THE UNIVERSITY OF CALGARY

An Adaptive Local Scaling Function Representation

by

Bryan Quaife

A DISSERTATION
SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE
DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF MATHEMATICS AND STATISTICS

CALGARY, ALBERTA

August, 2006

© Bryan Quaife 2006

THE UNIVERSITY OF CALGARY

FACULTY OF GRADUATE STUDIES

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies for acceptance, a dissertation entitled "An Adaptive Local Scaling Function Representation" submitted by Bryan Quaife in partial fulfillment of the requirements for the degree of MASTER OF SCIENCE.

Supervisor, Dr. A.F. Ware
Department of Mathematics and Statistics

Dr. L.P. Bos
Department of Mathematics and Statistics

Dr. F.F. Samavati
Department of Computer Science

Date

Abstract

Wavelet analysis is one of many methods to represent arbitrary square-integrable functions. One of the benefits of using this representation comes from the locality of the wavelets. This allows one the flexibility to generalize to an adaptive grid with very little extra effort.

This thesis will examine how one can construct one representation called an adaptive local scaling function representation. This requires an investigation of how to approximate scaling function coefficients and how to predict an appropriate grid. At this point, the adaptive projection can be formed.

Applications of a local scaling function representation will be considered and some generalizations will be mentioned.

Acknowledgements

First, I need to thank my supervisor Dr. A.F. Ware for his advice and patience with my research. I have learned a lot of Mathematics from him over the previous two years. I must also thank the many faculty members who assisted me with my research, taught me courses and offered encouragement during my Master's program.

Next, my family who have always supported me to pursue my dreams, especially my parents who reminded me that I should go to school as long as I want.

Finally and most importantly is my fiance Lindsey. She has waited patiently for me many nights when I worked late on my research, and has signed up to do it all over again.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Representing functions in an optimal manner is a problem that can be studied from different angles. One of the earliest more sophisticated methods of representing a function was accomplished by Fourier when he developed what we now call Fourier analysis. However, function representation in Fourier analysis is a non-localized approximation.

A more recent method of representing functions that is local uses techniques of wavelet analysis. There is still a lot of work to be done in wavelet analysis in order to understand it at the level we understand Fourier analysis, but there are promising results coming out of wavelet analysis. Cohen gives a nice historical summary of ideas leading up to wavelet analysis in [5]. The first record of a wavelet was accomplished by Haar in his Ph.D. dissertation in 1909. Throughout the next 80 years, there were hints of wavelet analysis coming from time-frequency and time-scale analysis, harmonic analysis, approximation theory and multiresolution image processing. However, it was not until the late 1980s that wavelet analysis was truly born.

A thorough comparison of Fourier analysis and wavelet analysis is summarized by Gilbert Strang in [17]. The paper was published in 1993 when wavelets were relatively new, yet he does a nice comparison of the benefits of each technique for constructing representations of functions.

One of the most appealing and important properties of wavelet analysis is that adaptivity is relatively easy to implement. In other words, the distance between

any two points in the partition of a domain need not be fixed. The simplicity in adaptivity is inherited from the locality of wavelets. The importance of adaptivity is that it allows us to refine our grid in problematic regions while leaving a coarser grid in well-behaved regions. This means that we can distribute the error in our representation evenly throughout the entire domain.

As this thesis is concerned with representing compactly supported functions on an adaptive grid, there is a large computational component to it. Thus, we will always be concerned with the computational cost of algorithms. After looking at results, illustrations taken from Matlab at times will be given in order to demonstrate some of the results we will be proving.

In Chapter 2, we will look at some necessary preliminary results. As notation is heavy in this thesis, a list of notation is given in this chapter. As well, we will look at some function spaces that are crucial to studying wavelet analysis. These spaces will become important when we look at sequences of approximations and when we decide how to partition domains of functions.

Chapter 3 will introduce the wavelet setting. A thorough construction will not be considered as we will only require certain results and definitions. References to more detailed constructions of wavelet analysis will also be given in this chapter. We will look at some computational aspects of wavelet analysis and finally, some error bounds will be stated without proof.

Chapter 4 will be devoted to methods of approximating the necessary coefficients for constructing representations of functions. A brief description of one method of approximating these coefficients will be given, and then we will move on to the major result of this chapter. That is, we will look at how to define a quasi-

interpolation scheme and properties following from the definition. Two examples of quasi-interpolation schemes, along with error bounds, will be given. A problematic consequence of quasi-interpolants will be considered and we will see that it is in some sense unavoidable. Finally, an iterative technique that decreases the error in the approximations will be discussed.

In Chapter 5, we will consider two representations of a function in the wavelet setting. As these representations are essentially identical, we will only study one of the representations in depth. More specifically, we will study local scaling function representations. In this chapter, the notion of adaptive grids will be introduced. Constructing an adaptive local scaling function representation asks the questions of how to decide on an adaptive grid to project the function on to, and how to construct the projection on to the adaptive grid. Both these questions will be answered by the DSX algorithm. We will then show how this algorithm can be simplified by selecting an appropriate quasi-interpolation scheme. As well, throughout the chapter, some necessary assumptions will be stated and justified.

