2.5. Using Figure 2.5 we can see how the difference between each discretization level is the details that would be added in, were the wavelet approximation included. In other words if we define V_n and W_n as the space represented by the scaling and wavelet functions respectively, then V_{n+1} can be approximated using the combination of V_n and W_n.

Using this relationship, it is clear that we can represent more of the missing information by using a scaling/wavelet approximation than by simply using just the scaling function. Thus we can decrease the approximation error by using both the scaling and wavelet functions without increasing

the discretization. We are able to do this because the scaling and wavelet functions are orthogonal as we can plainly tell from Figures 2.1 and 2.2.

As one could imagine, using a collection of Haar scaling functions to approximate more complex valued functions like sine and cosine functions would be a poor choice. The reason for this is that sine and cosine functions have an infinite number of continuous derivatives where as the Haar function has none. But the fact that the scaling functions are orthonormal is of paramount importance and even more so that it can be extended to higher order Daubechies scaling and wavelet functions. Often, the Haar function is referred to as $D2$ indicating that it is the lowest order wavelet in the Daubechies family of wavelets.

2.2 The Daubechies Scaling Function

The Daubechies family of wavelets [6] are of particular interest to us because they have compact support, are orthonormal, and attempt to maximize the highest number of vanishing moments. Daubechies wavelets are constructed to have as narrow of a supported domain as possible while maximizing the number of vanishing moments. In fact, the wavelet genus, or number of vanishing moments, N, is directly related to the domain that is supported: $[0, N-1]$. Often the Daubechies scaling and wavelet functions are referred to in terms of their genus using the format DN, where D denotes the Daubechies family of wavelets and N represents the number of coefficients needed to define the wavelet on the domain $[0, N-1]$. The number of vanishing moments that a wavelet possesses gives us an idea of the order of accuracy that said wavelet can approximate. According to Nielson [14], given a wavelet with p vanishing moments one can exactly represent any polynomial with order $p = \frac{N}{2}$ or less.

As previously mentioned, Daubechies wavelets are formulated by using a scaling or dilation function, sometimes called a father wavelet, to construct the wavelet equation. Strang & Nguyen [17] present an approach of deriving the scaling and wavelet functions. The scaling function for Daubechies wavelets is given in equation 2.1, also known as the dilation equation. The set of a_k coefficients are determined numerically and from equation 2.2, we can then note that since:

$$1 = \int_{-\infty}^{\infty} \phi(x)dx = \int_{-\infty}^{\infty} \sum a_k \phi(2x-k)dx \qquad \text{let} \quad u = 2x-k$$
$$= \frac{1}{2} \sum a_k \int_{-\infty}^{\infty} \phi(u)du$$

clearly $\sum a_k = 2$. These coefficient values are determined such that all moments, in terms of $k = 0, \ldots, N-1$ disappear for a Daubechies wavelet of DN. The moment equation can be given by the relation:

$$\int_{-\infty}^{\infty} x^k w(x)dx = 0 \tag{2.10}$$

Table 2.1 shows the numerically computed values of the coefficients that were initially introduced by Ingrid Daubechies.

Table 2.1: The Daubecheis scaling function coefficients(c_k)

D10	D12	D14	D16	D18	D20
1.60102e-001	1.11541e-001	7.78521e-002	5.44158e-002	3.80779e-002	2.66701e-002
6.03829e-001	4.94624e-001	3.96539e-001	3.12872e-001	2.43835e-001	1.88177e-001
7.24309e-001	7.51134e-001	7.29132e-001	6.75631e-001	6.04823e-001	5.27201e-001
1.38428e-001	3.15250e-001	4.69782e-001	5.85355e-001	6.57288e-001	6.88459e-001
-2.42295e-001	-2.26265e-001	-1.43906e-001	-1.58291e-002	1.33197e-001	2.81172e-001
-3.22449e-002	-1.29767e-001	-2.24036e-001	-2.84016e-001	-2.93274e-001	-2.49846e-001
7.75715e-002	9.75016e-002	7.13092e-002	4.72485e-004	-9.68408e-002	-1.95946e-001
-6.24149e-003	2.75229e-002	8.06126e-002	1.28747e-001	1.48541e-001	1.27369e-001
-1.25808e-002	-3.15820e-002	-3.80299e-002	-1.73693e-002	3.07257e-002	9.30574e-002
3.33573e-003	5.53842e-004	-1.65745e-002	-4.40883e-002	-6.76328e-002	-7.13941e-002
	4.77726e-003	1.25510e-002	1.39810e-002	2.50947e-004	-2.94575e-002
	-1.07730e-003	4.29578e-004	8.74609e-003	2.23617e-002	3.32127e-002
	D8	-1.80164e-003	-4.87035e-003	-4.72320e-003	3.60655e-003
	2.30378e-001	3.53714e-004	-3.91740e-004	-4.28150e-003	-1.07332e-002
	7.14847e-001	D6	6.75449e-004	1.84765e-003	1.39535e-003
	6.30881e-001	3.32671e-001	-1.17477e-004	2.30386e-004	1.99241e-003
	-2.79838e-002	8.06892e-001	D4	-2.51963e-004	-6.85857e-004
	-1.87035e-001	4.59878e-001	4.82963e-001	3.93473e-005	-1.16467e-004
	3.08414e-002	-1.35011e-001	8.36516e-001		9.35887e-005
	3.28830e-002	-8.54413e-002	2.24144e-001		-1.32642e-005
	-1.05974e-002	3.52263e-002	-1.29410e-001		

Often in the literature, these coefficients shown in Table 2.1 will appear normalized either by $\sqrt{2}$ or 2 depending on which set of coefficients is more convenient for the problem at hand. Ingrid Daubechies initially presented these coefficients normalized by $\sqrt{2}$ and many authors today follow her example. Now to keep things straight we will define three different coefficients that illustrate these three different normalizations.

$$\sum a_k = 2 \tag{2.11}$$

$$\sum c_k = \sqrt{2} \tag{2.12}$$

$$\sum h_k = 1 \tag{2.13}$$

For example, the dilation equation, that we will be using to define our basis functions in terms of the coefficients, h_k is:

$$\phi(x) = \sqrt{2} \sum_{k=0}^{N-1} h(k)\phi(2x - k) \tag{2.14}$$

2.3 Generating the Scaling and Wavelet Functions

One may notice that in order to explicitly determine the scaling function values at certain points one must use a recursive technique. Although it is not necessary to represent the Daubechies scaling or wavelet functions explicitly for the purposes of this work it is useful to understand the mechanics. In this section, we will generate some example scaling and wavelet function plots allowing us to get a glimpse as to why it is difficult to represent scaling function derivatives with quadrature rules.

Daubechies scaling functions are only determined at what are called dyadic points. Dyadic points split the domain into a set of points separated by $\frac{1}{2^j}$ where j is an integer or level that represents the coarseness of the mesh. The method used to generate the scaling and wavelet functions for this work initially calculates the function values at the first set of dyadic points, $x \in 0, 1, \cdots, N-2, N-1$, i.e., the integer values. Then the dilation equation is used to generate the values at half points, then quarter points and so on until an acceptable discretization is reached. To calculate the function values at the first set of dyadic points one must solve an eigenvalue problem using the dilation equation at the dyadic points. The linear system in equation 2.15 uses the dilation

equation 2.1 to set up a system of equations in order to solve the first set of dyadic points for the $D4$ scaling function. Using the fact that the support of $D4$ is $[0, 3]$, we can define the linear system:

$$\begin{bmatrix} a_0 & 0 & 0 & 0 \\ a_2 & a_1 & a_0 & 0 \\ 0 & a_3 & a_2 & a_1 \\ 0 & 0 & 0 & a_3 \end{bmatrix} \begin{bmatrix} \phi(0) \\ \phi(1) \\ \phi(2) \\ \phi(3) \end{bmatrix} = \begin{bmatrix} \phi(0) \\ \phi(1) \\ \phi(2) \\ \phi(3) \end{bmatrix} \tag{2.15}$$

So, we wish to find an eigenvector corresponding to the eigenvalue equal to one. Of course there are an infinite number of such vectors, but it turns out that we need the one whose components sum to one [5]. Once the function values at these dyadic points have been generated, it becomes a programming exercise to generate the function values at the next level of dyadic points ($\Delta x = \frac{1}{2}, \frac{1}{4}, \frac{1}{8}$, etc.). Figure 2.6 shows the plots of the scaling function at the first four levels of dyadic points using the $D4$ scaling function, which has a domain of support: $[0, 3]$. This method of constructing the scaling function is often called the Cascade algorithm.

Now that we have presented a method to calculate scaling function values at the dyadic points, we can use these values to determine the Daubechies wavelet function at the dyadic points. Using the general wavelet equation 2.3, we can construct the wavelet function at the dyadic points using the scaling function values at the dyadic points in combination with the coefficient values, a_k.

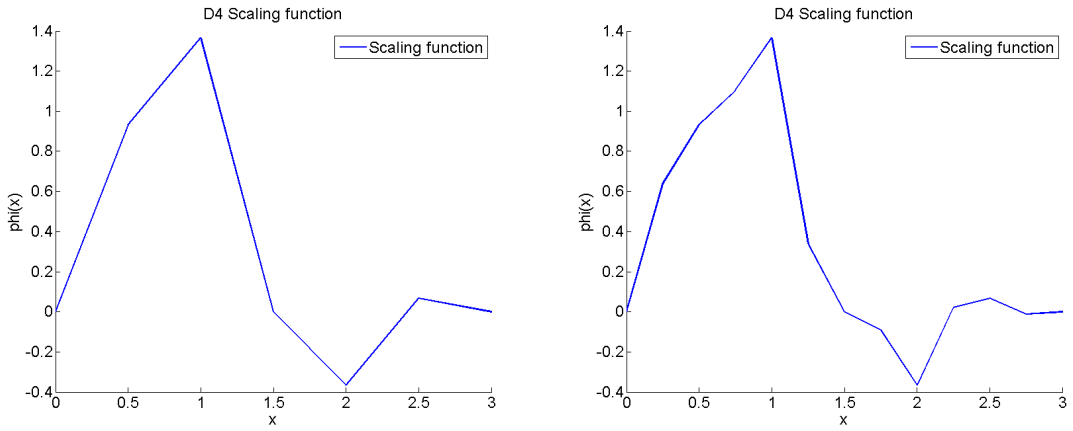
$$w(x) = \sum_{k=0}^{N-1} (-1)^k a_{N-1-k} \phi(2x - k) \tag{2.16}$$

Figure 2.7 shows a few samples of various Daubechies wavelets and scalings functions after 7 levels of dyadic points, or a resolution of $\frac{1}{2^7}$ constructed using the methods outlined in this section.

2.4 Multi-resolution

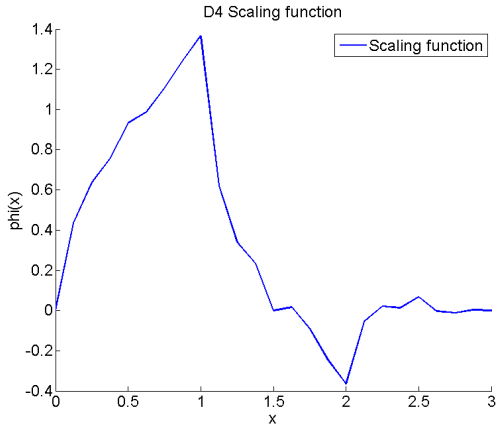
Another attractive feature of implementing a wavelet-Galerkin method is the concept of multi-resolution. Since we are going to be using the Daubechies scaling functions as a basis in our Galerkin approximation, we must first understand what a basis is and how it will be used. Basis functions are defined as functions that are not only linearly independent but also span the space that they occupy.

As we saw in the Haar wavelet example, the translations of a given scaling function at level m are orthogonal, fulfilling the first condition that the functions be linearly independent. The second

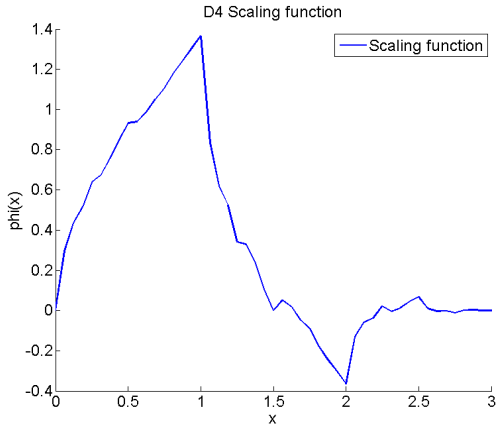


(a) $j = 1$ first level

(b) $j = 2$ second level



(c) $j = 3$ third level



(d) $j = 4$ fourth level

Figure 2.6: Approximation to $D4$ scaling function using dyadic points: $\frac{1}{2}$, $\frac{1}{4}$, $\frac{1}{8}$, $\frac{1}{16}$

condition requires us to define a space that is spanned by the set of orthogonal functions. So, we can define a space, V_m that is spanned by our scaling functions $\phi_{m,k}(x)$ for all k; where k represents all the possible shifts of our scaling function, defined in V_m.

We can say that our set of scaling functions are a basis on V_m, because they are orthogonal. Now if we increase the value of m which can also be thought of as the resolution or level of detail represented by our scaling function, we can apply multi-resolution analysis. Multi-resolution analysis is defined by a number of axioms, the most important in our case being the fact that each scaling function space contains a less resolved space and it is in turn contained by a more resolved

space. This concept can be represented in general by the relation:

$$\cdots \subset V_{m-2} \subset V_{m-1} \subset V_m \subset V_{m+1} \subset V_{m+2} \subset \cdots \tag{2.17}$$

which are all subspaces of $L^2(R)$. For example when $m = 4$ we have a resolution of $\frac{1}{2^4}$ and when $m = 5$ our resolution is $\frac{1}{2^5}$ which contains all the dyadic points from $m = 4$. Previously we showed how one could combine the wavelet function at m with a scaling function approximation at m to get the scaling function approximation at $m+1$. We will now define a general wavelet space, W_m as the space containing all wavelet functions $w_{m,k}$ defined with scaling and translational terms m and k respectively. Also, due to the orthogonality of the wavelet and scaling function, we know that W_m is the orthogonal complement of V_m in L_2. Because the subspaces are nested we can define the orthogonal complement of V_m as W_m in V_{m+1}. Using this we can say:

$$V_{m+1} = V_m \oplus W_m \tag{2.18}$$

Using this relation we can continue to extend it to say that each space, V_m, can be represented by a series of orthogonal complements:

$$V_{m+1} = V_{m-i} \oplus W_{m-i} \oplus W_{m-i+1} \oplus W_{m-i+2} + \cdots W_m \tag{2.19}$$

where i is an arbitrary integer used to show that any subspace of the initial space can be added to the orthogonal complement expression. This shows that the space $L^2(R)$, can be represented using this collection of subspaces to approximate the $L^2(R)$ space. This relation also confirms what we noticed in the Haar example, adding wavelet subspaces to our approximation yields greater details to the solution. And also importantly we can use our scaling functions to analyze a problem at multiple scales.

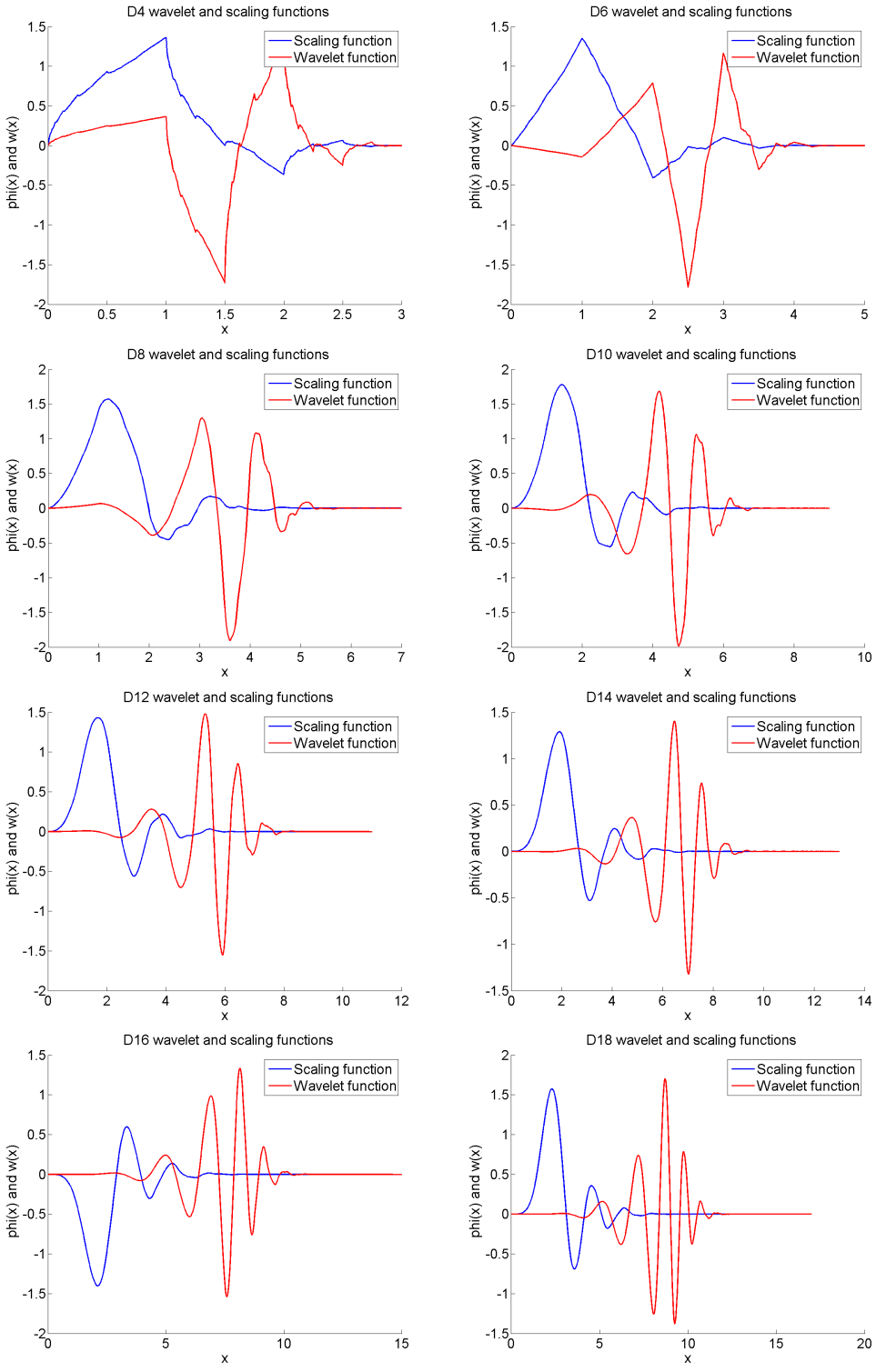


Figure 2.7: Daubechies Wavelet and Scaling functions $D4$ to $D18$

CHAPTER 3

THE GALERKIN FINITE ELEMENT METHOD

The aim of Galerkin methods is to approximate Ordinary and Partial Differential Equations (ODE/PDE's) using a discrete set of basis functions. One of the most commonly used types of Galerkin methods is the Finite Element Method (FEM). Finite element methods were initially designed by engineers for use in structural mechanics, analyzing stresses and strains [4]. The flexibility and adaptability [2] inherent in finite element methods provide a major advantage of the method over many of its counterparts, like the finite difference method. The finite element method typically uses continuous piecewise polynomial functions as a set of basis functions to approximate the solutions to the ODE's and PDE's. Common choices for bases in the finite element method are piecewise linear and quadratic polynomials. Later in this paper we will define how we can use wavelet scaling functions as a basis in our Galerkin approximation. In this chapter we will discuss the formulation and solution of a sample finite element method problem to understand the process of a standard Galerkin method. Additionally reduced order modeling approaches in the context of FEM will be explored in the next chapter.

In this chapter we will first examine the setup and computation of the finite element method while discussing some of the theory surrounding it. We will use Poisson's equation as an example in our preliminary analysis presenting the topics that one must consider when implementing a finite element method like basis functions and quadrature. Then we will show a standard algorithm that can be used to handle the finite element method. This chapter will conclude by demonstrating that the finite element method can be applied to a modified one dimensional heat equation which will act as our test case throughout the remainder of this work.

3.1 Preliminary Definitions

Before we start working through the finite element method and some of the theory surrounding it we must first define some of the notation that will be used in this chapter. The first piece of notation that we must define is the concept of a Hilbert space, H. Hilbert spaces are infinite

dimensional spaces defined with a complete inner product, where the inner product induces a norm on the space. The $L_2(\Omega)$ space, which we are going to be approximating the solution to our differential equation on, is a Hilbert space over Ω which defines an open connected set within $\mathbf{R}^n$. The $L_2(\Omega)$ space has an inner product, $(\cdot,\cdot)_{L_2}$, and norm, $\|\cdot\|_{L_2}$, where we define the inner product as:

$$I = (u, v) = \int_\Omega uvdx \tag{3.1}$$

and the norm as:

$$\|u\|_{L_2} = (u, u)^{\frac{1}{2}} = \left(\int_\Omega u^2 dx\right)^{\frac{1}{2}} \tag{3.2}$$

Then in order to account for functions that have derivatives, we must constrain our Hilbert space to account for only functions that are sufficiently smooth. This type of space is called a Sobolev space which contains only functions that have at least k weak derivatives in L_2 which is denoted as $H^k(\Omega)$. In this work, we will be particularly interested in the H^1 space and the constrained space, H_0^1. To further restrict the Sobolev space we will be solving on, we define $H_0^1(\Omega)$, which consists of all functions in H^1 which satisfy the homogeneous boundary conditions. In H^1 we define the norm:

$$||\cdot||_1 = \left(\int_0^1 u^2 + \int_0^1 (\nabla u)^2\right)^{\frac{1}{2}} \quad \forall u \in H^1(\Omega) \tag{3.3}$$

Finally we must define a bilinear form which is used on our Hilbert space to transform $H \times H$ to $\mathbf{R}^1$. A bilinear form must have the properties:

$$\begin{array}{rl} A(\alpha_1 u_1 + \alpha_2 u_2, v) & = \alpha_1 A(u_1, v) + \alpha_2 A(u_2, v) \;\; \forall\, u_1, u_2, v \;\in H \\ A(u, \beta_1 v_1 + \beta_2 v_2) & = \beta_1 A(u, v_1) + \beta_2 A(u, v_2) \;\; \forall\, u, v_1, v_2 \;\in H \end{array} \tag{3.4}$$

where $A(\cdot,\cdot)$ is our bilinear form and α and β are in $\mathbf{R}^1$. Bilinear forms will be very useful in formulating our weak formulation.