Chapter 6 looks at some applications of adaptive local scaling function representations. Some of the simpler results such as numerical differentiation and numerical integration of real-valued functions are based on performing point evaluation. Thus, point evaluation will be looked at in depth. As well, some of the classical applications of wavelet analysis will be briefly discussed in this setting. Finally, possible generalizations of this thesis will be stated.

Concluding the thesis, a summary of results will be given in Chapter 7.

Chapter 2

Preliminaries

2.1 Notation

Unfortunately, this thesis is quite heavy in notation. This section gives a summary of some of the notation that will be used.

$\mathbb{R}$	the space of real numbers.
$\|f\|_X$	the norm of the vector f in the vector space X.
$\inf \sup_{x \in \mathbb{R}} \|f(x)\|$	the infimum over sets of measure 0 of the supremum of $\|f(x)\|$.
$\langle f, g \rangle_{L_2(\mathbb{R})}$	the usual inner product on $L_2(\mathbb{R})$.
$\mathbb{N}$	the space of non-negative integers.
$\langle x, y \rangle_{l^2(\mathbb{R})}$	the usual inner product on $l^2(\mathbb{R})$.
D	the weak differential operator.
$f^{(i)}$	the i^{th} weak derivative of f.
Δ_n^h	the n^{th} forward difference operator in the direction h.
span X	the space of all linear combinations of elements of X.
$\overline{X}$	the norm closure of X.
$\mathbb{Z}$	the space of integers.
$X \oplus Y$	the direct sum of the vector spaces X and Y.
$X \perp Y$	every element of X is orthogonal to every element of Y.
λ	an ordered pair containing the scale j and shift k.
$\|\lambda\|$	the scale j of λ.

$\delta_{x,y}$	the Kronecker delta of x and y.
χ_X	the characteristic function of the set X.
$A \otimes B$	the Kronecker tensor product of A and B.
Π_d	the space of polynomials of degree less than or equal to d.
$x \lesssim y$	$x < ky$ where k is independent of x and y.
$x \sim y$	$x \lesssim y$ and $y \lesssim x$.
supp f	the set of real numbers x where $f(x) \neq 0$.
$\hat{f}(\omega)$	the Fourier transform of f evaluate at ω.
$f\|_X$	the function f restricted to the set X.
M_i	the i^{th} continuous moment of ϕ.
m_i	the i^{th} discrete moment of h_k.
$\binom{n}{k}$	n choose k.
n_i	the i^{th} discrete moment of g_k.
$\hat{f}_k$	the k^{th} coefficient of the discrete Fourier transform of $f(x_k)$.
T^*	the adjoint operator of the operator T.
$\|X\|$	the Lebesgue measure of the set X.
$\|\Lambda\|$	the counting measure of the set Λ.
w_λ	the support of ϕ_λ.

2.2 Function Spaces

Function spaces are linear spaces where the elements are functions. Addition and scalar multiplication are defined in the obvious manner. Some function spaces arise more naturally than others in wavelet analysis. All of our function spaces, except for

one, will be equivalence classes of functions where f is equivalent to g if $f(x) = g(x)$ almost everywhere. That is, two functions are equivalent if they agree everywhere except for on a set of Lebesgue measure zero.

We will only be considering function spaces that are equipped with a norm. Thus, the notion of a Cauchy sequence can be defined. A Banach space is defined as a normed function space such that every Cauchy sequence converges inside the function space. A special case of a Banach space is a Hilbert space. A Hilbert space is simply a Banach space whose norm is induced by an inner product in the usual manner. Thus we can define orthogonality and have intuitive projections in a Hilbert space.

This thesis is only concerned in real-valued functions. That is, we are only considering functions that map $\mathbb{R}$ to $\mathbb{R}$.

2.2.1 L_p Spaces

A real-valued function $f : \mathbb{R} \to \mathbb{R}$ is said to be in the space L_p, where p is an integer and $1 \leq p < \infty$, if the Lebesgue integral

$$\int_{\mathbb{R}} |f|^p$$

is finite. If $f \in L_p(\mathbb{R})$, the norm of f is defined as

$$\|f\|_{L_p(\mathbb{R})} := \left(\int_{\mathbb{R}} |f|^p \right)^{1/p}.$$

The space $L_\infty(\mathbb{R})$ is the space of real-valued functions $f : \mathbb{R} \to \mathbb{R}$ such that the quantity

$$\inf \sup_{x \in \mathbb{R}} |f(x)|$$

is finite. In this case, the norm of f is defined as

$$\|f\|_{L_\infty(\mathbb{R})} := \inf \sup_{x\in\mathbb{R}} |f(x)|.$$

$L_\infty(\mathbb{R})$ is the space of real-valued functions that are bounded almost everywhere.

For $1 \leq p \leq \infty$, it can be shown that $L_p(\mathbb{R})$ is a Banach space. So, discussing the convergence of a sequence of approximations in $L_p(\mathbb{R})$ makes sense. The space $L_2(\mathbb{R})$, called the space of square-integrable functions, is a Hilbert space equipped with the inner product

$$\langle f, g\rangle_{L_2(\mathbb{R})} := \int_{\mathbb{R}} fg.$$

If it is clear that $f, g \in L_2(\mathbb{R})$, then we will simply write

$$\langle f, g\rangle = \langle f, g\rangle_{L_2(\mathbb{R})}.$$

The only L_p space that we will be considering is $L_2(\mathbb{R})$. At times, we will also write

$$L_p = L_p(\mathbb{R}).$$

2.2.2 l^p Spaces

$l^p(\mathbb{R})$ is the discrete case of $L_p(\mathbb{R})$ spaces. They are the only function space that we will consider that do not require equivalence classes. A real sequence,

$$x = (x_1, x_2, \ldots).$$

indexed over $\mathbb{N}$, is said to be in $l^p(\mathbb{R})$ (where p is an integer satisfying $1 \leq p < \infty$) if

$$\left(\sum_{n\in\mathbb{N}} |x_n|^p\right)$$

is finite.

At times, we will have a sequence indexed over a partially ordered set other then $\mathbb{N}$. However, the properties of these sequences will be clear in the context. Note that if $x \in l^p(\mathbb{R})$, assuming that x is indexed over $\mathbb{N}$, then

$$\lim_{n\to\infty} x_n = 0.$$

If $x \in l^p(\mathbb{R})$, the norm of x is defined as

$$\|x\|_{l^p(\mathbb{R})} := \left(\sum_{n\in\mathbb{N}} |x_n|^p\right)^{1/p}.$$

In the case of $p = \infty$, we say that $x \in l^\infty(\mathbb{R})$ if

$$\sup_{n\in\mathbb{N}} |x_n|$$

is finite, and its norm is given by

$$\|x\|_{l^\infty(\mathbb{R})} := \sup_{n\in\mathbb{N}} |x_n|.$$

Thus, $l^\infty(\mathbb{R})$ is the space of bounded sequences.

$l^p(\mathbb{R})$ is a Banach space for all $1 \leq p \leq \infty$ and the case of $p = 2$ is a Hilbert space. The inner product on $l^2(\mathbb{R})$ is given by

$$\langle x, y\rangle_{l^2(\mathbb{R})} := \sum_{n\in\mathbb{N}} x_n y_n.$$

Again, if it is clear that $x, y \in l^2(\mathbb{R})$, then we will simply write

$$\langle x, y\rangle = \langle x, y\rangle_{l^2(\mathbb{R})}.$$

Also, at times we will write

$$l^p = l^p(\mathbb{R}).$$

2.2.3 Sobolev Spaces

If $f \in L_p$, there are no restrictions on the derivatives of f. So, a subspace of L_p can be constructed by putting conditions not only on the function, but also on its derivatives in the sense of distributions. A function f is in some Sobolev space if f and some of its weak derivatives are all in some L_p space. There is a nice relationship, as we will see, between Sobolev norms and wavelet analysis.

Let $D^\alpha f$ denote the weak derivative of order α of f. Let s be a positive integer and p be a positive integer or infinity. Then a Sobolev space of order s is denoted by $W_p^s(\mathbb{R})$ and is equipped with the norm

$$\|f\|^p_{W_p^s(\mathbb{R})} := \sum_{0 \le \alpha \le s} \|D^\alpha f\|^p_{L_p(\mathbb{R})}.$$

The elements of $W_p^s(\mathbb{R})$ are real-valued functions such that the above norm is finite. The special case when $p = 2$ is denoted by $H^s(\mathbb{R})$. That is,

$$H^s(\mathbb{R}) := W_2^s(\mathbb{R}).$$

Again, for any p and any s, $W_p^s(\mathbb{R})$ is a Banach space. So discussing convergence of approximations in these spaces make sense as well. $H^s(\mathbb{R})$ is a Hilbert space if we give it the inner product

$$\langle f, g \rangle_{H^s(\mathbb{R})} := \langle f, g \rangle_{L_2(\mathbb{R})} + \sum_{i=1}^{s} \langle f^{(i)}, g^{(i)} \rangle_{L_2(\mathbb{R})},$$

where $f^{(i)}$ is the i^{th} weak derivative of f. In this thesis, we will only be considering the Sobolev spaces $H^s(\mathbb{R})$.