3.2 Poisson's Equation

Before we can use the finite element method, we must first determine the problem we are trying to solve. In order to give a brief overview of the finite element method, the Poisson's equation in one dimension will be used as an example. Poisson's equation is straightforward and will allow us to formulate the weak form, show how to determine inner products and finally solve a linear system

to determine the finite element solution. Equation 3.5 defines Poisson's equation along with its homogeneous Dirichlet boundary conditions.

$$\begin{aligned} -a\frac{d^2u(x)}{dx^2} &= f(x) \quad 0 < x < 1 \\ u(0) = u(1) &= 0 \end{aligned} \tag{3.5}$$

where $u(x)$ is the solution to the ordinary differential equation, $f(x)$ is the forcing function and a is a leading coefficent in the ODE.

3.2.1 The Weak Formulation

Now that the problem has been chosen we must define a weak formulation. The weak formulation allows us to remove some previous restrictions in the standard problem description. Recalling the definition of our Sobolev space, H_0^1, we can write a weak form of our problem as:

$$\begin{aligned} &\text{seek } u \in H_0^1 \text{ such that} \\ &A(u,v) = F(v) \text{ for all } v \in H_0^1 \end{aligned} \tag{3.6}$$

where H_0^1 is the Sobolev space that will contain the solution to our differential equation and $A(u,v)$ is the bilinear form on $H_0^1 \times H_0^1$ such that $A(u,v) = a\int u'v'dx$. Equation 3.7 shows how we can multiply our Poisson equation by a test function v, which is in H_0^1, and integrate over the entire domain to attempt to solve for our solution u, giving us the weak formulation.

$$-a\int_0^1 \frac{\partial^2 u}{\partial x^2}vdx = \int_0^1 f(x)vdx \tag{3.7}$$

Because v is in H_0^1 we can use integration by parts to balance the derivatives on the second order term so that the weak formulation of Poisson's equation can be written as is shown in equation 3.8.

$$a\int_0^1 \frac{\partial u}{\partial x}\frac{\partial v}{\partial x}dx = \int_0^1 f(x)vdx \tag{3.8}$$

which can also be written as the definition of our bilinear form:

$$a\int_0^1 u'v'dx = \int_0^1 fvdx \tag{3.9}$$

Clearly a solution to 3.5 also satisfies 3.6 and the converse is true if u is sufficiently smooth. We know that the Lax-Milgram theorem [11] gives a condition which guarantees the uniqueness of

our solution. Lax Milgram requires that $F(v)$ is a bounded linear functional and the bilinear form $A(\cdot,\cdot)$ satisfies both continuity and coercivity conditions, i.e.,

$$\begin{array}{ll} |A(u,v)| \leq C\|u\|_1\|v\|_1 & \forall\ u,v \in H_0^1 \\ A(u,u) \geq c\|u\|_1^2 & \forall\ u,v \in H_0^1 \end{array} \tag{3.10}$$

where C and c are two positive constants that do not have any relation to u and v. Using the definition of the bilinear form and the definition of the H^1 norm, along with the Cauchy-Schwartz and Poincaré inequalities, it can be shown that the continuity and coercivity conditions are satisfied for our definition of $A(\cdot,\cdot)$. We also note that we can write the definition of the bounded linear functional, $F(v)$ as $F(v) = \int f(x)v dx$.

To discretize we let S_0^h be a finite dimensional subspace of H_0^1. And then we have the discretized weak form:

$$\begin{array}{l} \text{seek } u^h \in S_0^h \text{ such that} \\ A(u^h, v^h) = F(v^h) \text{ for all } v^h \in S_0^h \end{array} \tag{3.11}$$

Now that we have defined our fully discrete weak form of the one dimensional Poisson equation we must define the function u^h so that, using the discrete weak formulation, we can determine an approximation to our differential equation. We hope to be able to transform the discrete weak form into a linear set of equations. In a Galerkin method we accomplish this by defining u^h as a linear combination of a set of basis functions i.e., a set of functions which are linearly independent and span the space. We can write this relation as:

$$u^h = \sum_{j=1}^{n} c_j \phi_j(x) \tag{3.12}$$

where c_j is a constant value that scales our basis function, $\phi_j(x)$. We must also note that because we have defined $u^h \in S_0^h$ and v^h is also defined in S_0^h then v^h must be represented by the same basis functions as u^h. Thus v^h will be represented by our basis functions used to define our solution, $\{\phi(x)\}_{j=1}^n$.

Now using the definition of u^h, we can modify our discrete weak formulation to account for these unknowns:

$$\sum_{j=1}^{n} c_j \left[a \int_0^1 \phi_j'(x)\phi_i'(x)dx \right] = \int_0^1 f(x)\phi_i(x)dx \text{ for } i = 1, \ldots n \tag{3.13}$$

Thus the only unknowns in equation 3.13 are the scaling coefficients c_j, which we will demonstrate that we can solve using a linear system. We must remember that the $\phi_j(x)$ variables represent functions; so in order to determine these inner products a quadrature rule should be used. In the next section we will discuss a choice of basis functions and then in a following section will discuss an appropriate quadrature rule.

3.2.2 Using Piecewise Linear Polynomial Basis Functions

In order to solve the linear system defined in equation 3.13 we must define functions $\{\phi_j(x)\}_{j=0}^{n}$ that fit our basis requirement. Continuous piecewise polynomial basis functions are often used in the finite element method due to their versatility and ease of implementation. One advantage of using piecewise polynomial basis functions is that in a number of their implementations, including the ones we will use in this work, the functions fall into the nodal basis category. This means that each basis element has a value of one at exactly one node and is zero at every other node. Nodes allow us to discretize our continuous domain into a set of equally or variably spaced elements (space in between nodes) which is necessary for piecewise polynomials. There are some implementations within the piecewise polynomial category that do no have nodal basis functions, like cubic splines, but these will not be analyzed in this work.

On the domain $[0, 1]$ we can discretize using nodes, x_i such that $0 = x_0 < x_1 < \ldots < x_n < x_{n+1} = 1$. The most basic of continuous piecewise polynomials is, of course, the linear function. A nodal basis known as the "hat function" can be given by equation 3.14.

$$\phi_i(x) = \begin{cases} \frac{x - x_{i-1}}{x_i - x_{i-1}} & \text{for } x_{i-1} \leq x \leq x_i \\ \frac{x_{i+1} - x}{x_{i+1} - x_i} & \text{for } x_i \leq x \leq x_{i+1} \\ 0 & \text{for } x < x_{i-1} \text{ and } x > x_{i+1} \end{cases} \tag{3.14}$$

From this equation it is clear that the basis function is a nodal basis; where it starts at zero, increases to one and then decreases back to zero. It is important to notice that the support of this linear basis function spans over two elements between $[x_{i-1}, x_i]$ and $[x_i, x_{i+1}]$. Figure 3.1 shows a set of piecewise linear basis functions with a uniform spacing of 0.2 that we could be used on our domain of $[0, 1]$.

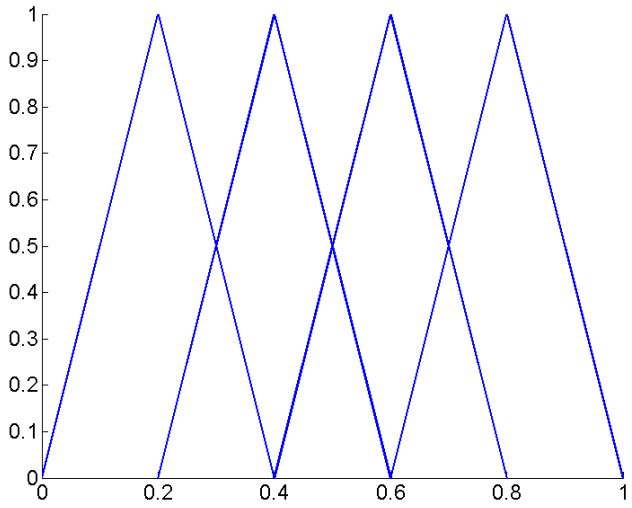


Figure 3.1: Piecewise linear basis functions on $[0, 1]$

3.2.3 Error Estimate

Before we move to the next section on quadrature we will briefly discuss the error we would expect to encounter by implementing a finite element method with piecewise linear basis functions. Our aim will be to find a way to quantify the error in terms of our discretization, h. We will start by noting Galerkin or Cea's lemma which, assuming the conditions of Lax-Milgram provides the statement:

$$\|u - u^h\|_H \leq \frac{C}{c} \inf_{\chi^h \in V^h} \|u - \chi^h\|_H \tag{3.15}$$

where c and C are the constants that appeared in Lax-Milgram, V^h is a family of finite dimensional subspaces of the underlying Hilbert space and $\|\cdot\|_H$ defines the norm on H. This relation shows that the solution u^h, which is in V^h is the best approximation to u on V^h. Using interpolation theory we can define the V^h-interpolant of u as $I^h u$. And because we know the result of Galerkin's lemma we can state that the error in the interpolant of u must be greater than or equal to the best approximation to u using the relation:

$$\inf_{\xi^h \in V^h} \|u - \chi^h\|_H \leq \|u - I^h u\|_H \tag{3.16}$$

which can then be related back to Galerkin's lemma so that we can state:

$$\|u - u^h\|_H \leq \frac{C}{c}\|u - I^h u\|_H \tag{3.17}$$

Now our goal becomes finding a way that we can bound the error in the interpolant using the discretization, h. If we are able to do this we will have a way to relate the error to the discretization. Using approximation theory we know that we can relate the error of the interpolant to the smoothness of the function we are trying to interpolate. Thus, for the H^1 norm, we can write the relation:

$$\|u - I^h u\|_1 \leq b_1 h^k \|u\|_{k+1} \tag{3.18}$$

where b_1 is an arbitrary positive constant and again V^h defines the set of piecewise polynomials of order $\leq k$ with $u \in H^{k+1}$. Then using this relation and 3.17 we can write:

$$\|u - u^h\|_1 \leq \beta_1 h^k \|u\|_{k+1} \tag{3.19}$$

Using relation 3.19 we can now relate the error in our finite element approximation to the discretization for the H_0^1 norm. In order to get an approximation for the L_2 error we need to make use of the Aubin-Nitsche lemma also known as "Nitsche's trick". If we assume that u is sufficiently smooth ($u \in H^2$), then Nitsche's trick allows us to state:

$$\|u - u^h\|_0 \leq \beta_2 h^{k+1} \|u\|_{k+1} \tag{3.20}$$

where b_2 is a positive arbitrary constant and the H^0 space is often referred to as L_2.

This analysis shows that if we use piecewise linear basis functions to approximate a solution $u \in H^2$ then the error will behave with $\mathcal{O}(h)$ accuracy in the H^1 norm and $\mathcal{O}(h^2)$ in the L_2 error norm.

In order to numerically understand the ability of our finite element method with piecewise linear basis functions to approximate the solution we must compute the rate of convergence. The rate of convergence is a measure of how much the error decreases as we increase the discretization of our domain and should converge towards the exponent of the error term. The equation used to solve for this rate can be given by the following formula.

$$Rate_i = \frac{\log \frac{E_{i-1}}{E_i}}{\log \frac{\Delta x_{i-1}}{\Delta x_i}} \tag{3.21}$$

where E represents the error with whatever measure you desire and Δx refers to the spatial discretization. So we should expect that the rate of convergence using piecewise linear basis functions is one and two for the H^1 and L_2 error norms respectively. Were we to require our finite element method to have a higher order of accuracy, we could use higher order piecewise polynomials, but for our sample problem we will stick to using piecewise linear basis functions.

3.2.4 Quadrature Rules

The weak formulation of our problem (equation 3.8) requires us to calculate the inner products, defined in equation 3.1, of our basis functions a number of times. For the piecewise linear case, this is not difficult, in fact with a little manipulation we can compute the inner products of our basis functions exactly. The problem is that with other basis functions and variable coefficients, the algebra required to resolve this integral exactly becomes very complex and in some cases impossible; in addition we also have to compute (f, ϕ_i) which we may not be able to do exactly. To avoid this problem we can use quadrature rules. A quadrature rule is a way of approximating an integral with a summation of the function values at particular points multiplied by predetermined weights.

$$\int f(x)dx \approx \sum_{i=0}^{n-1} w_i f(x_i) \tag{3.22}$$

where w_i is the predetermined weight to use. There are many quadrature methods that one can choose and some may work better for different types of integrands. One of the most commonly used quadrature methods is Gaussian quadrature. Gaussian quadrature was constructed to be able to represent polynomials of order $2n-1$ exactly, assuming n points. So, because we are only using piecewise linear basis functions in our problem we will need to use a single quadrature point on each element. This greatly decreases the complexity of our problem as we only need to calculate the value of the function at a single point and then multiply that by a weight value that can be stored.

3.2.5 The FEM Algorithm

Because the finite element method has been around for such a long time, a large number of algorithms have emerged that decrease the computational intensity of the method. Object oriented and parallel finite element codes have proven advantageous in a number of applications from structural mechanics to fluid flow. But for this research we will present a general method for computing the finite element method approximation with a straight-forward implementation. This method will be useful because the way it is set up allows for easy integration of higher order basis functions and non-uniform element sizing.

First we will define the matrix that we would like to solve for and how it should be constructed. The stiffness matrix, whose name comes from structural mechanics defines the derivative term that we are interested in:

$$S_{i,j} = \int_0^1 \phi_j'(x)\phi_i'(x)dx \tag{3.23}$$

which we can use to write our linear system:

$$aS\vec{c} = F \tag{3.24}$$

Using the definition of the stiffness matrix, we can write the general matrix form that we are interested in:

$$S = \begin{bmatrix} \int \phi_1'^2 & \int \phi_2'\phi_1' & 0 & & \cdots \\ \int \phi_1'\phi_2' & \int \phi_2'^2 & \int \phi_3'\phi_2' & 0 & \cdots \\ 0 & \int \phi_1'\phi_2' & \int \phi_2'^2 & \int \phi_3'\phi_2' \quad 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots \quad \vdots & \ddots \end{bmatrix} \tag{3.25}$$

We should take notice that this matrix is a tri-diagonal system. This comes about because we know that our basis functions are only defined over two elements and everywhere else they must be zero. Thus for each piecewise linear basis function we have, only the two adjacent to it will have non-zero inner products. This will be useful as it will reduce the computational cost to solve the linear system.

From the algorithmic point of view we will want to have a routine to assemble this S matrix as well as the right hand side. Below is some pseudo-code that is used as the basis to assemble our S matrix and right hand side vector. We will need to know some information about the geometry including vectors keeping track of: the element size, the relation between local and global nodes

Table 3.1: One dimensional Poisson equation solved with piecewise linear basis functions

h	L_2 Error	H^1 error	L_2 rate	H^1 rate
0.250000	3.81143e-002	3.57550e-001		
0.125000	9.61116e-003	1.78237e-001	1.987551	1.004344
0.062500	2.40786e-003	8.90554e-002	1.996962	1.001026
0.031250	6.02260e-004	4.45199e-002	1.999291	1.000254
0.015625	1.50564e-004	2.22589e-002	2.000010	1.000065

and the nodes for which we are trying to solve (unknown nodes). Also we will need a routine to calculate the value of the basis function and its derivative at a given point, this routine will be called: `phi`.

```
Loop over all the elements in the domain (ie=1:nelements)
  Loop over all the quadrature points in the element (iq=q:nquad)
    Loop over all the basis functions in the element (il=1:nbasis)
      Determine the unknown node corresponding to basis function il
      F(iu) += element spacing*f(quad point)*phi(il,quad point)
      Loop over all the basis functions in the element (jl=1:b=nbasis)
        Determine the unknown node corresponding to basis function jl
        S(iu,ju) += element spacing*(
                    a*phi'(il,quad point)*phi'(jl,quad point))
```

Using this algorithm we can construct our linear system and all that is left to do is solve.

3.2.6 Results

Now we have everything we need to solve the one dimensional Poisson equation with homogeneous Dirichlet boundary conditions. Because this equation is not the objective of this work we will merely present a rate of convergence table confirming that the finite element method is approximating our differential equation at the expected convergence rate. Table 3.1 shows the convergence rate of the finite element method with linear basis functions assuming zero boundary conditions with $f(x) = \pi^2 \sin(\pi x)$ whose exact solution is $u(x) = \sin(\pi x)$.

From Table 3.1 we see that the convergence rates we expect to get when using piecewise linear basis functions is validated. And now that we understand the abilities and uses of the finite element method we can dive into some more complex problems.

3.3 The One Dimensional Heat Equation

The sample problem that we will attempt to solve throughout the remainder of this work is a modified form of the heat equation in one dimension with homogeneous Dirichlet boundary conditions. The heat equation is a parabolic partial differential equation that is typically used to measure the heat distribution within a given domain. One can think of the one dimensional heat equation with homogeneous Dirichlet boundaries as a slice taken from a homogeneous material with ends kept at constant temperature but non-constant flux. Unfortunately we cannot use the Lax-Milgram theorem to guarantee the existence and uniqueness of the weak solution to the heat equation; so instead we reference Thomée [18] for the analysis. The modified 1D heat equation can be written in equation 3.26.

$$\begin{aligned} a_1\frac{\partial u}{\partial t} - a_2\frac{\partial^2 u}{\partial x^2} + a_3 u &= f(x,t) \text{ for } 0 < x < 1,\ 0 < t \leq T \\ u(0,t) = u(1,t) &= 0 \text{ for } 0 < t \leq T \\ u(x,0) &= u_0(x) \text{ for } 0 < x < 1 \end{aligned} \tag{3.26}$$

where $u(x,t)$ is the continuous solution to the PDE in space (x) and time (t) with a forcing function $f(x,t)$, where $u_0(x)$ is a given initial condition and our scalar coefficients a_1, a_2 and a_3. We also note that the general heat equation does not include the non derivative term $a_3 u$. We added this term to increase the number of parameters that could be varied for our reduced order model. Using the finite element method we can solve this PDE with respect to space at a fixed time in terms of a linear combination of basis functions.

3.3.1 Weak Formulation

Now that we have added two extra terms to our differential equation we must account for these terms in our weak formulation. But first for a fixed time t^k we can write the general weak problem as:

$$\begin{aligned} &\text{seek } u(x,t^k) \in H_0^1 \\ &a_1\int_0^1 u_t v dx + a_2\int_0^1 u'v'dx + a_3\int_0^1 uvdx = \int_0^1 fvdx \ \ \forall v \in H_0^1 \end{aligned} \tag{3.27}$$

The major term to take care of before we can write the final discrete weak form of the modified one dimensional heat equation is to decide what to do with the temporal derivative. There are really two main options on how to handle the derivative: we can define the solution vector of c_j's

as dependent on time or we can use a finite difference method to approximate the derivative. The first way would require us to solve a system of ordinary differential equations at each time step. Because the focus of this work is to demonstrate the ability to use a reduced order model on a standard Galerkin method, we will implement the straightforward finite difference method. So we will use the first order backward difference method to approximate the temporal derivative which will preserve stability. The backward difference approximation can be written as:

$$\frac{\partial u(x,t)}{\partial t} \approx \frac{u(x,t) - u(x,t-\Delta t)}{\Delta t} \tag{3.28}$$

with some error term of order Δt added on to the approximation. Using this backward difference approximation we can generate our fully discrete weak problem. Then by defining $U^k, v^h \in S_0^h$ we will seek U^k, using the notation $U^k \approx u(x,t^k)$:

$$\begin{aligned} a_1 \frac{1}{\Delta t}\int_0^1 U^k v^h dx - a_1 \frac{1}{\Delta t}\int_0^1 U^{k-1} v^h dx + a_2 \int_0^1 U_x^k v_x dx + a_3 \int_0^1 U^k v dx = \\ \int_0^1 f(x,t) v dx \end{aligned} \tag{3.29}$$

which is the fully discrete weak problem. Then if we let $\{\phi_i\}_{j=1}^n$ be a basis for S_0^h, where $U^k = \sum_{j=1}^n C_j^k \phi_j(x)$, we can write:

$$\begin{aligned} \sum_{j=1}^n C_j^k \left[a_1 \frac{1}{\Delta t}\int_0^1 \phi_j(x)\phi_i(x)dx + a_2 \int_0^1 \phi_j'(x)\phi_i'(x)dx + a_3 \int_0^1 \phi_j(x)\phi_i(x)dx \right] \\ = \int_0^1 f(x,t)\phi_i(x)dx + a_1 \frac{1}{\Delta t}\int_0^1 u^{k-1}\phi_i(x)dx \end{aligned} \tag{3.30}$$

and the only unknowns in equation 3.30 are the scaling coefficients C_j^k, which can be solved for using a linear system in a similar way as the Poisson's equation.