It should be noted that the restriction that $s \in \mathbb{N}$ is unnecessary. s can be any positive real number and we can still define the Sobolev space by means of the

introduction of fractional derivatives. These spaces will be very briefly considered, but we really only require that they are Banach spaces and that if $s_1 > s_2$, then

$$H^{s_1}(\mathbb{R}) \subset H^{s_2}(\mathbb{R}).$$

2.2.4 Besov Spaces

A Sobolev space has two parameters that can be adjusted to change what our function space looks like. More specifically, we can change the values of p and s. We can think of p and s as dials that can be turned in order to adjust what functions live in our space. Besov spaces give us one more parameter. The three parameters are classically denoted by p, q and s. The value of p corresponds to the L_p and l^p norms that we will be considering. s corresponds to the order of smoothness, and q allows us to fine tune the order of smoothness. In order to define Besov spaces, we will need a series of definitions.

Definition 2.2.1 *The n^{th} forward difference operator in the direction of some real number h is denoted by Δ_h^n and is defined recursively by*

$$\begin{aligned} \Delta_h^1 f(x) &= f(x+h) - f(x) \\ \Delta_h^n f(x) &= \Delta_h^1 \Delta_h^{n-1} f(x). \end{aligned}$$

Definition 2.2.2 *Let n be a positive integer, $\Omega \subseteq \mathbb{R}$ and h be a real number. Consider the subset of Ω defined by*

$$\Omega_{n,h} := \{x \in \Omega \mid x + lh \in \Omega, l = 0, \ldots, n\}.$$

For $t > 0$, the standard L_p moduli of continuity of f is defined by

$$w_n(f, t, \Omega)_p := \sup_{|h| \leq t} \|\Delta_h^n f\|_{L_p(\Omega_{n,h})}.$$

Definition 2.2.3 *Let p, q be positive integers or infinity, $s > 0$ and fix any $n > s$. The Besov semi-norm $|\cdot|_{B_q^s(L_p(\Omega))}$ is defined by*

$$|f|^q_{B_q^s(L_p(\Omega))} := \sum_{j=0}^{\infty} 2^{qsj} w_n(f, 2^{-j}, \Omega)_p^q$$

and the Besov norm is defined by

$$\|f\|^q_{B_q^s(L_p(\Omega))} := \|f\|^q_{L_p(\Omega)} + |f|^q_{B_q^s(L_p(\Omega))}.$$

A function f is said to be in the Besov space $B_q^s(L_p(\Omega))$ if the above norm is finite.

Note that the value of n was picked arbitrarily. However, it is shown in [5] that if $n_1 > s$ and $n_2 > s$, then

$$\sum_{j=0}^{\infty} 2^{qsj} w_{n_1}(f, 2^{-j}, \Omega)_p \sim \sum_{j=0}^{\infty} 2^{qsj} w_{n_2}(f, 2^{-j}, \Omega)_p.$$

That is, the two semi-norms are equivalent. The notion of equivalent semi-norms means that

$$C_1 \sum_{j=0}^{\infty} 2^{qsj} w_{n_1}(f, 2^{-j}, \Omega)_p \leq \sum_{j=0}^{\infty} 2^{qsj} w_{n_2}(f, 2^{-j}, \Omega)_p \leq C_2 \sum_{j=0}^{\infty} 2^{qsj} w_{n_1}(f, 2^{-j}, \Omega)_p,$$

where C_1 and C_2 are independent of f. Thus,

$$\|f\|_{B_q^s(L_p(\Omega))}$$

is not well-defined, but the elements of $B_q^s(L_p)$ are well-defined. Also, we have that [5]

$$B_{q_1}^s(L_p) \subseteq B_{q_2}^s(L_p), \text{ if } q_1 \leq q_2,$$

and

$$B_{q_1}^{s_1}(L_p) \subseteq B_{q_2}^{s_2}(L_p), \text{ if } s_1 \geq s_2,$$

regardless of the values of q_1 and q_2. This verifies that s is the order of smoothness and q allows for fine tuning to the order of smoothness. As well, if s is not an integer, it is also shown in [5] that

$$W_p^s(\Omega) = B_p^s(L_p(\Omega)),$$

so that

$$H^s(\Omega) = B_2^s(L_2(\Omega)).$$

Now, it turns out that in general, $\|\cdot\|_{B_q^s(L_p)}$ is not quite a norm, rather it is a quasi-norm, i.e. a map that satisfies

$$\begin{aligned} \|f+g\|_{B_q^s(L_p)} &\leq C(\|f\|_{B_q^s(L_p)} + \|g\|_{B_q^s(L_p)}) \\ \|f+g\|^\mu_{B_q^s(L_p)} &\leq \|f\|^\mu_{B_q^s(L_p)} + \|g\|^\mu_{B_q^s(L_p)}, \end{aligned}$$

for all $f, g \in B_q^s(L_p)$, where C and μ are independent of f and g. However, every Cauchy sequence in the Besov space converges inside the Besov space with respect to this quasi-norm. So, we have what is called a quasi-Banach space. The generalization to quasi-norms and quasi-Banach spaces will not pose a problem as we will in some sense be neglecting independent constants in our inequalities.

Besov spaces are very natural in wavelet analysis because the Besov norm of a function, as we shall see, is equivalent to a sum of scaled norms of sequences of wavelet coefficients. Thus the wavelet coefficients give us an indication as to the Besov norm of the function. This equivalence will become important when we need to calculate local Besov norms of functions as computing these norms from their definition is highly non-trivial.

We need one final definition regarding any normed function space.

Definition 2.2.4 *Let X be a function space, then*

$$\overline{X} := \{x \mid \textit{There is a Cauchy sequence } (x_n) \in X \textit{ such that } x_n \to x\}.$$

Note that if X is any function space, then $\overline{X}$ is a Banach space.

Chapter 3

Wavelet Setting

Some of the ideas necessary for wavelet analysis have been around for centuries. For instance, the notion of Riesz basis and Gaussian quadrature formulae. However, the late twentieth century laid out most of the ground work. Wavelet analysis, as it is understood today, began in the late 1980s. First, in 1988, Daubechies constructed the first compactly supported orthogonal wavelets with arbitrary smoothness [8]. Then, in 1989, Mallat defined the multiresolution analysis, or the MRA [14]. Since then, there have been many developments and applications arising from wavelet analysis.

One of the most fundamental developments in wavelet analysis was the construction of biorthogonal wavelets. Biorthogonal wavelets allow us to capture symmetry when we are considering approximations. As well, some functions arising from biorthogonal wavelets have a simple analytic expression. As we will be approximating the inner product of an arbitrary function against functions arising from wavelet analysis, this allows us to calculate the error between true values and approximated values.

Another more recent development is lifting. This was first introduced by Wim Sweldens and Ingrid Daubechies in [10]. It is a method to construct new wavelets out of old wavelets in a computationally simple manner. In fact, Sweldens and Daubechies proved that all wavelets can be lifted with finitely many steps from a trivial wavelet called the lazy wavelet. This plays an important role when we look at the quasi-interpolant designed by Tony Ware. It will be shown that given a quadra-

ture formula satisfying certain properties, we can lift it to form a new quadrature formula with the same properties. Furthermore, lifting the quadrature formula is done in the same manner that Sweldens and Daubechies lifted wavelets.

Some applications of wavelets are given in [15]. These include compression, denoising, derivatives and differential equations. We will look at these applications briefly in a later chapter.

Our main concern for now is to outline some definitions and look at some of the theory and properties of wavelet analysis. The objective is to use this theory to form an efficient representation of an arbitrary square-integrable function.

3.1 Multiresolution Analysis

The multiresolution analysis introduced by Mallat in 1989 [14] is one of the cornerstones of wavelet analysis . It is only one of many ways to consider wavelet analysis, but it will be a very convenient way to consider it in this thesis. I am only going to define a multiresolution briefly and state without proof some of its consequences. A detailed construction can be found in [5] or in [14].

Let $(V_j)_{j\in\mathbb{Z}}$ be a sequence of vector spaces contained in $L_2(\mathbb{R})$ satisfying

1. $V_j \subset V_{j+1}$ for all $j \in \mathbb{Z}$.

2. There is a function $\phi(\cdot) \in V_0$ satisfying

$$V_0 = \text{span}\{\phi(\cdot - k) \,\big|\, k \in \mathbb{Z}\}.$$

3. $\bigcap_{j\in\mathbb{Z}} V_j = \{0\}$ and $\bigcup_{j\in\mathbb{Z}} V_j$ is dense in $L_2(\mathbb{R})$.

4. $f(\cdot) \in V_j$ iff $f(2\,\cdot) \in V_{j+1}$.

The notion of $\bigcup_{j\in\mathbb{Z}} V_j$ being dense in $L_2(\mathbb{R})$ means that

$$\overline{\bigcup_{j\in\mathbb{Z}} V_j} = L_2(\mathbb{R}).$$

A sequence satisfying the above four conditions is said to be a multiresolution or a multiresolution analysis of $L_2(\mathbb{R})$. Multiresolutions of different spaces can be formed, however we will only be concerned with multiresolutions of $L_2(\mathbb{R})$. Once we have a multiresolution, we can define another sequence of vector spaces $(W_j)_{j\in\mathbb{Z}}$ by

$$V_j \oplus W_j = V_{j+1}$$

$$V_j \perp W_j.$$

Here, $X \oplus Y$ denotes the vector space

$$\{x + y \mid x \in X, y \in Y\},$$

and $X \perp Y$ means that for all $x \in X$ and for all $y \in Y$,

$$\langle x, y\rangle = 0.$$

So, W_j is the orthogonal complement of V_j in V_{j+1}. This gives another function ψ satisfying