3.3.2 FEM Algorithm for Time Dependent Problems

The difference between the Poisson and our modified heat equation is that the heat equation includes a time term that we must discretize using a finite difference scheme. From the weak problem in equation 3.30 we notice that there seem to be three inner product terms. But if we look a little closer we can see that the temporal derivative and the non-derivative terms can be combined. We can also write this fully-discrete weak problem in terms of a linear system of equations. The

difference being that we now must introduce the mass matrix, M, to our linear system. We define the elements of our mass matrix as:

$$M_{i,j} = \int_0^1 \phi_j \phi_i dx \tag{3.31}$$

then by factoring out the coefficients we have the linear system:

$$\left(\frac{a_1}{\Delta t} + a_3\right) M\vec{c} + a_2 S\vec{c} = F \tag{3.32}$$

where the mass matrix M also derives its name from structural mechanics. These two matrices will need to be constructed according to the inner products they represent. The mass matrix can be constructed in the form:

$$M = \begin{bmatrix} \int \phi_1^2 & \int \phi_2\phi_1 & 0 & & \cdots \\ \int \phi_1\phi_2 & \int \phi_2^2 & \int \phi_3\phi_2 & 0 & \cdots \\ 0 & \int \phi_1\phi_2 & \int \phi_2^2 & \int \phi_3\phi_2 \quad 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots \quad \vdots & \ddots \end{bmatrix} \tag{3.33}$$

And then there are only a few minor changes that need to be made including modificaiton of the algorithm to now account for the added terms. We must add a time stepping loop as well as find a way to keep track of the solution at previous time steps. We can write a general pseudo-code to determine an A matrix, representing the sum of M and S, at a fixed time step:

```
Loop over all the elements in the domain (ie=1:nelements)
  Loop over all the quadrature points in the element (iq=q:nquad)
    Loop over all the basis functions in the element (il=1:nbasis)
      Determine the unknown node corresponding to basis function il
      Determine the value of the computed solution the previous
          time step (ukm1) at our quadrature point
      F(iu) += element spacing*f(quad point)*phi(il,quad point) + ...
               (1/dt)*a1*ukm1*phi(il,quad point)
      Loop over all the basis functions in the element (jl=1:b=nbasis)
        Determine the unknown node corresponding to basis function jl
        A(iu,ju) += element spacing*(
                    (1/dt)*a1*phi(il,quad point)*phi(jl,quad point) + ...
                    a2*phi'(il,quad point)*phi'(jl,quad point) + ...
                    a3*phi(il,quad point)*phi(jl,quad point))
```

Table 3.2: Finite element convergence rates using piecewise-linear basis functions

h	L_2 Error	L_2 Rate	Euclidean Error	Euclidean Rate
0.250000	1.984484e-001		1.110977e-001	
0.125000	5.068219e-002	1.969213e+000	2.230613e-002	2.316318e+000
0.062500	1.272618e-002	1.993680e+000	5.270417e-003	2.081451e+000
0.031250	3.184845e-003	1.998504e+000	1.298985e-003	2.020532e+000
0.015625	7.964149e-004	1.999631e+000	3.235905e-004	2.005144e+000
0.007813	1.991164e-004	1.999908e+000	8.082550e-005	2.001287e+000

3.3.3 Numerical Results

Now that we understand how to work our way through the finite element method we should present some results on our sample test problem. Using the definition for the modified heat equation defined in equation 3.26 we can set the values of our coefficients and right hand side. For the test problem we will define our coefficients as $a_1 = a_2 = a_3 = 1$, an initial condition of $u_0 = 0$ and the right hand side as $f(x) = (1 + 4t\pi^2 + t)\sin(2\pi x)$. With this definition it can be shown that the exact solution is $u(x,t) = t\sin(2\pi x)$. We will use the H^1 and L_2 error measures whose expected results can be found in the Poisson example section.

Also in order to get the best approximation to the actual solution we must take care to select the values of Δt and Δx appropriately. As we know the backwards difference approximation of our temporal derivative has error of order Δt whereas we found that the error in our finite element method had order Δx^2 in L_2 for a sufficiently smooth u. So in order to get the best rate of convergence, we must set $\Delta t = \Delta x^2$ so that the error depends on Δx^2 and not Δt.

One other error measure that will be captured is the Euclidean error which measures the point-wise error between the actual and computed solutions. Although we know that our approximation to the solution is a continuous function and it should be treated as such, we must use this error to compare to our Wavelet-Galerkin scheme that will be explained in the coming chapters. The Euclidean error can also be thought of as the distance between the two solution vectors: actual and computational. We must be sure to normalize the Euclidean error by the Euclidean norm of the exact solution in order to compare errors at higher resolutions. Table 3.2 shows the convergence rates that can be computed using the finite element method with piecewise linear basis functions.

Using piecewise linear basis functions we can expect a convergence rate of 2 in L_2, which is what we observe from the results. In order to increase this convergence rate we would need to increase the order of the basis function we are using which also increases the complexity of the problem and the bandwidth of the linear system we must solve. Figure 3.2 shows the comparison between the finite element solutions we computed and the exact solution at a few different domain discretizations.

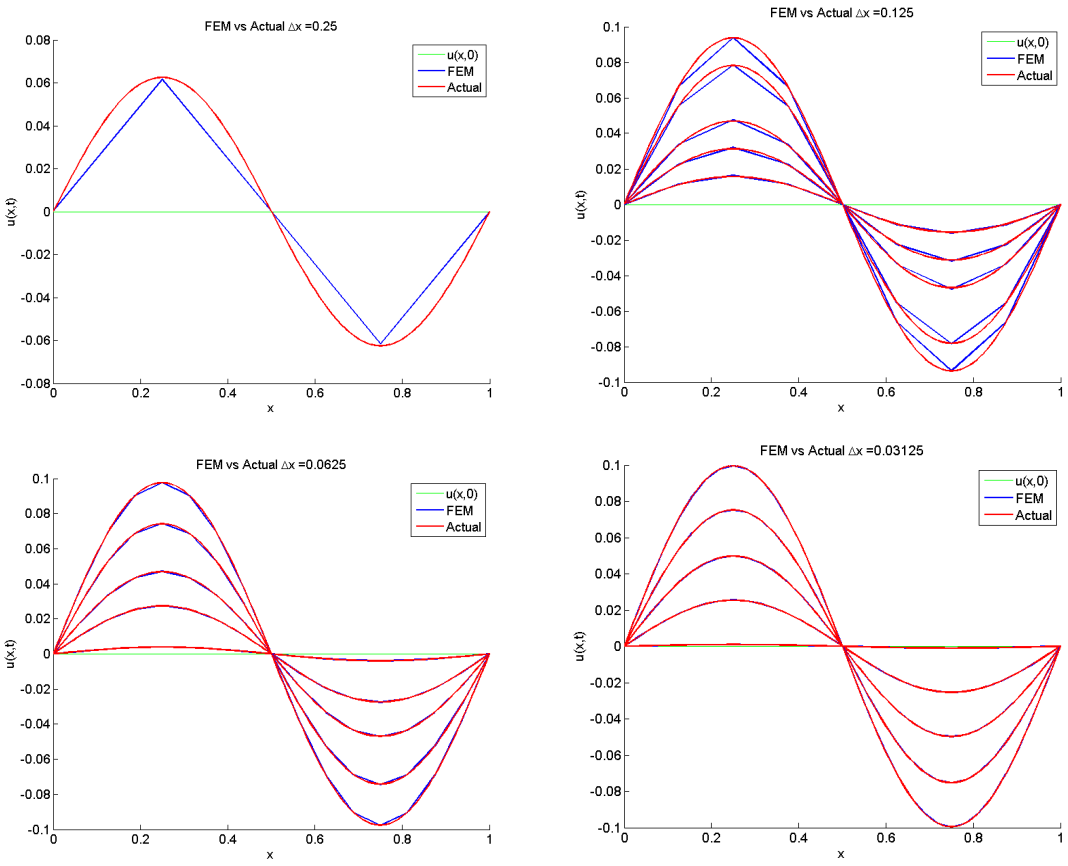


Figure 3.2: FEM solution to our modified one-dimensional heat equation vs. actual solution at 5 time instances

From these plots we can see that as the spatial discretization is increased, and consequently the temporal discretization, the finite element solution converges to the actual solution. Each of these solutions have been generated using the time domain, $t = [0, 0.1]$, capturing five instances of the solution within that range. Because the temporal discretization is so small for the first

plot, only one time instance in our temporal domain is realized. Every solution used the temporal discretization $\Delta t = \Delta x^2$ to preserve the convergence rate of our finite element method.

CHAPTER 4

REDUCED ORDER MODELING

As one could imagine, as the size and accuracy of the problem we are trying to solve increases, the computational complexity also increases. So recently a number of techniques have emerged aimed at decreasing the complexity of large problems via model reduction [9][15]. Model reduction in our case will be defined as the process of solving our differential equation which involves a given set of parameters, using a small set of basis vectors which were created from a large set of solution snapshots. These solution snapshots are precomputed in order to capture the behavior of the solution to the differential equation for a range of input parameters. A commonly used method to generate a basis is the Proper Orthogonal Decomposition or POD [7].

In order to formulate a reduced order model of a differential equation, there are a number of steps that we must go through including: sampling the parameter space, generating solution snapshots with the given input parameters, generating orthonormal functions using POD and then using these orthonormal functions as a basis in a Galerkin method. The first step, sampling the parameter space, requires knowledge of the range of input parameters to sample from and an appropriate sampling method for the parameter space. The parameters being referred to in the case of this work are the coefficient values of our differential equation. There are a number of different sampling method including Latin Hypercube Sampling (LHS) and Halton sequences[9]. Then once an acceptable parameter set has been defined the next step is to generate solution vectors at each of the parameter instances for a particular predefined time domain in order to understand the behavior of the differential equation solutions. Because we are using a time dependent problem, one must select particular solutions in time for each parameter instance to capture the behavior of the solution through time. After determining this set of solution vectors, called snapshots, the next step is to determine basis functions to describe the major modes of the solutions. To do this a number of methods can be used like the Centroidal Voronoi Tesselation (CVT) as well as the Proper Orthogonal Decomposition (POD) which will be used in this work. There are some methods like CVT that do not produce orthonormal basis functions but still capture some of the main features

of the solution and can always be made to be orthogonal by post-processing. Finally, once the orthonormal basis functions have been constructed one must use these basis functions to construct the solution via the Galerkin method used in snapshot generation. It is important to notice that these basis functions, in general, no longer have compact support. So we depend on the fact that the linear system we must solve is significantly smaller than the linear system required to solve the Galerkin linear system using piecewise basis functions.

This chapter will explain a general method of formulating a reduced order model and its application to the finite element method using the Proper Orthogonal Decomposition (POD). The first section will present the first three steps toward formulating the reduced order model including: sampling the parameter space, determining solution snapshots of the differential equation and then creating orthonormal reduced basis functions to describe the behavior of the differential equation solution. This chapter will then conclude with an explanation on how we construct the reduced order approximation using the Galerkin finite element method. Throughout this section we will use our simple linear modified heat equation example to work through the process of setting up a reduced order model.

4.1 Generating Reduced Basis Vectors using FEM

The first thing to be done before reduced basis vectors can be generated and the reduced order approximation can be constructed is the sampling of the parameter space. A number of different methods including LHS and Halton sequences are often used in defining input parameters from a parameter space. Because the main objective of this work is to demonstrate the feasibility of a wavelet based reduced order model we will not focus on using a specific method to determine the parameter space. So for this problem we will collect parameter snapshots using a uniform sampling method.

The next step of applying a reduced order model to a finite element problem is the collection of snapshots. Snapshots are a set of solutions to a given differential equation using the sampled parameter space we have just defined. First, we must recall our general one dimensional modified heat equation with Dirichlet boundary conditions from equation 3.26. We will modify the coefficient values a_2 and a_3 in order to define our parameter space and then generate our snapshots. In order to thoroughly test our reduced order approach and generate a non-trivial solution to the differential

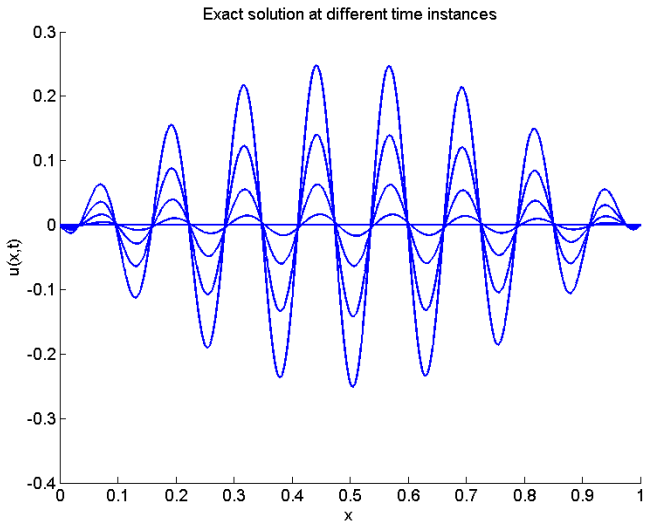


Figure 4.1: Actual solution at $t = 0, 0.025, 0.05, 0.075, 0.1$

equation we will set $f(x,t)$ to:

$$\begin{aligned} f(x,t) =& x(x-1)\left[\sin(50x) + 200t\cos(50x)\right] \\ & - t\left[2\left(\sin(50x) + 100t\cos(50x)\right) + 100(x-1)\left(\cos(50x) - 100t\sin(50x)\right)\right. \\ & \left. +100x\left(\cos(100x) - 100t\sin(50x)\right) + (50^2)x(x-1)\left(-\sin(50x) - 100t\cos(50x)\right)\right] \\ & + tx(x-1)\left(\sin(50x) + 100t\cos(50x)\right) \end{aligned} \tag{4.1}$$

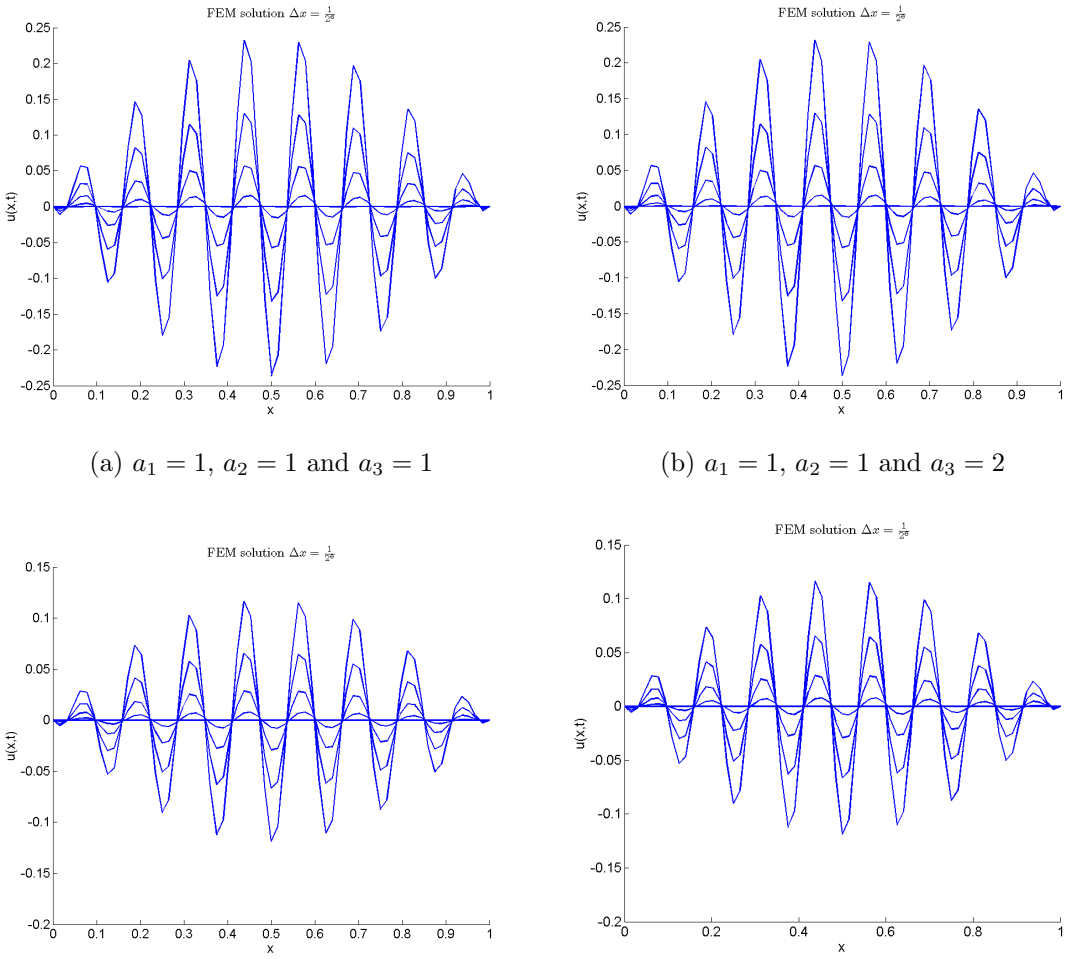
With parameters $a_1 = a_2 = a_3 = 1$ the exact solution can be show to be:

$$u(x,t) = tx(x-1)\left[\sin(50x) + 100t\cos(50x)\right] \tag{4.2}$$

which describes a highly oscillatory function that is zero at the boundaries. Figure 4.1 shows the exact solution to the differential equation on the domain $[0,1]$ for parameters $a_1 = a_2 = a_3 = 1$.

Using this choice of $f(x,t)$ in the modified heat equation along with the sampled parameter set, we can then go about gathering the snapshot set that will be used to define our reduced order model. Figure 4.2 shows some example finite element solutions to the differential equation with four different parameter sets at 5 discrete time instances, whose solutions could be extracted to form the snapshot set. To generate our snapshot set our parameters a_2 and a_3 were varied uniformly between $[0,2]$ with a step size of 1. Only the second two parameters are varied due to the fact that the parameters can be scaled with respect to a_1. Because this is a time dependent problem we

must also find a way to capture information about the solution at different points in time as well. Since we already have 9 instances of the parameter set, 5 random instances in time will be picked to include in the snapshot set bringing our total number of snapshots to 45. Because this is a linear problem with a relatively simple solution we expect 45 snapshots to be adequate in capturing most of the information about the differential equation. The solutions are then stored in column ordered matrix that will be decomposed using SVD.



(a) $a_1 = 1$, $a_2 = 1$ and $a_3 = 1$

(b) $a_1 = 1$, $a_2 = 1$ and $a_3 = 2$

(c) $a_1 = 1$, $a_2 = 2$ and $a_3 = 1$

(d) $a_1 = 1$, $a_2 = 2$ and $a_3 = 2$

Figure 4.2: Sample FEM snapshots with $t = 0, 0.025, 0.050, 0.075, 0.100$

Now, if we store our solution vectors columnwise in some matrix, A, we can use the Singular Value Decomposition (SVD) to extract the basis vectors or the most important components of the

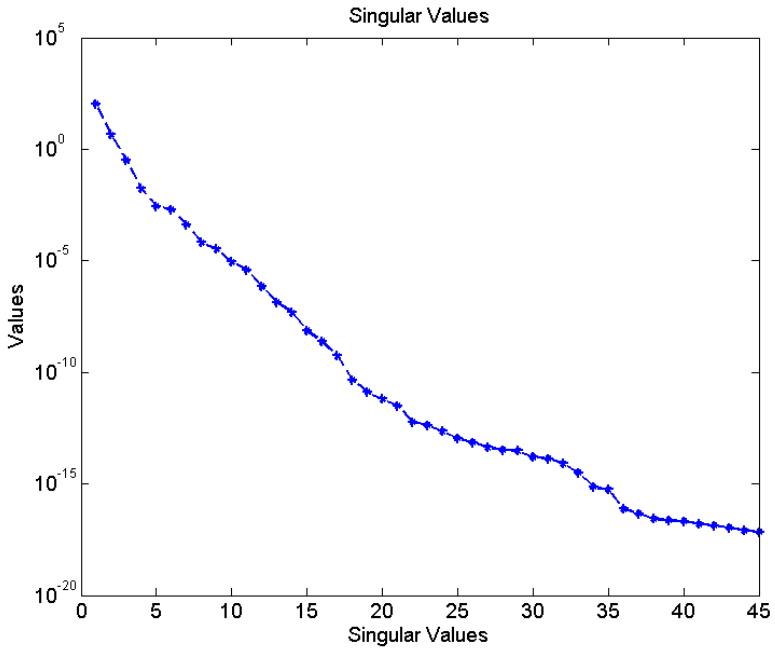


Figure 4.3: Singular values of FEM snapshot matrix

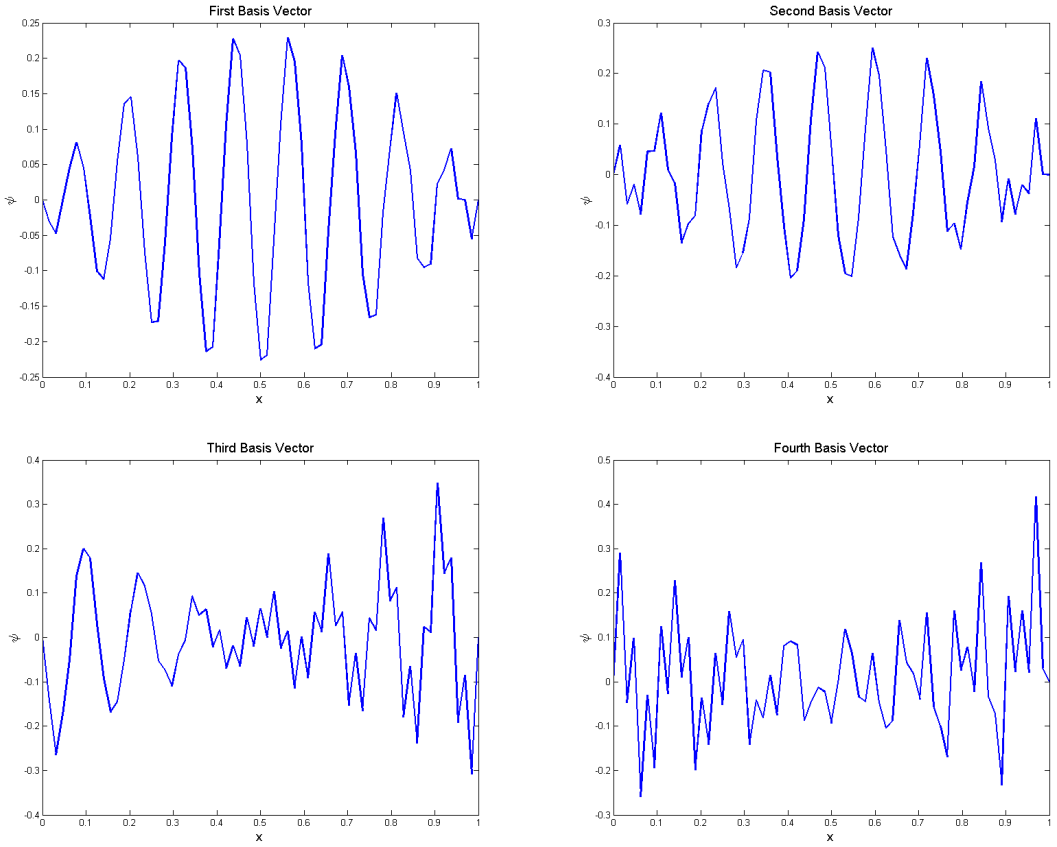
solution. For terminologies sake we refer to this step as the POD even though we are actually using SVD to decompose the solution vectors. Again since this is a linear problem, we expect there to be only a few basis vectors that have any major effect on the solution. The SVD can be written as:

$$A = USV^T \tag{4.3}$$

where U and V represent the column and row space of A respectively and are also orthogonal matrices, and S is a diagonal matrix that contains the ordered set of singular values from the decomposition. The singular values can be thought of as the importance of the columns of U and V and give an idea of how well the matrix A can be reconstructed. It is important to note that only the interior nodal information from the snapshots was included in the SVD. This is because we are using a homogeneous Dirichlet boundary condition. For non-homogeneous boundary conditions we will refer to the work done by Gunzburger et al. [9]. The linearity of this problem should mean that there should only be a few singular values that are effectively non-zero. The singular values of A can be plotted logarithmically as is shown in Figure 4.3. We notice that only the first few

singular values have values of any real significance, and by the tenth singular value the magnitude is less than 10^{-5}.