$$W_0 = \text{span}\{\psi(\cdot - k) \mid k \in \mathbb{Z}\}.$$

As well, W_j satisfies

$$W_j \perp W_k \text{ for all } j \neq k$$

$$\overline{\bigoplus_{j\in\mathbb{Z}} W_j} = L_2(\mathbb{R}).$$

The functions ϕ and ψ are called the scaling function and wavelet function respectively. Note that ϕ is not uniquely defined as we have not normalized it. The convention is to let

$$\int \phi = 1.$$

Now, it is shown in [5] that if the multiresolution is tending towards $L_2(\mathbb{R})$ in the sense that we are assuming, then

$$\sum_{k\in\mathbb{Z}} \phi(x-k) = 1.$$

Thus, we have that

$$1 \in V_0.$$

Since $V_0 \perp W_0$, this gives us that

$$\begin{aligned} \langle 1, \psi(\cdot)\rangle &= 0 \\ \Rightarrow \int_{\mathbb{R}} \psi(x)dx &= 0. \end{aligned}$$

Definition 3.1.1 *The shifted and scaled version of ϕ and ψ are given by*

$$\phi_\lambda(\cdot) = \phi_{j,k}(\cdot) = 2^{j/2}\phi(2^j \cdot -k)$$

and

$$\psi_\lambda(\cdot) = \psi_{j,k}(\cdot) = 2^{j/2}\psi(2^j \cdot -k).$$

Here, we are using the standard notation

$$\lambda = (j,k).$$

We will also write $|\lambda| = j$ if $\lambda = (j,k)$ for any $k \in \mathbb{Z}$.

Now, a number is said to be a dyadic number if it has a finite binary expansion.

That is, it can be written as a finite sum of terms of the form 2^i where $i \in \mathbb{Z}$. Then, Definition 3.1.1 indicates that scaling and wavelet functions are understood on a dyadic grid. We will see how to evaluate the scaling and wavelet functions on a dyadic grid later. This is important as understanding the scaling and wavelet functions at all dyadic numbers is enough to understand the scaling and wavelet functions on all of $\mathbb{R}$. This follows if we assume that the scaling and wavelet functions are continuous and if we use the fact that the dyadic numbers are dense in $\mathbb{R}$. That is, if $\mathbb{D}$ denotes the space of dyadic numbers, then

$$\overline{\mathbb{D}} = \mathbb{R}.$$

Now, Property 4 and Property 2 of the multiresolution give us that

$$\begin{aligned} V_j &= \operatorname{span}\{\phi_{j,k} \,\big|\, k \in \mathbb{Z}\} \\ &= \operatorname{span}\{\phi_\lambda \,\big|\, |\lambda| = j\}, \end{aligned}$$

and similarly for the wavelet spaces,

$$\begin{aligned} W_j &= \operatorname{span}\{\psi_{j,k} \,|\, k \in \mathbb{Z}\} \\ &= \operatorname{span}\{\psi_\lambda \,\big|\, |\lambda| = j\}. \end{aligned}$$

In this setting we are going to assume that ϕ is orthonormal with respect to the integer translates of the scaling function. That is,

$$\langle \phi(\cdot - k), \phi(\cdot) \rangle = \delta_{k,0} = \begin{cases} 1 & k = 0 \\ 0 & k \neq 0 \end{cases}. \tag{3.1}$$

As well, we will assume that ψ is orthonormal with respect to its integer translates. That is,

$$\langle \psi(\cdot - k), \psi(\cdot) \rangle = \delta_{k,0}. \tag{3.2}$$

Now $L_2(\mathbb{R})$ is a Hilbert space so projecting on to subspaces of $L_2(\mathbb{R})$ is quite natural if we have a basis of the subspace. As we do have a basis of V_j and W_j, for any $f \in L_2(\mathbb{R})$, we can define projections on to V_j and W_j as

$$\begin{aligned} P_j &: L_2(\mathbb{R}) \to V_j \\ P_j &: f \mapsto \sum_{|\lambda|=j} \langle f, \phi_\lambda \rangle \phi_\lambda \\ Q_j &: L_2(\mathbb{R}) \to W_j \\ Q_j &: f \mapsto \sum_{|\lambda|=j} \langle f, \psi_\lambda \rangle \psi_\lambda. \end{aligned}$$

Then, by the definition of the multiresolution analysis, for any $f \in L_2(\mathbb{R})$ and for any $j \in \mathbb{Z}$

$$P_j f + \sum_{k \geq j} Q_k f \to f. \tag{3.3}$$

The above convergence in $L_2(\mathbb{R})$ is actually in the weak sense, but this detail will not be considered. In other words, we will call weak convergence simply convergence.