Because we wish to capture the columnwise information from our snapshot matrix A, we only really need the U matrix to determine our reduced basis functions. Using the U matrix, we can extract and plot the first four basis vectors to get an idea of the major themes in our finite element solution.



(a) $a_1 = 1$, $a_2 = 2$ and $a_3 = 0$

(b) $a_1 = 1$, $a_2 = 2$ and $a_3 = 2$

Figure 4.4: First four basis vectors calculated using SVD

From the plots in Figure 4.4 we can see that the first two basis vectors captures the majority of the general shape of the solution. We also note that the third basis vector seems to account for more of the oscillation near the boundaries but by the time we get to the fourth basis vector we notice that it is mostly noise. It is important to notice that when we wish to reconstruct our

solutions with our new basis functions we cannot simply use the nodal values because our basis is no longer a nodal basis function as it is nonzero at more than one node.

4.2 Solution Approximation using Reduced Basis

Now that we have the set of reduced basis vectors we can then reconstruct a solution to the differential equation using any set of parameters that lies within our parameter space. This parameter set must lie within the defined parameter space that we defined in our snapshot generation because otherwise we would be extrapolating a solution from a data set that we had no information about. To reconstruct the reduced order model solution we can use practically the same method that was used to generate a finite element approximation to the solution of the differential equation. The major difference is that our basis functions are now no longer compactly supported and now exist over our entire domain. Also we note that our basis functions are now orthogonal and are defined in terms of our piecewise FEM basis functions. Because the reduced basis functions we determined will, in general, be defined at every node, the linear system we must solve is typically dense. So it is important that we be able to say that the cost of solving a small dense linear system is smaller than being able to solve a larger sparse linear system. For this linear finite element test problem we will note that this condition is not going to be met as solving a tri-diagonal system is not computationally expensive. The idea here is to show that this is a viable approach that could be applied to a more complex or nonlinear problem where this condition could be met.

First we must define the reduced order model solution, assuming we will use a set number of N basis functions, as a linear combination of some coefficient, α_j, and the reduced basis vectors, ψ_j for $j = 1, \dots N$. But each reduced basis vector, ψ_j, is defined in terms of our piecewise polynomial basis function used to generate our snapshots. Then we have:

$$u^k_{ROM} = \sum_{j=1}^{N} \alpha^k_j \psi_j(x) \text{ where } \psi_j(x) = \sum_{l=1}^{n} C_{j,l}\phi_l \tag{4.4}$$

where N is the number of reduced basis functions we wish to include in our simulation, n is the number of piecewise basis functions that were used in our FEM approximation and u^k_{ROM} is an approximation to $u(x, t^k)$. The definition of our reduced basis solution can then be used to formulate another weak form of the problem, just this time in terms of ψ.

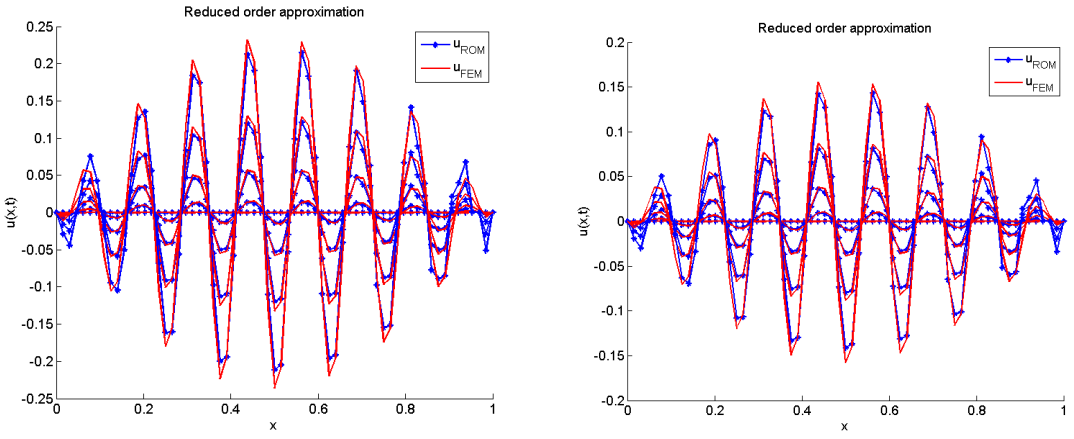
$$\sum_{j=1}^{N} \alpha_j^k \left[a_1 \frac{1}{\Delta t} \int_0^1 \psi_j(x)\psi_i(x)dx + a_2 \int_0^1 \psi_j'(x)\psi_i'(x)dx + a_3 \int_0^1 \psi_j(x)\psi_i(x)dx \right]$$
$$= \int_0^1 f(x,t)\psi_i(x)dx + a_1 \frac{1}{\Delta t} \sum_{j=1}^{N} \alpha_j^{k-1} \int_0^1 \psi_j \psi_i(x)dx \text{ for } j = 1, \ldots N \tag{4.5}$$

So by modifying the loops over our basis functions in our original finite element code we can reconstruct the reduced order solution with a given set of input parameters and reduced basis functions to use.

We now wish to test our reduced order model by constructing a reduced order solution for two different sets of input parameters. The examples we wish to examine are:

$$\begin{aligned} &\text{Example 1: } a_1 = a_2 = a_3 = 1 \\ &\text{Example 2: } a_1 = 1,\ a_2 = 1.5,\ \text{and } a_3 = 0.75 \end{aligned} \tag{4.6}$$

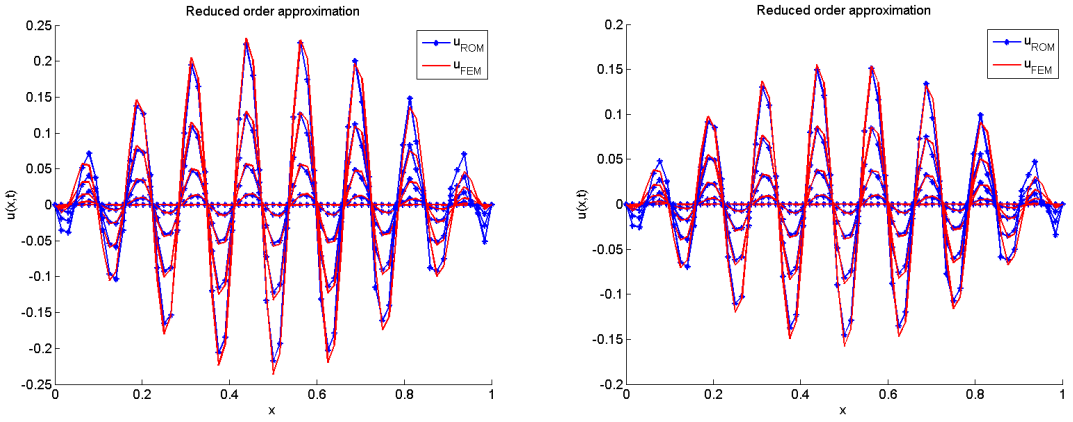
As we can see, the parameters for the first example, existed in our snapshot set whereas the parameters for example two, do not. This is an important concept because the point of reduced order modeling is to be able to approximate solutions that exist within the parameter space quickly. It is alright to generate solutions at parameters used in our snapshot set, but this is not the overall intent of reduced order modeling. Figure 4.5 shows an the reduced order solution to both of our example problems using only a single basis vector.



(a) Example 1 (b) Example 2

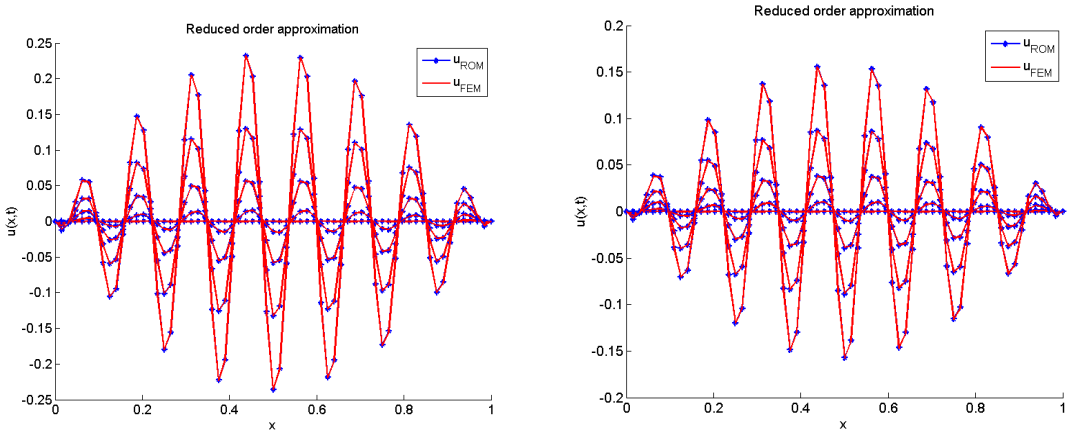
Figure 4.5: Reduced order model approximation to the finite element solution with one basis function for five time instances

From Figure 4.5, we can see that the reduced order solution to both of our example problems appears to do a relatively good job at approximating the solution to differential equation. Figures 4.6 and 4.7 show the reduced order solution using two and three basis functions respectively.



(a) Example 1 (b) Example 2

Figure 4.6: Reduced order model approximation to the finite element solution with two basis functions for five time instances



(a) Example 1 (b) Example 2

Figure 4.7: Reduced order model approximation to the finite element solution with three basis functions for five time instances

As we can see from Figure 4.6 the reduced order model solution lines up fairly closely to the finite

Table 4.1: Example 1: FEM Reduced order approximation error while increasing the number of basis functions

Basis functions	L_2 Error	Euclidean Error
1	6.46873e-001	6.48628e-001
2	2.66148e-001	2.87067e-001
3	1.96136e-001	2.06806e-001
4	1.56399e-002	1.50642e-002
5	2.12547e-002	2.02963e-002

element solution using only two basis functions! And then when we include three basis functions, our approximation gets even better! This is a trend we expect to see because as we add more basis functions, we add more information about the solution. Tables 4.1 and 4.2 show the error between the finite element solution and the reduced order solution for both examples. The reason that the reduced order solution is compared to the finite element solution is that the reduced order model solution was computed based on the finite element solution and not the actual solution. We can refer to the error bound to better understand how the errors we are representing compare to the actual solution:

$$||u - u_{ROM}|| \leq ||u - u_{FEM}|| + ||u_{FEM} - u_{ROM}|| \tag{4.7}$$

So we see that the combination of the error between our reduced order solution and the exact solution is bounded by the summation of the differences of these two errors with the finite element solution. Thus the error that we really care about is the difference between the finite element solution and the reduced order solution because we already have the error between the exact and the FEM solution.

Table 4.2: Example 2: FEM Reduced order approximation error while increasing the number of basis functions

Basis functions	L_2 Error	Euclidean Error
1	6.43105e-001	6.44850e-001
2	2.69186e-001	2.89482e-001
3	1.95596e-001	2.06052e-001
4	1.03010e-002	1.00612e-002
5	1.88020e-002	1.79143e-002

From Table 4.1 we can see that the error clearly is decreasing as we increase the number of basis functions we include in our reduced order model. The fact that the rate at which the error is decreasing is encouraging, showing us that this differential equation is able to be represented accurately with only a few basis functions. This is often the case for linear problems, reduced order modeling is very good at representing linear problems which is not as often the case for nonlinear problems.

CHAPTER 5

THE WAVELET-GALERKIN METHOD

Now that we understand the concepts of wavelets and a Galerkin method, we can begin to formulate one of the objectives of this paper: a wavelet-Galerkin method. Wavelet-Galerkin methods use scaling or wavelet functions as their bases, so the solution of the differential equation is approximated by a linear combination of the elements in the basis. The concepts of these methods have been around since the emergence of wavelet functions yet there were some major hurdles to overcome before they could be counted as viable methods. One of the biggest hurdles was the fact that standard quadrature methods do not work well on the derivatives of the wavelet basis functions. Then in the early 1990's Latto et al. [1] devised a way to exactly compute inner products of the wavelet scaling functions. This opened up new possibilities for the wavelet-Galerkin method to be applied to wide ranges of problems from the harmonic wave equations [13] to Poisson's and Schrödinger's equations [8]. For this work we will be using the scaling functions as a basis due to the considerable amount of work that has been done in this specific area. Nielson[14] presents the process and some examples of using wavelets as opposed to scaling functions as a basis for a Galerkin method.

This chapter will go through the steps we must take to solve our standard one dimensional modified heat equation with Dirichlet boundary conditions defined in equation 3.26 using a wavelet-Galerkin method. The first section will discuss setting up the basis functions to be used in the wavelet-Galerkin method as well as present the weak formulation that will be used throughout this chapter. Following this section will be a discussion of how to calculate the inner products occurring in the weak formulation using the method devised by Latto et al. [1]. Once the inner products have been calculated a method will be presented on how to handle Dirichlet boundary conditions within our linear system. Then we will show how the linear system can be constructed from the precomputed inner products. Finally a few results will be presented to demonstrate the capabilities of the wavelet-Galerkin method.

5.1 Weak Formulation

The first step in setting up our wavelet-Galerkin method is defining our weak formulation based on the one dimensional modified heat equation with Dirichlet boundary conditions that we defined in equation 3.26. The weak formulation has already been given for the finite element case and we will follow a similar process for the wavelet-Galerkin method. In the finite element method, we recall that we multiplied our equation by a test function, v, and then integrated by parts to balance the derivatives. This was done in order to avoid requiring a strict smoothness condition on the unknown solution which allowed us to use a piecewise polynomial basis. For the wavelet-Galerkin method we are going to be using infinitely differentiable scaling functions as our basis, so performing integration by parts is no longer necessary. Using this information we can write our continuous weak formulation:

$$a_1 \int_0^1 u_t v dx - a_2 \int_0^1 u_{xx} v dx + a_3 \int_0^1 uv dx = \int_0^1 f(x,t) v dx \tag{5.1}$$

where $u(x,t)$ is the solution to our differential equation, satisfying the given boundary and initial conditions. Because we wish to generate a fully discrete solution to the differential equation, we must define the solution to our wavelet-Galerkin method as a linear combination of the a scaling coefficient and the Daubechies scaling function. As in the finite element method we will represent the temporal derivative using a backward difference scheme. This means for a fixed time, t^k, we can define the wavelet-Galerkin solution at level m, as:

$$u_{wg}(x,t^k) = 2^{\frac{m}{2}} \sum_j \mu_{j,m}^k \phi(2^m x - j) \tag{5.2}$$

where u_{wg} is the wavelet-Galerkin solution and $\mu_{j,m}^k$ are the scaling coefficient values that we will try to determine. We will determine how many basis functions are needed throughout the domain in the section on boundary conditions as it is not exactly straightforward like the finite element method. Now the objective becomes determining a fully discrete weak formulation so that we can attempt to setup a linear system to solve for $\mu_{j,m}^k$. Remembering that the scaling term, $2^{\frac{m}{2}}$ allows us define our scaling functions as orthonormal, we can write the fully discrete weak formulation as:

$$2^m \sum_j \mu_{j,m}^k \Big[a_1 \frac{1}{\Delta t} \int_0^1 \phi(2^m x - j)\phi(2^m x - i)dx - a_2 \int_0^1 \phi''(2^m x - j)\phi(2^m x - i)dx + a_3 \int_0^1 \phi(2^m x - j)\phi(2^m x - i)dx \Big] = 2^{\frac{m}{2}} \int_0^1 f(x,t)\phi(2^m x - i)dx + a_1 \frac{2^{\frac{m}{2}}}{\Delta t} \int_0^1 u_{wg}^{k-1} \phi(2^m x - i)dx \tag{5.3}$$

As we mentioned, computation of the inner products of scaling function derivatives is not practical with current quadrature methods, due to the extreme oscillatory nature of scaling function derivatives. But if we apply a method defined by Latto et. al.[1], we can find the inner products of the form:

$$\int \phi''(x)\phi(x - l)dx \tag{5.4}$$

In order to get the terms in our weak formulation to conform to this definition of the inner product, we must perform a change of variables. So if we set $y = 2^m x - j$ for the double derivative integral terms in equation 5.3, then we can write the weak formulation inner product term as:

$$\int \phi''(2^m x - j)\phi(2^m x - i)dx = 2^{-m} \int \phi''(y)\phi(y + j - i)dy \tag{5.5}$$

where we can define a shift term, l, based on the arbitrary shift terms, i and j, such that $l = i - j$. Using this definition, it is clear that we have the inner product definition from equation 5.4.

In section 5.2 we will discuss how to exactly calculate the inner product defined in equation 5.4 using the method developed by Latto et. al.[1]. Section 5.3 will present a technique to deal with Dirichlet boundary conditions via modifying the number of scaling functions over the entire domain.

5.2 Calculating the Connection Coefficients

Due to the inherent oscillatory properties of the derivatives of scaling functions, we cannot use a standard quadrature rule [16] to approximate the values of the inner products like was done in the finite element method. To circumvent this problem we will use a method developed by Latto et al. [1] to generate, using Latto's terminology, connection coefficients which are used to determine the exact inner products between scaling functions and their derivatives.

First we will present Figure 5.1, which shows an example collection of scaling basis functions that can be used to construct our wavelet-Galerkin method. In the finite element case, it was clear which basis functions we needed to include because their support was only over two sub-intervals of the domain whereas in the wavelet case we must now account for the entire domain covered by the specific wavelet we choose. Based on the domain of support for each of our scaling functions we find that the number of overlapping basis functions is dependent on the domain of the scaling function and subsequently, the scaling function genus. This also means that our scaling basis functions do not form a nodal basis, but it turns out that we do not need to reconstruct the solution with respect to the explicit function values of our scaling basis functions; this will be explained in a later section.

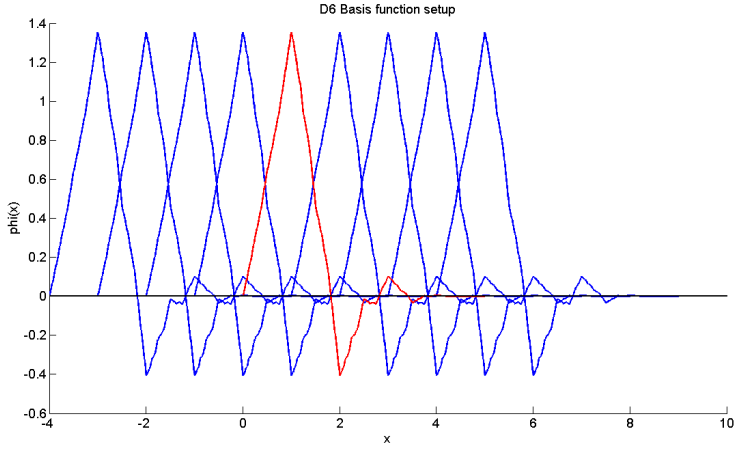


Figure 5.1: $D6$ Scaling basis functions with $m = 0$

In order to determine which connection coefficients we must compute, we must first understand the physical properties of our basis functions. For our Daubechies family of wavelets, DN, we know that each $\phi(x)$ has a domain of support equal to $[0, N-1]$. So for the example in Figure 5.1, our scaling function is $D6$ and has a domain of support $[0, 5]$, the red function indicates the scaling function with no shift term. From the plot we can see that the scaling functions $\phi(x+4)$ (support $[-4, 1]$) to $\phi(x-4)$ (support $[4, 9]$) have some overlap on the domain $[0, 5]$, of $\phi(x)$. Thus in general we can state that the scaling functions $\phi(x+(N-2))$ to $\phi(x-(N-2))$ overlap the original domain $[0, N-1]$. This means that the overall domain that we must account for is $[-(N-2), (N-2)]$ implying there are $2(N-2)+1$ basis functions that contribute to the inner product with $\phi(x)$. We can see this in the example $D6$, where we have nine total basis functions and thus nine integrals

to compute. Likewise for $D8$ and $D10$ there should be thirteen and seventeen integrals to compute respectively. This is important because it defines the number of connection coefficients we will have for each type of scaling function, as well as the resulting bandwidth of our linear system to be constructed.