Rearranging Equation (3.3) and taking norms, we have that

$$\|P_j f - f\|_{L_2(\mathbb{R})} = \left\| \sum_{k \geq j} Q_k f \right\|_{L_2(\mathbb{R})},$$

so that all the error in $P_j f - f$ is contained in the wavelet spaces at levels greater than or equal to j. As V_j is tending towards $L_2(\mathbb{R})$, for any $f \in L_2(\mathbb{R})$ and any $\epsilon > 0$, there is an integer j such that

$$\|P_j f - f\|_{L_2(\mathbb{R})} < \epsilon.$$

Therefore, as we let j increase, $P_j f$ becomes a better and better approximation of f. This fact is what will later motivate the construction of the local scaling function representation.

3.2 The Refinement Equation

For most multiresolutions, the scaling and wavelet functions have no analytic expression. However, we do need a method to uniquely represent these functions. Property 1 of the multiresolution indicates that as $\phi \in V_0$, we have $\phi \in V_1$. So, we can write ϕ in terms of any basis of V_1. In particular, we can write

$$\phi(x) = \sqrt{2} \sum_{k \in \mathbb{Z}} h_k \phi(2x - k) \tag{3.4}$$

where $h_k \in \mathbb{R}$ for all $k \in \mathbb{Z}$. By a similar argument, we can write ψ as

$$\psi(x) = \sqrt{2} \sum_{k \in \mathbb{Z}} g_k \phi(2x - k) \tag{3.5}$$

where $g_k \in \mathbb{R}$ for all $k \in \mathbb{Z}$. Equations (3.4) and (3.5) are called the refinement equations or the two-scale relations of the multiresolution. We will see that the sequences (h_k) and (g_k) are extremely powerful in the sense that in a lot of instances, they are enough to solve the task at hand. To solve for the coefficients h_k and g_k, as $\phi, \psi \in V_1$, we can write

$$\begin{aligned} \phi(x) &= \sum_{k \in \mathbb{Z}} \langle \phi(\cdot), \phi_{1,k}(\cdot) \rangle \phi_{1,k}(x) \\ \psi(x) &= \sum_{k \in \mathbb{Z}} \langle \psi(\cdot), \phi_{1,k}(\cdot) \rangle \phi_{1,k}(x). \end{aligned}$$

Writing $\phi_{1,k}(x) = \sqrt{2}\phi(2x - k)$, we have that

$$\phi(x) = 2 \sum_{k \in \mathbb{Z}} \langle \phi(\cdot), \phi(2 \cdot -k) \rangle \phi(2x - k) \tag{3.6}$$

$$\psi(x) = 2 \sum_{k \in \mathbb{Z}} \langle \psi(\cdot), \phi(2 \cdot -k) \rangle \phi(2x - k). \tag{3.7}$$

Comparing (3.4) to (3.6) and (3.5) to (3.7), we have that

$$\begin{aligned} 2\langle \phi(\cdot), \phi(2\cdot - k)\rangle &= \sqrt{2}h_k \\ 2\langle \psi(\cdot), \phi(2\cdot - k)\rangle &= \sqrt{2}g_k, \end{aligned}$$

giving us that

$$\begin{aligned} h_k &= \sqrt{2}\langle \phi(\cdot), \phi(2\cdot - k)\rangle \\ g_k &= \sqrt{2}\langle \psi(\cdot), \phi(2\cdot - k)\rangle. \end{aligned}$$

The above construction relies heavily on linear independence of the set

$$\{\phi_\lambda \,\big|\, |\lambda| = j\}$$

which is not hard to see as we are simply shifting ϕ by $2^{-j}k$ where $k \in \mathbb{Z}$. Note that, if ϕ and ψ have compact support, the above construction gives us that the sequences $(h_k)_{k\in\mathbb{Z}}$ and $(g_k)_{k\in\mathbb{Z}}$ have all but finitely many zero entries. In other words, if ϕ and ψ have compact support, then (h_k) and (g_k) can be considered as finite sequences, and thus we understand ϕ and ψ while only using a finite amount of memory. In this thesis, we will always assume that ϕ and ψ have compact support. The sequences $(h_k)_{k\in\mathbb{Z}}$ and $(g_k)_{k\in\mathbb{Z}}$ are called the filters or masks of the scaling function and wavelet function respectively.

3.2.1 Filter Conditions

We have demanded several conditions on ϕ and ψ. These conditions impose conditions on the filters h_k and g_k. Here, we are going to derive some of these conditions.