In order to solve for the connection coefficients, $\Gamma^d_{m,l}$, we must use the definition of the inner product to write:

$$\Gamma^d_{m,l} = \int_{-\infty}^{\infty} \phi^d(x)\phi(x-l)dx \tag{5.6}$$

where for our case, $d = 2$, but we will present the general case because the amount of work is the same. We will start by first substituting the definition of the scaling function from equation 2.14 into the connection coefficient relation:

$$\Gamma^d_{m,l} = 2(2^d)\int_{-\infty}^{\infty} \left(\sum_{k=0}^{N-1} h_k \phi^d(2x-k)\right)\left(\sum_{k-l=0}^{N-1} h_{k-l}\phi(2^m x - k - l)\right)dx \tag{5.7}$$

where we can define the variable $j = k - l$ and also refine the leading coefficient:

$$\Gamma^d_{m,l} = 2^{d+1}\int_{-\infty}^{\infty} \left(\sum_{k=0}^{N-1} h_k \phi^d(2x-k)\right)\left(\sum_{j=0}^{N-1} h_j \phi(2x-j)\right)dx \tag{5.8}$$

And then to put the equation in the form of the connection coefficient relation we can perform a change of variables with $y = 2x - k$ in order to put the shift term on only one scaling function.

$$\Gamma^d_{m,l} = 2^{d+1}\sum_{k=0}^{N-1}\sum_{j=0}^{N-1} h_k h_j \int_{-\infty}^{\infty} \phi^d(y)\phi(y-j+k)dy \tag{5.9}$$

Because j and k are arbitrary shift terms we can relate $j + k$ back to our original inner product shift term, l, we are left with:

$$\Gamma^d_{m,l} = 2^{d+1}\sum_{k=0}^{N-1}\sum_{l=0}^{N-1} h_k h_l \int_{-\infty}^{\infty} \phi^d(y)\phi(y-l)dy \tag{5.10}$$

Then since we have defined $\Gamma^d_{m,l}$ as the inner product of two scaling functions with a derivative term, we can represent the right hand side of equation 5.10 in terms of $\Gamma^d_{m,l}$. Thus we are going to end up with an eigenvalue problem:

$$\frac{1}{2^{d+1}}\Gamma^d_{m,l} = \sum_{k=0}^{N-1}\sum_{l=0}^{N-1} h_k h_l \Gamma^d_{m,l} \tag{5.11}$$

Now because we wish to solve this system of equations for all the possible connection coefficients, the strategy here is to turn this into a linear system of equations. Were we to replace the sum on the right hand side ($\sum_{k=0}^{N-1}\sum_{l=0}^{N-1} h_k h_l$) with a matrix T, then our equation would become:

$$\frac{1}{2^{d+1}}\Gamma^d_{m,l} = T\Gamma^d_{m,l} \tag{5.12}$$

Then we can turn this eignevalue problem into a linear system, which will need an additional constraint in order to get a unique solution:

$$(T - \frac{1}{2^{d+1}}I)\Gamma^d_{m,l} = 0 \tag{5.13}$$

So the next step involves setting up the matrix, T, to represent the coefficient summation on the right hand side of equation 5.11. To do this, we can define the two summation coefficients, h_k and h_l, as Toeplitz matrices. The h_k term in equation 5.11 will yield a coefficient matrix of $(\downarrow 2)H$ where $(\downarrow 2)$ lets us know that we must shift each row down two spaces. Then from the h_l term in 5.7 it can be shown that the Daubechies coefficients will yield an H^T Toeplitz matrix. Thus the coefficient matrix, T, we wish to generate can be defined by by the matrix multiplication of the two Toeplitz matrices such that $T = (\downarrow 2)HH^T$. Using this method we end up with a matrix that is size $2N-3 \times 2N-3$, and we can visualize the first three rows of matrix T for $D4$.

$$T = \begin{bmatrix} \sum_{k=0}^{2} h_k h_{k+1} & \sum_{k=0}^{3} h_k^2 & \sum_{k=0}^{2} h_k h_{k+1} & \sum_{k=0}^{1} h_k h_{k+2} & h_0 h_3 & \cdots \\ h_0 h_3 & \sum_{k=0}^{1} h_k h_{k+2} & \sum_{k=0}^{2} h_k h_{k+1} & \sum_{k=0}^{3} h_k^2 & \sum_{k=0}^{2} h_k h_{k+1} & \cdots \\ 0 & 0 & h_0 h_3 & \sum_{k=0}^{1} h_k h_{k+2} & \sum_{k=0}^{2} h_k h_{k+1} & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{5.14}$$

The problem with this linear system is, according to Besora[3], there is no unique nonzero solution to this linear system. It has been shown that using the moments of the scaling functions we can derive an additional normalization condition, leaving us with a unique solution to the connection coefficient problem. Thus, we must find a way to relate the moments to the connection coefficients we are trying to determine. Our strategy for approaching this problem will be to find a general relation between the moment equation and our scaling functions and eventually our

connection coefficients. To do this we will attempt to determine M_0^p using M_0^j for $j < p$ given M_0^0. First we must remind ourselves of the traditional moment equation:

$$M_l^p = \int_{-\infty}^{\infty} x^p \phi(x - l) dx \tag{5.15}$$

where l again represents the function shifts and p is the polynomial order we will multiply our scaling function by. From the definition of our scaling function, we noted that: $\int_{-\infty}^{\infty} \phi(x) dx = 1$. So this means that we can say that the zero-order moment, $M_l^0 = 1$ for all l in the domain of our scaling function.

Then we will use the definition of the dilation equation 2.14 to substitute into our original moment equation 5.15 letting $l = 0$.

$$M_0^p = \sqrt{2} \sum_{k=0}^{N-1} h_k \int_{-\infty}^{\infty} x^p \phi(2x - k) dx \tag{5.16}$$

where we will again use m as the discretization of our mesh and k as a shift parameter. Then if we want to perform a change of variables by substituting $y = 2x$ our moment equation would become:

$$M_0^p = \frac{2^{\frac{1}{2}}}{(2)2^p} \sum_{k=0}^{N-1} h_k \int_{-\infty}^{\infty} y^p \phi(y - k) dy \tag{5.17}$$

and cleaning it up a little:

$$M_0^p = \frac{1}{2^{p+\frac{1}{2}}} \sum_{k=0}^{N-1} h_k \int_{-\infty}^{\infty} y^p \phi(y - k) dy \tag{5.18}$$

Similar to the procedure that we used to calculate the connection coefficients, we notice that there appears to be a recurrence relation in equation 5.18. Thus we can write said recurrence relation as:

$$M_0^p = \frac{1}{2^{p+\frac{1}{2}}} \sum_{k=0}^{N-1} h_k M_k^p \tag{5.19}$$

Now we need to find a way to solve for M_k^p. To do this we can perform another change of variables on the definition of M_k^p given at the end of 5.18. We will set $u = y - k$ so that our scaling function is only a function of a single variable. Doing this then expands the y^p term which will now become $(u + k)^p$, which can be solved using a binomial expansion according to equation 5.20

$$(u + k)^p = \sum_{j=0}^{p} \binom{p}{j} u^j k^{p-j} \tag{5.20}$$

Thus we can use the definition of the binomial expansion 5.20 and the moment equation for M_k^p to define a relation between the moment at any shift to the moment with no shift.

$$\begin{aligned} M_k^p &= \int_{-\infty}^{\infty} (u+k)^p \phi(u) dx \\ M_k^p &= \sum_{j=0}^{p} \binom{p}{j} k^{p-j} \int_{\infty}^{\infty} u^j \phi(u) dx \\ M_k^p &= \sum_{j=0}^{p} \binom{p}{j} k^{p-j} M_0^j \end{aligned} \tag{5.21}$$

and then by putting this equation back into equation 5.19 we can generate a general equation for calculating the moment at any polynomial order, p, with no shifts, $k = 0$.

$$M_0^p = \frac{1}{2^{p+\frac{1}{2}}} \sum_{k=0}^{N-1} h_k \sum_{j=0}^{p} \binom{p}{j} k^{p-j} M_0^j \tag{5.22}$$

If we recall the fact that the moment equation should integrate to one when $p = 0$ we can calculate the moments at any shift and polynomial order. Now that we have found a way to calculate the moments we need to use them to normalize our connection coefficient linear system. To do this we need a way to relate the moments of our scaling function to the connection coefficients we are trying to find. Along with the definition of a moment in equation 5.15 we can define an infinite summation using the same relation except we are now just isolating the polynomial term:

$$x^d = \sum_{l=-\infty}^{\infty} M^d \phi^d(x-l) \tag{5.23}$$

If we differentiate this equation and multiply both sides by the integration of the scaling function $\int_{-\infty}^{\infty} \phi(x)$, which integrates to one, we will get a relation to the connection coefficients:

$$\sum_{l=2-N}^{N-2} M_l^d \Gamma_l^d = d! \tag{5.24}$$

But now we must adjust our scaling factor outside of equation 5.22 because we need to account for the derivative term. For our linear system we only need to relate the derivative term to the moment equation so the polynomial order is no longer necessary. So if we go back to equation 5.16 and instead set $p = 0$ and put a derivative term on our scaling function, it can be shown that the moment equation will become:

$$M^d = \frac{1}{2^{d+\frac{1}{2}}} \sum_{k=0}^{N-1} h_k \sum_{j=0}^{d} \binom{d}{j} k^{d-j} M_0^j \tag{5.25}$$

Now we have everything we need to solve our linear system in equation 5.13 as long as we add on an additional row to represent the normalizing moment equation that we have just derived. Using this method we can solve and store the connection coefficients as a pre-processing step so that they can just be indexed into our linear system that we will use to generate a wavelet-Galerkin solution to our differential equation. Tables 5.1 to 5.3 display a few of the computed connection coefficients at $D6$, $D8$ and $D10$.

Table 5.1: Connection Coefficients for $D6$

	$d=1$		$d=2$		
Γ	$m=1$	$m=3$	$m=1$	$m=2$	$m=3$
$k=-4$	-6.849315e-004	-1.095890e-002	2.142857e-002	3.428571e-001	5.485714e+000
$k=-3$	-2.922374e-002	-4.675799e-001	4.571429e-001	7.314286e+000	1.170286e+002
$k=-2$	2.904110e-001	4.646575e+000	-3.504762e+000	-5.607619e+001	-8.972190e+002
$k=-1$	-1.490411e+000	-2.384658e+001	1.356190e+001	2.169905e+002	3.471848e+003
$k=0$	-4.508571e-015	2.424328e-015	-2.107143e+001	-3.371429e+002	-5.394286e+003
$k=1$	1.490411e+000	2.384658e+001	1.356190e+001	2.169905e+002	3.471848e+003
$k=2$	-2.904110e-001	-4.646575e+000	-3.504762e+000	-5.607619e+001	-8.972190e+002
$k=3$	2.922374e-002	4.675799e-001	4.571429e-001	7.314286e+000	1.170286e+002
$k=4$	6.849315e-004	1.095890e-002	2.142857e-002	3.428571e-001	5.485714e+000

5.3 Handling Dirichlet Boundary Conditions

Now that we have the values of the improper connection coefficients, we want to see how to use these to set up a linear system to determine the wavelet-Galerkin approximation to our differential equation. To do this we must determine how to implement the Dirichlet boundary conditions. In the literature there are two main approaches to resolve the boundary conditions. The first is to modify the basis functions at the boundary so that their support is reduced[16]. The second approach is to extend the domain so that the support of all basis functions is included in the domain. Thus the improper connection coefficients that we computed can be used directly. For this research the method of extending the domain will be used to resolve the boundaries, this method is often referred to as the "Fictitious Boundary" approach [12].

Table 5.2: Connection Coefficients for $D8$

	$d=1$		$d=2$		
Γ	$m=1$	$m=3$	$m=1$	$m=2$	$m=3$
$k=-6$	-1.681701e-006	-2.690722e-005	6.368660e-005	1.018986e-003	1.630377e-002
$k=-5$	3.444124e-004	5.510598e-003	-6.521508e-003	-1.043441e-001	-1.669506e+000
$k=-4$	4.448099e-003	7.116959e-002	-4.229091e-002	-6.766546e-001	-1.082647e+001
$k=-3$	-6.716041e-002	-1.074567e+000	6.038916e-001	9.662266e+000	1.545962e+002
$k=-2$	3.839979e-001	6.143967e+000	-2.791476e+000	-4.466362e+001	-7.146180e+002
$k=-1$	-1.586019e+000	-2.537630e+001	1.056828e+001	1.690925e+002	2.705480e+003
$k=0$	2.874322e-014	-8.929573e-015	-1.666389e+001	-2.666223e+002	-4.265957e+003
$k=1$	1.586019e+000	2.537630e+001	1.056828e+001	1.690925e+002	2.705480e+003
$k=2$	-3.839979e-001	-6.143967e+000	-2.791476e+000	-4.466362e+001	-7.146180e+002
$k=3$	6.716041e-002	1.074567e+000	6.038916e-001	9.662266e+000	1.545962e+002
$k=4$	-4.448099e-003	-7.116959e-002	-4.229091e-002	-6.766546e-001	-1.082647e+001
$k=5$	-3.444124e-004	-5.510598e-003	-6.521508e-003	-1.043441e-001	-1.669506e+000
$k=6$	1.681701e-006	2.690722e-005	6.368660e-005	1.018986e-003	1.630377e-002

The fictitious boundary method requires us to add $N-2$ scaling functions to each end of the domain, in one dimension, so that the scaling functions at the end of each boundary are balanced with the $N-2$ scaling functions on either side of it. The $N-2$ scaling functions are added to both ends of the domain to preserve symmetry even though they are only needed on the right hand side of the domain to ensure that our final basis function has support contained in the region. The first and last equations in our linear system will be reserved for enforcing the Dirichlet boundary conditions. So the size of our system depends not only on the resolution of the domain but also the choice of scaling function. For example if we consider the node spacing, $\frac{1}{2^m}$, that will account for 2^m+1 equations, assuming a domain of $[0,1]$. Then if we include the fictitious points, $2(N-2)$, plus the two extra equations to enforce our Dirichlet boundary, we are left with a matrix size of $2^m+3+2(N-2)$. So we can see for high resolutions or large values of m, then $3+2(N-2)$ is less burdensome on the linear system.

We will reserve the first and last equations in the linear system to handle the Dirichlet boundary conditions. The equation that handles the Dirichlet boundary condition can be understood by examining the example at the left boundary where we set $u_{wg}(0,t)=g(t)$, so that:

Table 5.3: Connection Coefficients for $D10$

	$d=1$		$d=2$		
Γ	$m=1$	$m=3$	$m=1$	$m=2$	$m=3$
$k=-8$	-5.392104e-010	-8.627354e-009	1.415504e-008	2.264806e-007	3.623690e-006
$k=-7$	-5.048234e-007	-8.077175e-006	6.626177e-006	1.060188e-004	1.696301e-003
$k=-6$	-1.080946e-004	-1.729514e-003	1.468582e-003	2.349730e-002	3.759569e-001
$k=-5$	-4.784716e-004	-7.655546e-003	3.178482e-003	5.085572e-002	8.136915e-001
$k=-4$	1.492279e-002	2.387647e-001	-1.196319e-001	-1.914111e+000	-3.062577e+001
$k=-3$	-1.067051e-001	-1.707282e+000	7.238142e-001	1.158103e+001	1.852964e+002
$k=-2$	4.576404e-001	7.322246e+000	-2.598009e+000	-4.156814e+001	-6.650902e+002
$k=-1$	-1.651812e+000	-2.642899e+001	9.659161e+000	1.545466e+002	2.472745e+003
$k=0$	1.963903e-014	-1.481931e-013	-1.533998e+001	-2.454396e+002	-3.927034e+003
$k=1$	1.651812e+000	2.642899e+001	9.659161e+000	1.545466e+002	2.472745e+003
$k=2$	-4.576404e-001	-7.322246e+000	-2.598009e+000	-4.156814e+001	-6.650902e+002
$k=3$	1.067051e-001	1.707282e+000	7.238142e-001	1.158103e+001	1.852964e+002
$k=4$	-1.492279e-002	-2.387647e-001	-1.196319e-001	-1.914111e+000	-3.062577e+001
$k=5$	4.784716e-004	7.655546e-003	3.178482e-003	5.085572e-002	8.136915e-001
$k=6$	1.080946e-004	1.729514e-003	1.468582e-003	2.349730e-002	3.759569e-001
$k=7$	5.048234e-007	8.077175e-006	6.626177e-006	1.060188e-004	1.696301e-003
$k=8$	5.392093e-010	8.627358e-009	1.415504e-008	2.264807e-007	3.623690e-006

$$u_{wg}(0,t^k) = \sum_j \mu^k_{j,m}\phi(-j) = g(t^k) \tag{5.26}$$

where if we take the inner product with $\phi(-l)$ we find:

$$\sum_j \mu^k_{j,m} \int \phi(-j)\phi(-l)dx = g(t^k)\int \phi(-l)dx = g(t^k) \tag{5.27}$$

which by orthonormality of the scaling functions, we can see that the integral, on the left, just evaluates to the Kronecker-delta function, $\delta_{j,l}$. So this equation for the Dirichlet boundary conditions on the left side of the domain just becomes the identity with the one in the Nth column and the first element of the right hand side should be $g(t^k)$. The right boundary can be handled in a similar manner.

5.4 Solving for the Wavelet-Galerkin Approximation

Now that we have discussed the weak formulation, connection coefficients and the Dirichlet boundary conditions; we are now ready to begin setting up the linear system for the discrete

problem we discussed in the weak formulation section. So first we will write the simplified discrete weak formulation at time t^k:

$$\begin{aligned} 2^m \sum_j \mu_{j,m}^k \left[a_1 \frac{1}{\Delta t} \int \phi_{m,j}(x)\phi_{m,i}(x)dx - a_2 \int \phi''_{m,j}(x)\phi_{m,i}(x)dx + a_3 \int \phi_{m,j}(x)\phi_{m,i}(x)dx \right] \\ = 2^{\frac{m}{2}} \int f(x,t)\phi_{m,i}(x)dx + a_1 \frac{2^{\frac{m}{2}}}{\Delta t} \int u_{wg}^{k-1}\phi_{m,i}(x)dx \end{aligned} \tag{5.28}$$

where we recall that $\phi_{m,j}$ and $\phi_{m,i}$ are shorthand notation defined in equation 2.4. Then when we apply orthonormality, we can rewrite the discrete weak form:

$$\begin{aligned} 2^m \sum_j \mu_{j,m}^k \left[a_1 \frac{1}{\Delta t}\delta_{j,i} - a_2 \int \phi''_{m,j}(x)\phi_{m,i}(x)dx + a_3\delta_{j,i} \right] \\ = 2^{\frac{m}{2}} \int f(x,t)\phi_{m,i}(x)dx + a_1 \frac{2^{\frac{m}{2}}}{\Delta t} \int u_{wg}^{k-1}\phi_{m,i}(x)dx \end{aligned} \tag{5.29}$$

So using the discrete weak formulation defined above we can begin to formulate the linear system. Like the finite element method we can define a set of matrices that represent the left hand side of the linear system. So we can write the elements of the non-derivative matrix, D_1, as:

$$D_1 = a_1 \frac{1}{\Delta t}\delta_{j,i} + a_3\delta_{j,i} \tag{5.30}$$

and then the elements of the double derivative matrix, D_2:

$$D_2 = a_2 \int \phi''_{m,j}(x)\phi_{m,i}(x)dx \tag{5.31}$$

Then using the definitions of these two matrices, we can write the linear system of equations:

$$D_1\vec{\mu} + D_2\vec{\mu} = F \tag{5.32}$$

where D_1 and D_2 will represent the non derivative and the double derivative terms respectively. Also F is just used as a placeholder for the right hand side in this example and will be clarified later on in this section. Since D_1 represents the non-derivative terms and we know that these terms can be represented by the Kronecker-delta function, its construction is straightforward. We can construct D_1 as the identity matrix for all rows and columns excluding the first and last rows which are reserved for the Dirichlet boundary condition enforcement. The first six rows of D_1 can be visualized in equation 5.33 using $D6$ scaling function as the basis.

$$D_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & \cdots \\ 0 & \frac{a_1}{\Delta t} + a_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & \frac{a_1}{\Delta t} + a_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & \frac{a_1}{\Delta t} + a_3 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & \frac{a_1}{\Delta t} + a_3 & 0 & 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & 0 & \frac{a_1}{\Delta t} + a_3 & 0 & 0 & 0 & 0 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \end{bmatrix} \tag{5.33}$$

We can see that this matrix takes a similar form as the identity matrix, except for the first and last equations which are used to satisfy the Dirichlet boundary conditions. From the first six equations it is clear how this matrix can be represented throughout its domain and the right boundary.

For D_2, we must now take into account the connection coefficients due to the double derivative term. For this matrix we must associate each scaling function that we are trying to compute, with its $N-2$ neighbors on its right and left sides. So if we fix the resolution, m and the derivative $d = 2$ we can make the generalization $\Gamma^2_{m,k} = \Gamma_k$. Using this definition of the shifts in the connection coefficients, we can write the D_2 matrix:

$$D_2 = a_2 \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \cdots \\ \Gamma_{-1} & \Gamma_0 & \Gamma_1 & \Gamma_2 & \Gamma_3 & \Gamma_4 & 0 & 0 & 0 & 0 & \cdots \\ \Gamma_{-2} & \Gamma_{-1} & \Gamma_0 & \Gamma_1 & \Gamma_2 & \Gamma_3 & \Gamma_4 & 0 & 0 & 0 & \cdots \\ \Gamma_{-3} & \Gamma_{-2} & \Gamma_{-1} & \Gamma_0 & \Gamma_1 & \Gamma_2 & \Gamma_3 & \Gamma_4 & 0 & 0 & \cdots \\ \Gamma_{-4} & \Gamma_{-3} & \Gamma_{-2} & \Gamma_{-1} & \Gamma_0 & \Gamma_1 & \Gamma_2 & \Gamma_3 & \Gamma_4 & 0 & \cdots \\ 0 & \Gamma_{-4} & \Gamma_{-3} & \Gamma_{-2} & \Gamma_{-1} & \Gamma_0 & \Gamma_1 & \Gamma_2 & \Gamma_3 & \Gamma_4 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \end{bmatrix} \tag{5.34}$$

Using this setup, the interior part of the matrix (excluding the Dirichlet boundary enforcement) should be a banded linear system with bandwidth $2(N-2)+1$. Extending the pattern of this matrix for the first six nodes on the left boundary, one can see how the interior portion of the matrix will be constructed along with the equations on the right hand side of the domain.

Now that the matrix portion of the linear system has been expressed, we must determine a way to represent the right hand side of the weak formulation in equation 5.29. To do this we will cite two theorem's presented by Burrus et. al. [5] that make use of the projection properties of the scaling function space. Before we state the two theorems we must define the projection and sampling of our function signal, $f(x,t)$. We can define the orthogonal projection of $f(x,t)$ onto our subspace V_m, which we defined in wavelet chapter, at a fixed time t^k as:

$$P^m\{f(x,t^k)\} = \sum_j \left(\int f(x,t^k)\phi_{m,j}(x)dx \right) \phi_{m,j}(x) \tag{5.35}$$

And then we can also define the sampling of our function, $f(x,t)$ at a fixed time, t^k, as:

$$S^m\{f(x,t^k)\} = \sum_j 2^{\frac{-m}{2}} f(\frac{j}{2^m})\phi_{m,j}(x) \tag{5.36}$$

Using these definitions we can then present the two theorems. First theorem 5.4.1

Theorem 5.4.1 *If $m_1(l) = 0$ for $l = 0, 1, \cdots, L$ then the L^2 error is:*

$$\epsilon_1 = \|f(x,t^k) - P^m\{f(x,t^k)\}\|_2 \leq C2^{-m(L+1)} \tag{5.37}$$

where C is a constant independent of m and L but dependent on $f(x,t^k)$ and the wavelet system.

We note that $L+1$ is the number of vanishing moments for a given Daubechies scaling function, DN; this means that $N = L+1$. Then if we incorporate the definition of the sampling of our function, $S^m\{f(x,t^k)\}$ we have theorem 5.4.2:

Theorem 5.4.2 *If $m(l) = 0$ for $l = 0, 1, \cdots, L$ then the L^2 error is:*

$$\epsilon_1 = \|S^m\{f(x,t^k)\} - P^m\{f(x,t^k)\}\|_2 \leq C_2 2^{-m(L+1)} \tag{5.38}$$

where C_2 is a constant independent of m and L but dependent on $f(x,t^k)$ and the wavelet system.

From theorem 5.4.2 we can see that the error in the sampling and the projection are bound by $C_2 2^{-m(L+1)}$. Using this definition we can combine these theorems to see that the difference between the function itself and its sampling are bounded by the same term, $2^{-m(L+1)}$. Thus the elements in our unknown sampling vector $S^m\{f(x,t^k)\}$ can be approximated by the function values themselves at the defined nodes. This means that not only can we represent the right hand side of our linear system as simply the function values at the nodes, but we can also represent our solution directly in terms of our unknown vector, $\mu^k_{m,j}$.

5.5 Numerical Results

Now that we have formulated how one would solve a differential equation using this wavelet-Galerkin method we should go back to our test problem to see how well our one dimensional modified heat equation with Dirichlet boundary conditions fares under a wavelet-Galerkin scheme. Using this information we can use the wavelet-Galerkin method to solve the differential equation, present some plots and finally generate some convergence rates which can be compared to the exact solution. In this section we will present two important points devoted to explaining how the wavelet-Galerkin method is executed and the numerical results it produces. First, the structure of the wavelet-Galerkin code will be explained and comparisons will be made with the finite element method. Then we will present two numerical examples of the wavelet-Galerkin method and its solution to the one dimensional modified heat equation.

Before we present any numerical results, it will be useful to compare the wavelet-Galerkin method to the finite element method in order to understand the differences between the methods. The first thing to note is the assembly of the linear system in both the wavelet-Galerkin and finite element methods. We presented a standard algorithm that is often used in the construction of the finite element matrices in the finite element chapter. There is no de-facto algorithm for setting up the wavelet-Galerkin method, so one of the focuses of this research was to construct a unique algorithm for setting up the linear system given a choice of the scaling function, DN. We will recall that for the modified heat equation we only need to calculate connection coefficients for the case where $d = 2$ under a given resolution, m, and using a given scaling function choice, DN. With this information we can pre-compute and store the connection coefficients so that it just becomes an indexing exercise to put them into our linear system. This is a much different approach than the standard finite element method, presented earlier, where we were required to compute the integrals using a quadrature method, as we were constructing our linear system.

Additionally we should note some properties regarding the linear systems produced in the finite element and the wavelet-Galerkin methods. We recall that the finite element method with piecewise linear basis functions has a bandwidth of 3 whereas the wavelet-Galerkin method has a bandwidth of $2(N-2)+3$ if we account for the equations that handle the Dirichlet boundary conditions. It is important to note that if we wish to increase the scaling function genus to the next order, our

bandwidth will increase by a size of four. But, because the connection coefficients are pre-computed and each individual element in our matrix is independent, this method is easily parallelizable.

Now that we understand the differences between the algorithmic side of the wavelet-Galerkin and finite element methods we should present some numerical results for the wavelet-Galerkin method to see how this method performs when solving the one dimensional modified heat equation. Throughout the remainder of this chapter we will focus on two example modified heat equation problems, one with a homogeneous right hand side and the same equation that we solved for in the finite element example 3.26. The two example modified heat equations are expressed below:

$$\begin{aligned}
&\text{Example 1: } u_t - u_{xx} = 0 \\
&\qquad \text{with boundary and initial conditions: } u(0,t) = 2t,\ u(1,t) = 1 + 2t \text{ and } u(x,0) = x^2 \\
&\text{Example 2: } u_t - u_{xx} + u = (1 + 4t\pi^2 + t)\sin(2\pi x) \\
&\qquad \text{with boundary and initial conditions: } u(0,t) = u(1,t) = u(x,0) = 0
\end{aligned} \tag{5.39}$$

We will note that the exact solution for the two examples can be shown to be: $u(x,t) = x^2 + 2t$ and $u(x,t) = t\sin(2\pi x)$ respectively.

First, we should look at a single scaling function and the ability for this scaling function to approximate our one dimensional modified heat equation. So using $D12$ to start off we can generate the plots shown in Figures 5.2 and 5.3 that illustrate the ability of a wavelet-Galerkin method to approximate the solution to the modified heat equation. These simulations were run at varying resolutions over the time domain $t = [0, 0.1]$ to demonstrate that as the resolution is increased, the wavelet-Galerkin approximation approaches the exact solution. In solving for each of these results we will consistently keep the temporal discretization set at $\Delta t = \Delta x^2$.

From Figures 5.2 and 5.3 we can see that as we increase the resolution, the wavelet-Galerkin solution appears to better approximate the exact solution for the one dimensional modified heat equation.

Next we would like to see how the wavelet-Galerkin method performs when higher order scaling functions are used. So for both examples we attempt to determine how the scaling function order affects the precision of our wavelet-Galerkin method. Figures 5.4 and 5.5 show the wavelet-Galerkin solution using the eight different scaling functions examined in this research compared to the exact

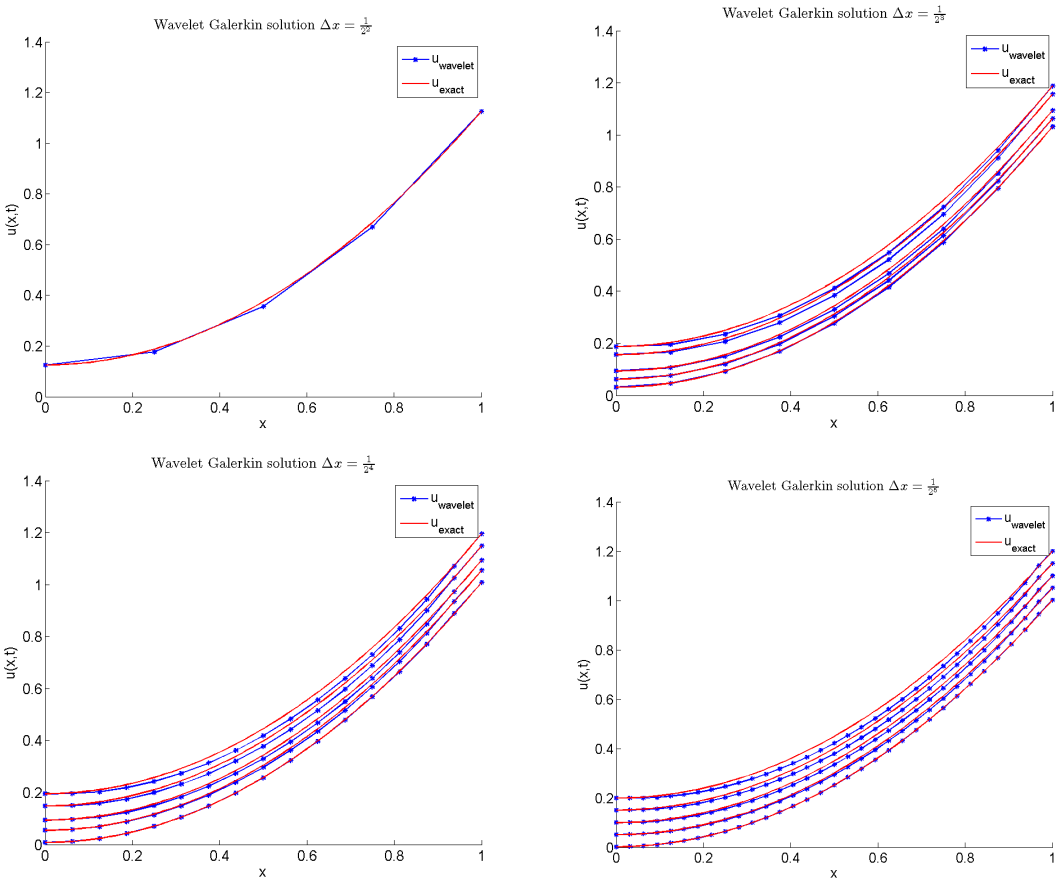


Figure 5.2: Example 1 Wavelet-Galerkin solution compared to actual solution at varying resolutions at 5 time instances

solution at time $t = 0.1$ and level $m = 3$. The plots are split up into the first four scaling functions $D6 - D12$ and $D14 - D20$ with a zoomed in view where there are some visible differences.

From Figures 5.4 and 5.5 we can see that as we increase the order of the scaling function the wavelet-Galerkin approximation appears to trend closer to the actual solution. This property is most easily observed in the jump in accuracy between scaling functions $D6$ and $D8$. We can see in Tables 5.4 and 5.5, that the error decreases as the scaling function order is increased. In the finite element section, the Euclidean error rate was computed alongside the, more appropriate, L^2 error so that we could compare the rate to the wavelet-Galerkin method. We will use this metric to compute the error of the wavelet-Galerkin method because unlike the finite element method we do not have an explicit method of determining the function values within the elements, which is

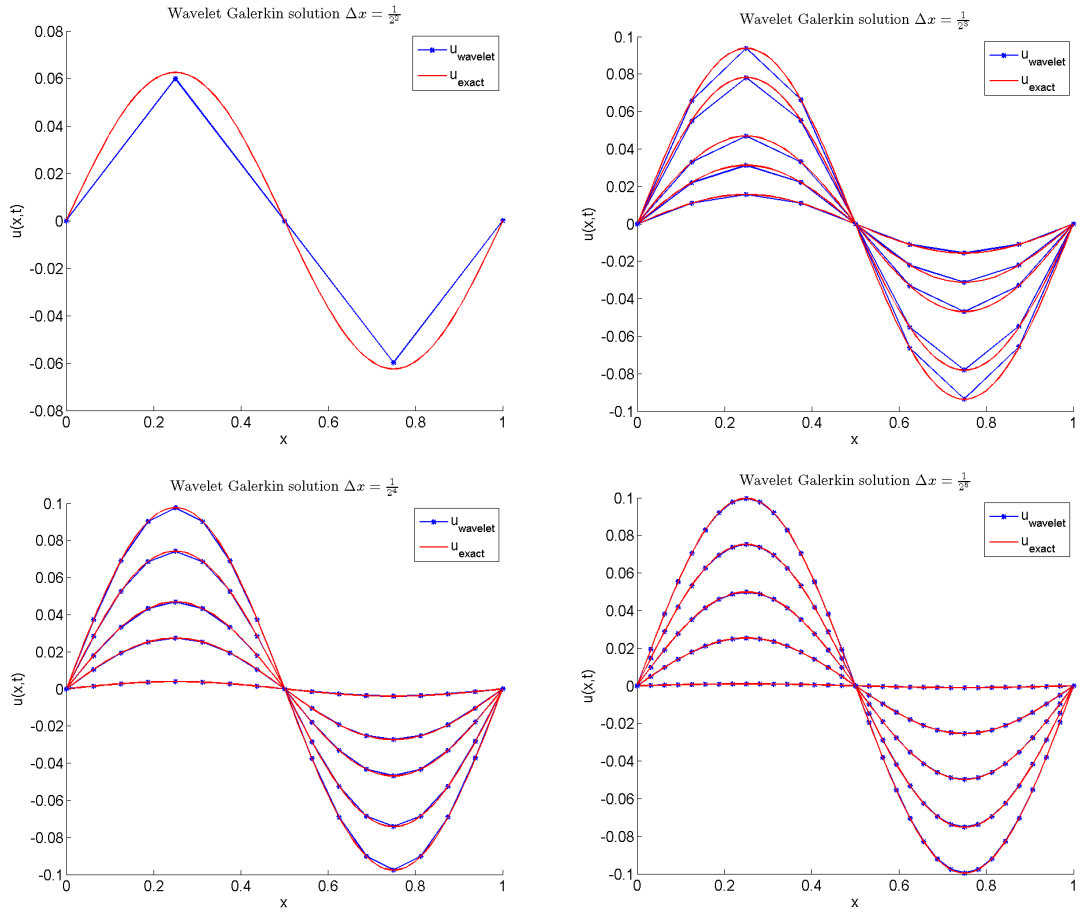


Figure 5.3: Example 1 Wavelet-Galerkin solution compared to actual solution at varying resolutions at 5 time instances

required for a quadrature rule. We also make note of the condition numbers in Tables 5.4 and 5.5, which we believe may be one of the contributing sources of error in the wavelet-Galerkin method.

Now that we feel confident that this wavelet-Galerkin method appears to be approximating the solution correctly we can examine some convergence rates and compare them to the convergence rates from Table 3.2. Table 5.6 presents the Euclidean error and convergence rates found when comparing the exact solution to our wavelet-Galerkin method using $D10$ as the basis function. Additionally, Table 5.6 contains a column for the condition number of the matrix.

From Table 5.6 we notice a few things. First off, the condition number appears to be rather large in relation to a comparable linear system used to solve a differential equation. Although, it is difficult to classify what constitutes a "high" condition number, as it is often problem specific.

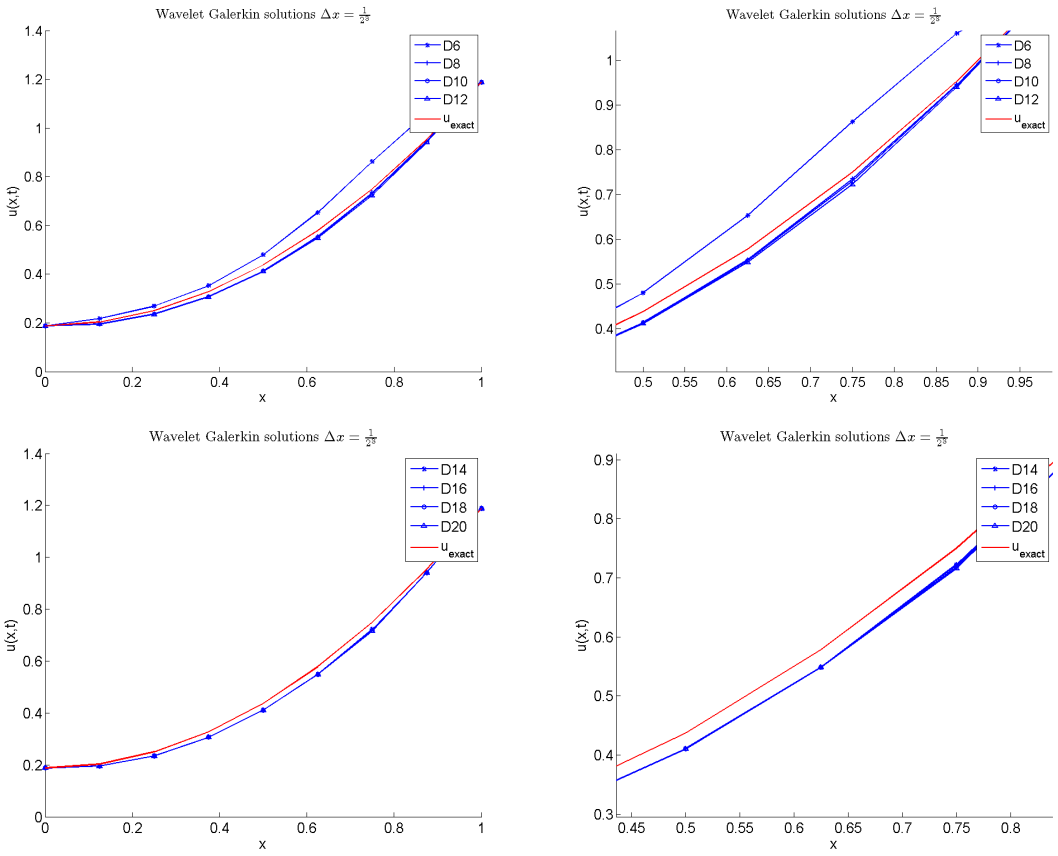


Figure 5.4: Example 1 Wavelet-Galerkin solution compared to actual solution at different ordered scaling functions (plots on the right are zoomed in)

We also noticed this trend when we varied the scaling function order, implying that this condition number problem affects our entire wavelet-Galerkin method regardless of the scaling function order. It is also useful to note that the convergence rate is initially better than the standard finite element method.

Now that we have seen the error decrease as a result of the scaling function order we should make a note regarding the convergence rate of our wavelet-Galerkin method. First we will recall that in the finite element method we showed that the L^2 error was $\mathcal{O}(h^2)$ when using piecewise linear basis functions. We showed this using Galerkin's lemma and interpolation theory because our exact solution was adequately smooth. For the wavelet-Galerkin method, we note that the errors and convergence rate appears to indicate that the wavelet-Galerkin method is approximating the solution to the differential equation. We also notice that the condition number is relatively large

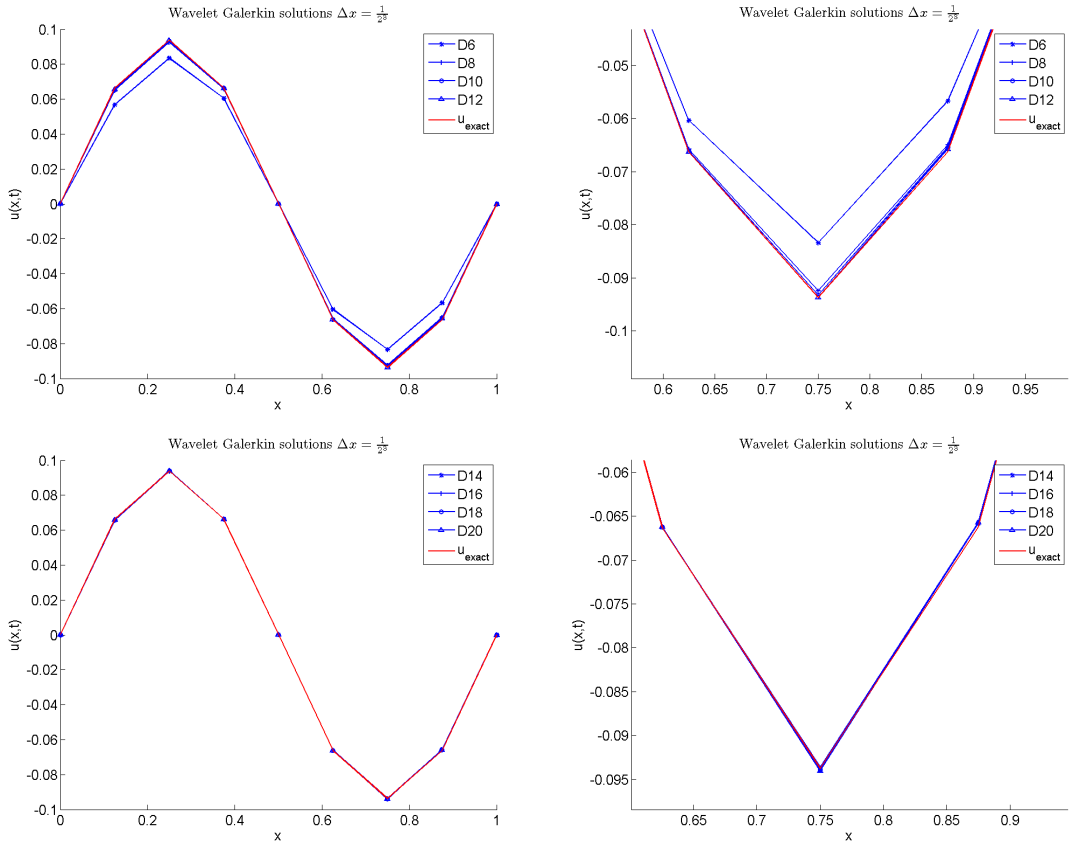


Figure 5.5: Example 2 Wavelet-Galerkin solution compared to actual solution at different ordered scaling functions (plots on the right are zoomed in)

for the $D10$ case, and it even grows as we increase the scaling function order. For example we see that for a fixed Δx, the condition number jumps from order 10^6 to order 10^{11} for $D10$ and $D20$ respectively. We believe that this may be an inadvertent result of using the fictitious boundary approach in order to enforce the Dirichlet boundary conditions. A QR decomposition was applied to the system with no significant improvement. Preconditioning could also be applied to better condition the matrix before solving but this was not a main goal of this research, so we will consider it in future work.

Table 5.4: Example 1: Euclidean error and the condition numbers for scaling functions: $D6$ through $D20$

Scaling function	Euclidean Error	Condition Number
D6	9.45617e-002	1.42459e+005
D8	2.41718e-002	7.27053e+005
D10	2.54864e-002	5.16747e+006
D12	2.97679e-002	3.76474e+007
D14	3.01458e-002	2.81199e+008
D16	3.12934e-002	2.10011e+009
D18	3.21286e-002	1.57091e+010
D20	3.03903e-002	1.19025e+011

Table 5.5: Example 2: Euclidean error and the condition numbers for scaling functions: $D6$ through $D20$

Scaling function	Euclidean Error	Condition Number
D6	1.15291e-001	1.42459e+005
D8	1.43859e-002	7.27053e+005
D10	7.56035e-003	5.16747e+006
D12	4.28288e-003	3.76474e+007
D14	3.96366e-003	2.81199e+008
D16	3.66746e-003	2.10011e+009
D18	4.70821e-003	1.57091e+010
D20	1.05054e-002	1.19025e+011

Table 5.6: Example 2: wavelet-Galerkin convergence rates using scaling function $D10$

h	Euclidean Error	Euclidean Rate	Condition number
0.250000	8.057278e-002		1.600595e+006
0.125000	9.316841e-003	3.236006	5.167466e+006
0.062500	2.698771e-003	1.652036	1.970388e+007
0.031250	1.212473e-003	1.000168	7.802231e+007

CHAPTER 6

REDUCED ORDER MODEL OF THE WAVELET-GALERKIN METHOD

The wavelet-Galerkin method has been shown to be a viable method for solving differential equations as documented in the references[14] [3] [13] [1] [16]. These works explore a variety of methods to overcome obstacles inherent in a wavelet-Galerkin method. Also, throughout the past few decades, the concept of reduced order modeling has been widely studied in the finite element setting[9] [7] [15]. One new application of the wavelet-Galerkin method that has yet to be examined is the implementation of a reduced order model built on a wavelet-Galerkin solution of a differential equation. Successful combination of a reduced order modeling approach to the existing wavelet-Galerkin method would combine a number of attractive features including the multi-resolution and orthogonality properties with a way to quickly reconstruct wavelet-Galerkin methods using POD. There are a number of applications that could benefit from some of the capabilities that are inherent to both of these methods. This work focuses on demonstrating the ability to create and implement a reduced order model based on the wavelet-Galerkin method.

This chapter will discuss how a POD basis using spapshots created from wavelet-Galerkin approximations can be generated in a similar manner to the method discussed in chapter 4 where we used the finite element method to create snapshots for our POD basis. The first section will discuss the reduced order setup including determining the parameter space and generating solution snapshots from the input parameters in the context of the one dimensional modified heat equation. We will then use these solution snapshots to generate a set of reduced basis vectors using the POD method that was discussed in the finite element reduced order modeling chapter. Following the reduced basis generation, we will use the reduced basis vectors and the definition of our wavelet-Galerkin method to construct a reduced order solution to the differential equation at a given parameter set within our parameter space. Finally we will present some results demonstrating the effectiveness of using a reduced order approach in combination with wavelet-Galerkin method in the context of our one dimensional modified heat equation.

6.1 Parameter Selection, Snapshot Generation and Reduced Basis Construction

There are no practical differences between the procedure to construct the reduced basis vectors in the finite element and wavelet-Galerkin methods. The concept of selecting a parameter space that defines the coefficients of our differential equation, remains the same. The one difference in this procedure is the actual snapshot generation. Instead of using the finite element method to determine solutions to the differential equation at a given parameter set, we will now use the wavelet-Galerkin method. And because we saw in the wavelet-Galerkin section that the solution to our differential equation could be represented directly by the solution to our linear system, we can simply treat the snapshot solution vectors as nodal basis solutions. So in this section we will be examining the highly oscillatory one dimensional modified heat equation example that was first presented in the finite element reduced order modeling section.

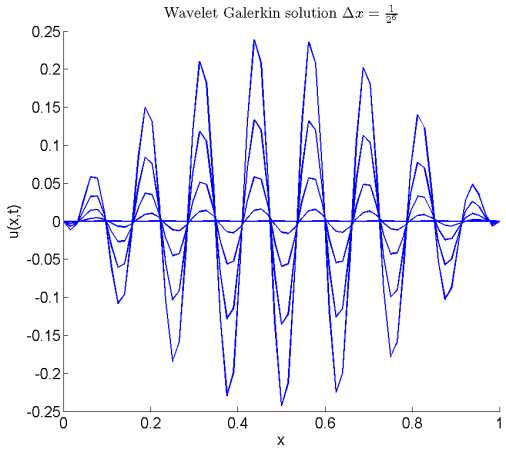
First we will remind ourselves of the one dimensional modified heat equation (equation 3.26) that was used to create a reduced order model for the finite element solution. For this problem we recall that we set the the right hand side of the equation equal to equation 6.1 shown here:

$$\begin{aligned} f(x,t) =& x(x-1)\left[\sin(50x) + 200t\cos(50x)\right] \\ & - t\left[2\left(\sin(50x) + 100t\cos(50x)\right) + 100(x-1)\left(\cos(50x) - 100t\sin(50x)\right)\right. \\ & \left. +100x\left(\cos(100x) - 100t\sin(50x)\right) + (50^2)x(x-1)\left(-\sin(50x) - 100t\cos(50x)\right)\right] \\ & + tx(x-1)\left(\sin(50x) + 100t\cos(50x)\right) \end{aligned} \tag{6.1}$$

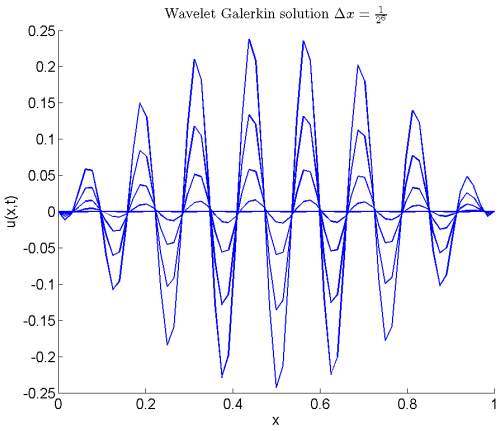
with parameters $a_1 = a_2 = a_3 = 1$, the exact solution can be show to be:

$$u(x,t) = tx(x-1)\left[\sin(50x) + 100t\cos(50x)\right] \tag{6.2}$$

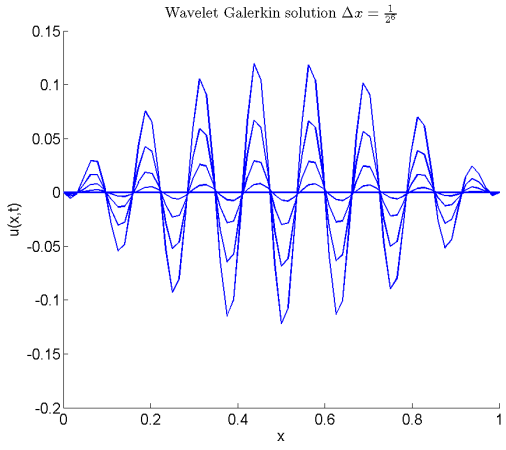
which describes a highly oscillatory function that is zero at the boundaries. Using this definition of the one dimensional modified heat equation, we should be able to construct a reduced order model based on the wavelet-Galerkin method.



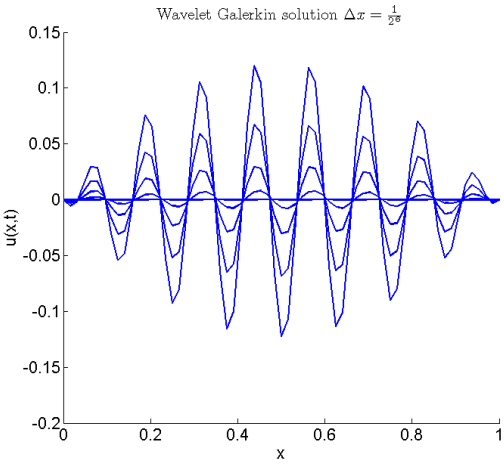
(a) $a_1 = 1$, $a_2 = 1$ and $a_3 = 1$



(b) $a_1 = 1$, $a_2 = 1$ and $a_3 = 2$



(c) $a_1 = 1$, $a_2 = 2$ and $a_3 = 1$



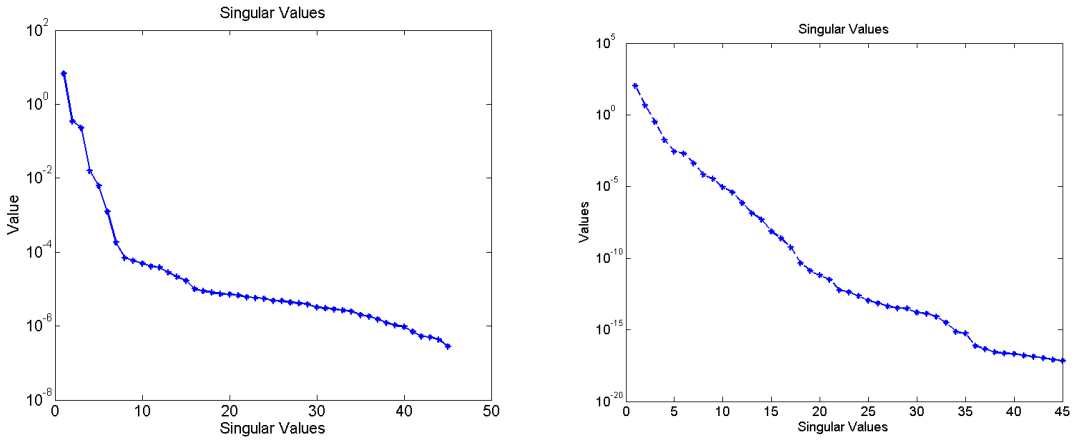
(d) $a_1 = 1$, $a_2 = 2$ and $a_3 = 2$

Figure 6.1: Sample FEM snapshots with $t = 0, 0.025, 0.050, 0.075, 0.100$

So to begin, we must choose an appropriate parameter space and decide how to sample said space. The most straightforward thing to do is to simply use the parameter space that was used in the finite element method to generate the snapshots of the differential equation. The parameters that define this differential equation: a_1, a_2 and a_3 were varied equally by steps of 1 between $[1, 1]$, $[0, 2]$ and $[0, 2]$ respectively for a total of nine parameter sets in order to generate a snapshot set that can be used to then generate reduced basis functions. We again notice that the first parameter was not varied as we could just divide by the leading coefficient to get another parameter set, this

should decrease redundancy. So Figure 6.1 shows the snapshot set of the wavelet-Galerkin solutions to the one dimensional modified heat equation with homogeneous Dirichlet boundary conditions using scaling function $D12$ with $\Delta x = \frac{1}{2^6}$ and $\Delta t = \Delta x^2$ where we have sampled randomly over 5 time steps. Thus with this choice of parameter sets we have a total of 45 snapshots, which is the same setup as the finite element reduced order model.

We can compare Figures 6.1 and 4.2 to see that using the same input parameter sets, generates similar solutions to the differential equation using the wavelet-Galerkin and finite element methods. Now that the snapshot sets have been generated, the next step is to determine the reduced basis vectors for the solution. To do this the same singular value decomposition that was performed in the finite element case will be applied here. First we examine Figure 6.2 which presents the singular values from the singular value decomposition on the wavelet-Galerkin snapshot set compared to the singular values from the finite element method.



(a) wavelet-Galerkin (b) finite element

Figure 6.2: Singular Values of the snapshot sets

We can see from Figure 6.2 that the singular values decay as expected with the first few basis vectors containing the majority of the information about the snapshot set. We also note that by the fourth singular value, the magnitude of the remaining singular values has become less than 10^{-4}. If we compare the wavelet-Galerkin and finite element singular values, we may notice that the wavelet-Galerkin singular values don't appear to decrease to the same magnitude as the finite element singular values. This means that the order of the error between our reduced order and

wavelet-Galerkin approximations may not decrease to as high of an order as the finite element method. But the general "elbow" shape of the wavelet-Galerkin singular values is more desirable than the log linear trend for the finite element singular values. Regardless, we should be able to construct a reasonable reduced order approximation using only a few of the wavelet-Galerkin reduced basis vectors.

Along with the singular values, it is also useful to visualize the reduced basis vectors to get an idea of the major modes or trends contained within the snapshot set. Figure 6.3 presents the first four basis functions for the one dimensional modified heat equation solved when using the wavelet-Galerkin method.

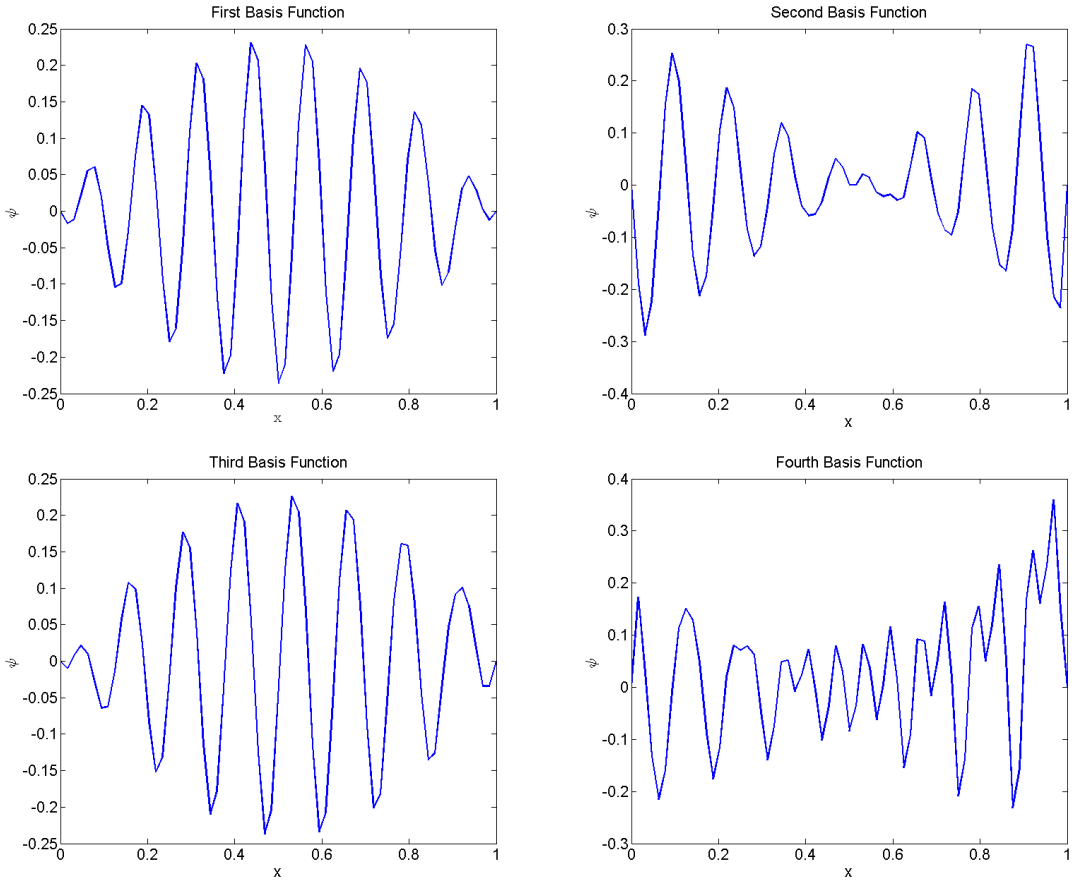


Figure 6.3: First four basis vectors of the wavelet-Glerkin solution approximation to the one dimensional modified heat equation

It is interesting to note in these plots that the first and the third basis vectors appear to contain

the general shape of the solution whereas the second and fourth appear to be a little more noisy. We should recall that the solution near the boundaries appeared to account for a majority of the error in the finite element reduced order case. So it is interesting to note that the second basis vector appears to contain most of the information near the boundaries of our domain.

6.2 Construction of the Reduced Wavelet-Galerkin Solution

Up until this point the steps taken to generate and use a reduced order model in a wavelet-Galerkin setting have been effectively the same as the steps used to generate a finite element method reduced order model. But now we must alter the way that we construct our reduced order approximation to the differential equation. This must be done because our global basis functions are no longer expressed in terms of piecewise linear functions but rather in terms of the Daubechies scaling functions. The approach is similar in that we must construct a linear system that is dense but hopefully small. It is this idea that makes these reduced order approaches attractive, there are many cases where we would rather solve a small dense system than a large sparse system. Due to the linearity of our examples, we can expect that the number of basis functions required to accurately represent the solution will be very small as indicated by our singular values.

In order to construct a reduced order model solution in the wavelet-Galerkin method we must first define the reduced basis vectors. Equation 6.3 defines the reduced basis vector, $\psi_i(x)$, in terms of a general Daubechies scaling function, $\phi(2^m x)$ at level m, used in our wavelet-Galerkin method. We define l as all possible shifts to the left and right within our domain, including no shift.

$$\psi_i(x) = \sum_l \sigma_{i,l} 2^{\frac{m}{2}} \phi(2^m x - l) \tag{6.3}$$

where $\sigma_{i,l}$ defines the coefficients for basis vector, ψ_i. Using this reduced basis definition we can express the reduced order solution (u^k_{rom}) in terms of a linear combination of the reduced basis functions and a set of scaling terms, c^k_j, at a fixed time t^k,

$$u_{rom} = \sum_{j=1}^{nB} c^k_j \psi_j(x) \tag{6.4}$$

where nB is the number of basis functions we intend on including in the construction of our reduced order solution. Now that we have defined the reduced basis functions, we must turn back to the

original weak form of the one dimensional modified heat equation to generate a weak form based on our reduced order solution. We will start by writing the weak form in terms of the reduced order solution multiplied by a test reduced basis function:

$$\begin{aligned} a_1 \int \frac{\partial u_{rom}}{\partial t}\psi_i(x)dx - a_2 \int u''_{rom}\psi_i(x)dx + a_3 \int u_{rom}\psi_i(x)dx \\ = \int f\psi_i(x)dx \text{ for } i = 1, \dots \ nB \end{aligned} \tag{6.5}$$

Then by substituting in the definition for the reduced order solution to the differential equation and breaking up the temporal derivative into a finite difference approximation we get:

$$\begin{aligned} \sum_{j=1}^{nB} c_j^k \left[\frac{a_1}{\Delta t} \int \psi_j\psi_i dx - \ a_2 \int \psi''_j\psi_i dx + a_3 \int \psi_j\psi_i dx \right] \\ = \int f\psi_i dx + \frac{a_1}{\Delta t} \int u_{rom}^{k-1}\psi_i dx \end{aligned} \tag{6.6}$$

where in this equation k refers to the time step. Similar to the analysis of the right hand side of the wavelet-Galerkin method we presented in the wavelet-Galerkin chapter, if we expand f and u_{rom} into the wavelet space we will end up just representing the right hand side by the function values at each nodal point. For the non-derivative and temporal terms we can substitute in equation 6.3 to find the inner product of ψ_j and ψ_i:

$$\int \psi_j\psi_i = 2 \int \sum_l \sigma_{j,l}\phi(2x - l) \sum_p \sigma_{i,p}\phi(2x - p) \tag{6.7}$$

where we define p in the same manner as l as all possible shifts to the left and right within our domain, including no shift. If we take this equation and re-arrange it we can find that the inner product becomes a set of summations multiplied by the definition of the inner product of our scaling functions:

$$\int \psi_j\psi_i = 2 \sum_l \sigma_{j,l} \sum_p \sigma_{i,p} \int \phi(2x - l)\phi(2x - p) \tag{6.8}$$

And because of the orthonormality properties of our Daubechies scaling functions, we know that the inner product in equation 6.8 becomes the Kronecker-delta function. Meaning that to compute $\int \psi_j\psi_i$ we simply need to compute n dot products of σ when $l = p$, where n is the number of scaling basis functions. Thus our weak form is significantly reduced to:

$$\sum_{j=1}^{nB} c_j \left[\left(\frac{a_1}{\Delta t} + a_3 \right) \sum_l \sum_p \sigma_{j,l} \sigma_{i,p} \delta_{l,p} - a_2 \int \psi_j'' \psi_i dx \right] = \sum_l f \sigma_{i,l} + \frac{a_1}{\Delta t} u_{rom}^{k-1} \sigma_{i,l} \tag{6.9}$$

So the final term that we need to deal with is the second derivative term. In order to resolve this term we must expand the definition of our reduced basis functions given in equation 6.3. Using this notation we are left with the product of two summations:

$$\int \left[\sum_l \sigma_{j,l} 2^{\frac{m}{2}} \phi''(2^m x - l) \right] \left[\sum_p \sigma_{i,p} 2^{\frac{m}{2}} \phi(2^m x - p) \right] \tag{6.10}$$

From equation 6.10 one can begin to imagine how these two summations are going to interact with the connection coefficients. We can combine the summations in such a way that one basis vector will stay stationary while we compute the dot product for every basis vector that has an effect on our stationary basis vector. Thus we can generalize this equation into an algorithmic sense:

$$\Psi_{i,j} = \sum_{l=0}^{n} \sigma_{j,l} \sum_{q=0}^{n} \sigma_{i,q} \Gamma_q^2 \tag{6.11}$$

ensuring that the summations are only taking place where l and q overlap and where $\Psi_{i,j}$ defines the elements of our linear system. A pseudo-code algorithm that we follow to construct this matrix is shown below:

```
Loop over all basis functions to include (j)
   Loop over all basis functions to include (i)
      Loop over all the points in the domain (l)
   PSI(j,i) = a_2*sigma(k,j)*dot_product(sigma(k-(N-2):k+(N-2),i),Gamma)
```

where we define `sigma` as a column-wise vector of our reduced basis functions and `Gamma` is the set of connection coefficients that we have pre-computed. This method is similar to the finite element method in that we must loop over all the basis functions to generate the elements within our linear system. One distinguishing factor between this method and the finite element method is that this method is easily parallelizable. Because the core part of this method requires a number of dot products to be computed from existing and precomputed vectors, the elements of the matrix can be formulated independently which is crucial for parallelization.

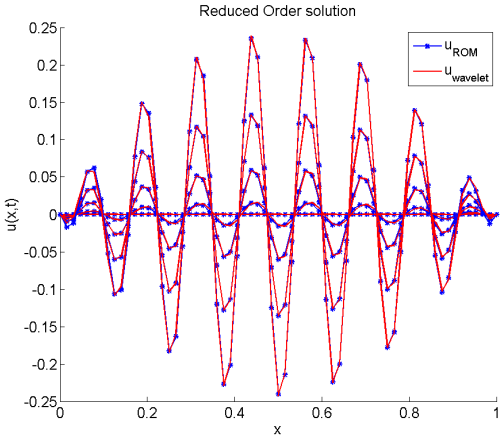
6.3 Results

With a method for determining the solution to the differential equation for any parameter set within our parameter space we can now construct a reduced order solution to the one dimensional modified heat equation with homogeneous Dirichlet boundary conditions. We will examine two cases of the prototype modified heat equation that we have defined for this chapter:

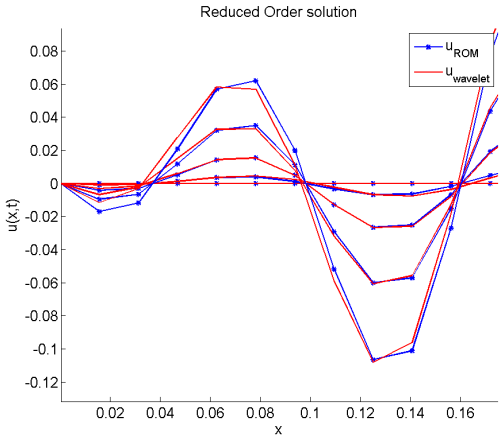
$$\begin{aligned} &\text{Example 1: } a_1 = a_2 = a_3 = 1 \\ &\text{Example 2: } a_1 = 1,\ a_2 = 1.5,\ \text{and } a_3 = 0.75 \end{aligned} \tag{6.12}$$

where we note that the parameters in the first example exist in the snapshot set, whereas the parameters in the second example do not. These are the same examples that were used in the finite element case. Both examples will be used to show that we can use this reduced order model to approximate solutions to the differential equation that exist in our parameter space. Using this definition of the differential equation and the reduced basis functions we have generated, we can construct reduced order solutions to the differential equation.

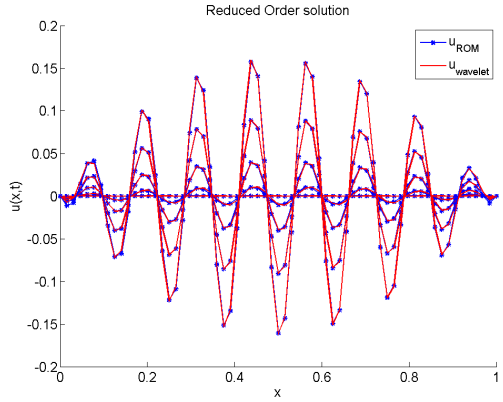
So first we will attempt to construct the reduced order wavelet-Galerkin solution using only a single basis function to better understand how well the reduced basis vectors can be used to represent the solution to the differential equation. From Figure 6.4 we can see that using onlly a single basis function yields a reasonable estimation of the wavelet-Galerkin solution. This is expected as we are not only solving a linear problem but also approximating a solution that exists in our snapshot set. Figure 6.4 also includes an arbitrarily choosen zoomed in plot near the left hand boundary that illustrates some separation between the standard wavelet-Galerkin and the reduced order solution. We should remind ourselves that we should compare the reduced order solution to the standard wavelet-Galerkin solution because the reduced order solution is based on the information generated from the snapshots and has no knowledge of the exact solution. Also if we recall from the finite element reduced order model, only a few basis functions were needed to approximate the solution to a decent precision. The reason for this is the problem is linear and can be easily represented by only a few basis functions. But we do know that as we include more and more reduced basis functions, the reduced order solution should continue to improve.



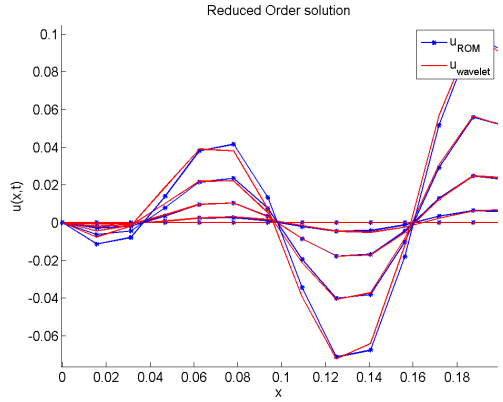
(a) Example 1: Using one basis function



(b) Example 1: Zoom-in view of the left boundary



(c) Example 2: Using one basis function

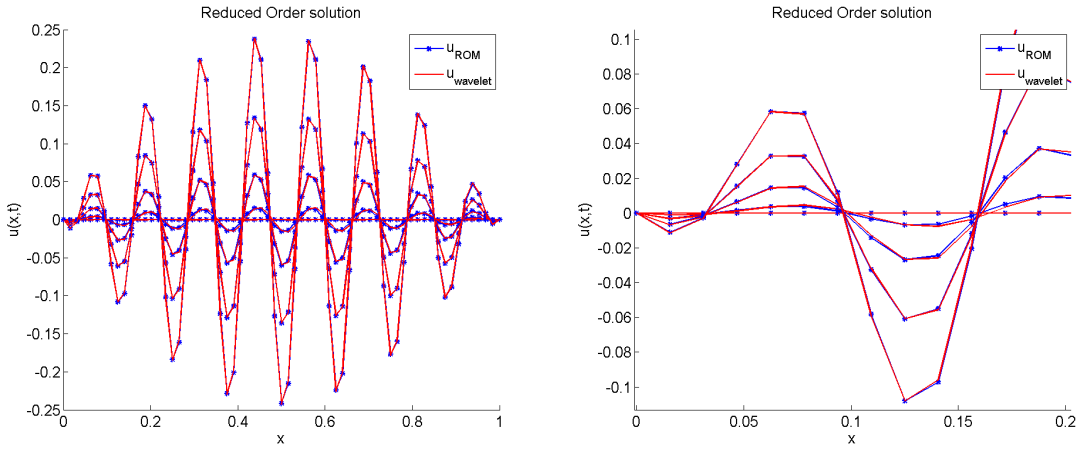


(d) Example 2: Zoom-in view of the left boundary

Figure 6.4: Five time instances of the reduced order solution compared to the wavelet-Galerkin solution using only a single basis function

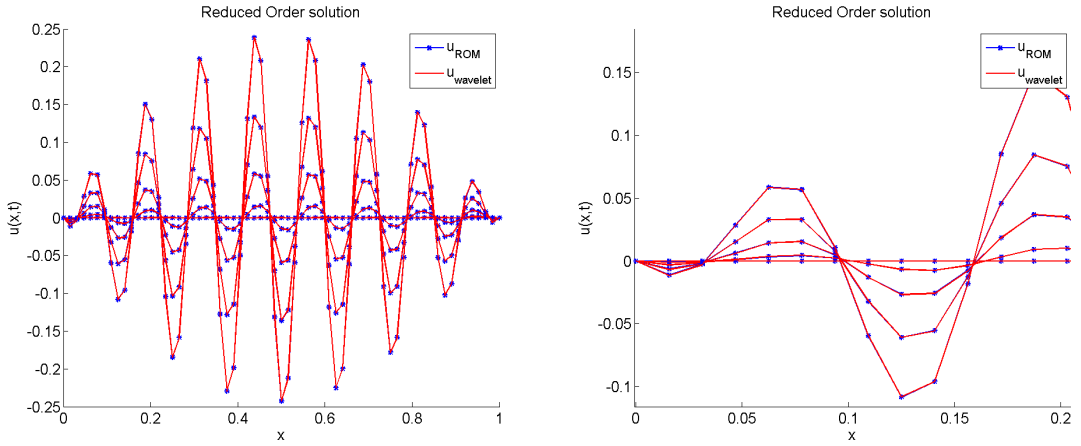
Although only a single basis function appears to generate a reasonable approximation to the wavelet-Galerkin solution to the differential equation, we should examine a few more cases where we increase the number of basis functions included. We wish to show how increasing the number of basis functions included, generates a better approximation to the wavelet-Galerkin solution. So Figures 6.5 and 6.6 display the reduced order solution compared to the wavelet-Galerkin solution for the cases when two and three basis functions are included in the reduced order solution construction.

We also include the zoomed in portion near the boundary to better visualize how the reduced order solution improves as the number of basis functions is increased.



(a) Using two basis functions

(b) Zoom-in view of the left boundary



(c) Using three basis functions

(d) Zoom-in view of the left boundary

Figure 6.5: Example 1: Five time instances of the reduced order solution compared to the wavelet-Galerkin solution using two and three basis functions

As we can see clearly from the zoomed-in plots, the reduced order solution approaches the standard wavelet-Galerkin solution as we increase the number of basis functions in the solution construction. Further we can recognize this trend by computing the Euclidean error as the number of basis functions is increased. Tables 6.1 and 6.2 compare the Euclidean error between the reduced order wavelet-Galerkin and the finite element approximations.

Table 6.1: Example 1: Euclidean error comparison between wavelet-Galerkin and finite element reduced order models

Basis functions	Wavelet-Galerkin Error	FEM Error
1	3.66375e-002	6.48628e-001
2	2.20387e-002	2.87067e-001
3	2.33198e-003	2.06806e-001
4	2.36702e-003	1.50642e-002

Table 6.2: Example 2: Euclidean error comparison between wavelet-Galerkin and finite element reduced order models

Basis functions	Wavelet-Galerkin Error	FEM Error
1	3.70628e-002	6.44850e-001
2	2.21079e-002	2.89482e-001
3	2.15290e-003	2.06052e-001
4	2.24912e-003	1.00612e-002

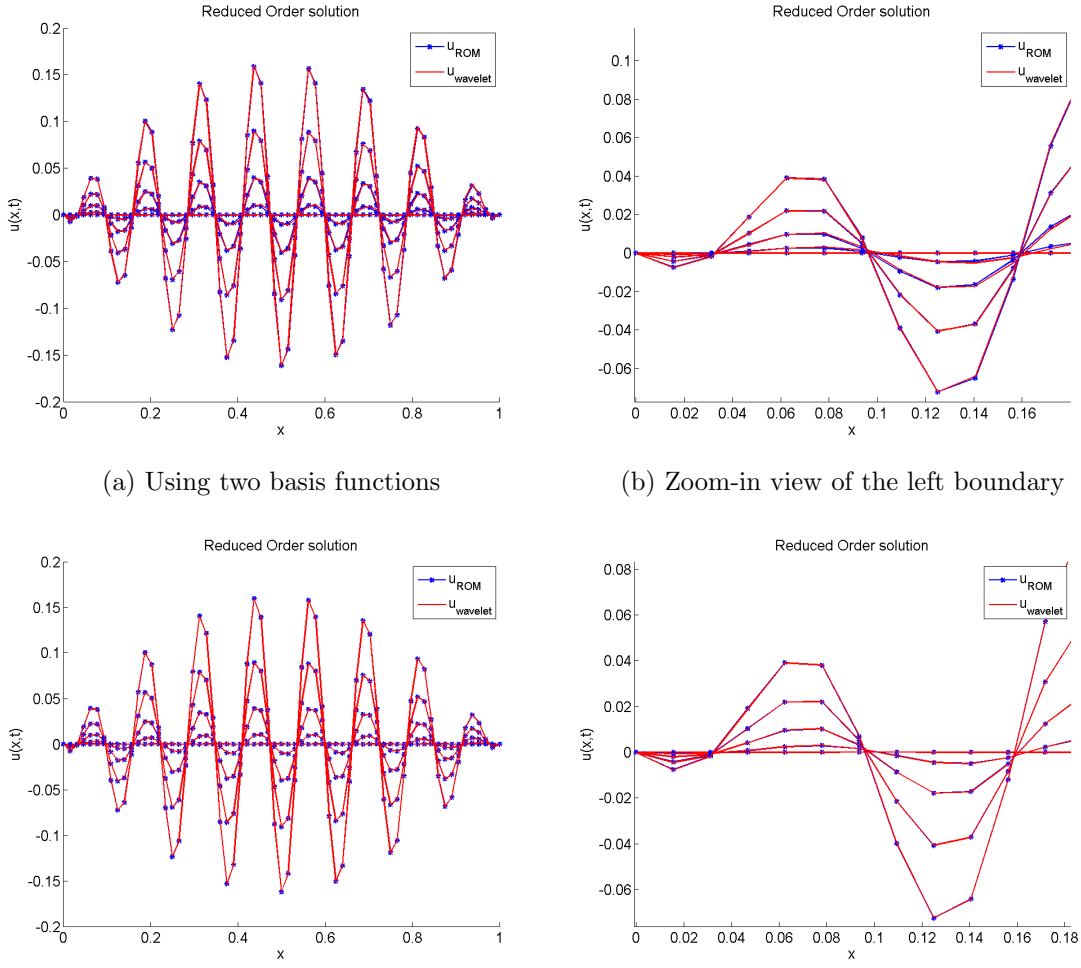
From Tables 6.1 and 6.2 we notice that the error does indeed decrease as we include more basis functions. We should also note that the wavelet-Galerkin error magnitude appears to be smaller than the error in the finite element case. This shows that the reduced order model that we have created based on the wavelet-Galerkin method and then used to construct a solution appears to be a valid method for computing the reduced order solution to a differential equation.

6.4 Conclusions and Future Work

We have shown that the wavelet-Galerkin method can be used to build a set of reduced basis functions that can then be used to construct a reduced order solution to the differential equation with a given set of parameters. Two example problems has been examined to show the differences between a standard finite element approach and the wavelet-Galerkin method. We were able to show how the wavelet-Galerkin method appears to converge to the exact solution and results are promising going forward in the use of Daubechies scaling functions as bases. Also we found that when comparing to FEM, the wavelet-Galerkin method is easier to parallelize in the standard method and the reduced order model construction. Finally the construction of our reduced order

model wavelet-Galerkin solution was presented in the example context and shown to be a viable method at estimating the solution to the differential equation with a given set of parameters.

Research of this type lends itself to numerous opportunities for future work. One of the most pressing issues for this research is that the methods shown here should be applied to a nonlinear problem in order to test the resiliency of a reduced order modeling approach. This will involve the calculation of so-called 3-term connection coefficients due to the nonlinear terms in our weak formulation. To go along with this future work a more concrete problem with a more specific application, potentially in two or three dimensions, should be examined to understand how our wavelet-Galerkin method works in a real-world problem. Additionally, there are more improvements that could be made to improve the precision of the wavelet Galerkin method including finding a better way to numerically determine the connection coefficients so as to avoid poorly conditioned systems. Finally it would be interesting to use a more complex wavelet, like a bi-orthogonal wavelet (currently being used in jpeg image compression) within our wavelet-Galerkin system in order to apply some of the properties that go along with such a wavelet.



(a) Using two basis functions

(b) Zoom-in view of the left boundary

(c) Using three basis functions

(d) Zoom-in view of the left boundary

Figure 6.6: Example 2: Five time instances of the reduced order solution compared to the wavelet-Galerkin solution using two and three basis functions

BIBLIOGRAPHY

[1] E. Tenenbaum A. Latto, H. L. Resnikoff. The evaluation of connection coefficients of compactly supported wavelets. In *Proceedings of the French-USA Workshop on Wavelets and Turbulence*. Springer-Verlag, New York, 1991.

[2] I Babuvška and Werner C Rheinboldt. Error estimates for adaptive finite element computations. *SIAM Journal on Numerical Analysis*, 15(4):736–754, 1978.

[3] Jordi Besora. Galerkin wavelet method for global waves in 1d. *Masters degree project stockholam, Sweden*, 2004.

[4] Dietrich Braess. *Finite elements: Theory, fast solvers, and applications in solid mechanics*. Cambridge University Press, 2007.

[5] C Sidney Burrus, Ramesh A Gopinath, Haitao Guo, Jan E Odegard, and Ivan W Selesnick. *Introduction to wavelets and wavelet transforms: a primer*, volume 23. Prentice hall Upper Saddle River, 1998.

[6] Ingrid Daubechies et al. *Ten lectures on wavelets*, volume 61. SIAM, 1992.

[7] Qiang Du and Max Gunzburger. Model reduction by proper orthogonal decomposition coupled with centroidal voronoi tessellation. In *Proc. Fluids Engineering Division Summer Meeting, FEDSM2002-31051, ASME*, 2002.

[8] Stefan Goedecker. *Wavelets and their application for the solution of partial differential equations in physics*. Presses polytechniques et universitaires romandes, 1998.

[9] Max D. Gunzburger, Janet S. Peterson, and John N. Shadid. Reduced-order modeling of time-dependent PDEs with multiple parameters in the boundary data. *COMPUTER METHODS IN APPLIED MECHANICS AND ENGINEERING*, 196(4-6):1030–1047, 2007.

[10] Alfred Haar. Zur theorie der orthogonalen funktionensysteme. *Mathematische Annalen*, 69(3):331–371, 1910.

[11] Peter D Lax and AN Milgram. Parabolic equations. *Selected Papers Volume I*, pages 8–31, 2005.

[12] Dianfeng Lu, Tadashi Ohyoshi, and Lin Zhu. Treatment of boundary conditions in the application of wavelet-galerkin method to an sh wave problem. *INTERNATIONAL JOURNAL-SOCIETY OF MATERIALS ENGINEERING FOR RESOURCES*, 5:15–25, 1997.

[13] Vinod Mishra. Sabina, wavelet-galerkin solutions of ordinary differential equations, int. *Journal of Math. Analysis*, 5:407–424, 2011.

[14] Ole Møller Nielsen. *Wavelets in scientific computing.* PhD thesis, Technical University of Denmark, 1998.

[15] BJ ODonnell and BT Helenbrook. Proper orthogonal decomposition and incompressible flow: An application to particle modeling. *Computers & fluids*, 36(7):1174–1186, 2007.

[16] CH Romine and BW Peyton. Computing connection coefficients of compactly supported wavelets on bounded intervals. *Computer Science and Mathematical Division, Mathematical Sciences Section, Oak Ridge National Laboratory, Oak Ridge, Techical Report ORNL/TM-13413, http://citeseer. ist. psu. edu/romine97computing. html*, 1997.

[17] Gilbert Strang and Truong Nguyen. Wavelets and filter banks. 1996. *Wellesley-Cambridge Press, Wellesley, MA*, 1996.

[18] Vidar Thomée. *Galerkin finite element methods for parabolic problems*, volume 25. Springer, 2006.

[19] Bryan E Usevitch. A tutorial on modern lossy wavelet image compression: foundations of jpeg 2000. *Signal Processing Magazine, IEEE*, 18(5):22–35, 2001.

BIOGRAPHICAL SKETCH

David Witman received a BS in Mechanical Engineering in May of 2011 from Clarkson University in Potsdam New York. In August of 2011 he began graduate work in the Department of Scientific Computing at The Florida State University in Tallahassee, Florida.