First, we have the orthonormality condition

$$\langle \phi(\cdot), \phi(\cdot - l)\rangle = \delta_{l,0}.$$

Then, using the refinement equation, we have that

$$2\left\langle \sum_{k_1\in\mathbb{Z}} h_{k_1}\phi(2\cdot -k_1), \sum_{k_2\in\mathbb{Z}} h_{k_2}\phi(2\cdot -2l-k_2)\right\rangle = \delta_{l,0}.$$

Since we are assuming that the sums are finite, we can interchange the inner product and the sum. Thus, we have

$$2\sum_{k_1\in\mathbb{Z}} h_{k_1}\sum_{k_2\in\mathbb{Z}} h_{k_2}\langle\phi(2\cdot -k_1),\phi(2\cdot -2l-k_2)\rangle = \delta_{l,0}.$$

A substitution gives us

$$\begin{aligned}
\delta_{l,0} &= \sum_{k_1\in\mathbb{Z}} h_{k_1}\sum_{k_2\in\mathbb{Z}} h_{k_2}\langle\phi(\cdot -k_1),\phi(\cdot -2l-k_2)\rangle \\
&= \sum_{k_1\in\mathbb{Z}} h_{k_1}\sum_{k_2\in\mathbb{Z}} h_{k_2}\delta_{k_1,2l+k_2} \\
&= \sum_{k\in\mathbb{Z}} h_{k+2l}h_k.
\end{aligned}$$

Another condition that we required was that the scaling function integrates to 1. Again, using the refinement equation

$$\begin{aligned}
1 = \int\phi(x)dx &= \sqrt{2}\int\sum_{k\in\mathbb{Z}} h_k\phi(2x-k)dx \\
&= \sqrt{2}\sum_{k\in\mathbb{Z}} h_k\int\phi(2x-k)dx \\
&= \frac{\sqrt{2}}{2}\sum_{k\in\mathbb{Z}} h_k\int\phi(x)dx \\
&= \frac{1}{\sqrt{2}}\sum_{k\in\mathbb{Z}} h_k.
\end{aligned}$$

So,

$$\sum_{k\in\mathbb{Z}} h_k = \sqrt{2}.$$

To drive conditions for g_k, first recall that we have orthonormality in the integer translates. This condition is identical to the orthonormality with respect to the integer translates of ϕ. That is,

$$\begin{aligned}
\delta_{l,0} = \langle \psi(\cdot), \psi(\cdot - l) \rangle &= 2 \sum_{k_1 \in \mathbb{Z}} g_{k_1} \sum_{k_2 \in \mathbb{Z}} g_{k_2} \langle \phi(2 \cdot - k_1) \phi(2 \cdot - 2l - k_2) \rangle \\
&= \sum_{k_1 \in \mathbb{Z}} g_{k_1} \sum_{k_2 \in \mathbb{Z}} g_{k_2} \langle \phi(\cdot - k_1), \phi(\cdot - 2l - k_2) \rangle \\
&= \sum_{k_1 \in \mathbb{Z}} g_{k_1} \sum_{k_2 \in \mathbb{Z}} g_{k_2} \delta_{k_1, 2l + k_2} \\
&= \sum_{k \in \mathbb{Z}} g_{k+2l} g_k .
\end{aligned}$$

Next, using the fact that $V_0 \perp W_0$, we have that for all $l \in \mathbb{Z}$

$$\begin{aligned}
0 &= \langle \psi(\cdot), \phi(\cdot - l) \rangle \\
&= 2 \sum_{k_1 \in \mathbb{Z}} g_{k_1} \sum_{k_2 \in \mathbb{Z}} h_{k_2} \langle \phi(2 \cdot - k_1), \phi(2 \cdot - 2l - k_2) \rangle \\
&= \sum_{k_1 \in \mathbb{Z}} g_{k_1} \sum_{k_2 \in \mathbb{Z}} h_{k_2} \langle \phi(\cdot - k_1), \phi(\cdot - 2l - k_2) \rangle \\
&= \sum_{k_1 \in \mathbb{Z}} g_{k_1} \sum_{k_2 \in \mathbb{Z}} h_{k_2} \delta_{k_1, k_2 + 2l} \\
&= \sum_{k \in \mathbb{Z}} g_{k+2l} h_k .
\end{aligned}$$

Finally, recalling that

$$\int_{\mathbb{R}} \psi(x) = 0,$$

the refinement equation gives us that

$$\begin{aligned} 0 &= \int_{\mathbb{R}} \psi(x)dx \\ &= \int_{\mathbb{R}} \sqrt{2}\sum_{k\in\mathbb{Z}} g_k \phi(2x-k)dx \\ &= \sqrt{2}\sum_{k\in\mathbb{Z}} g_k \int_{\mathbb{R}} \phi(2x-k)dx \\ &= \sqrt{2}\sum_{k\in\mathbb{Z}} \frac{1}{2} g_k \int \phi(x)dx \\ &= \frac{\sqrt{2}}{2}\sum_{k\in\mathbb{Z}} g_k. \end{aligned}$$

So,

$$\sum_{k\in\mathbb{Z}} g_k = 0.$$

To summarize, the filters of an orthogonal wavelet system must satisfy

$$\sum_{k\in\mathbb{Z}} h_k = \sqrt{2} \tag{3.8}$$

$$\sum_{k\in\mathbb{Z}} g_k = 0 \tag{3.9}$$

$$\sum_{k\in\mathbb{Z}} h_{k+2l} h_k = \delta_{l,0} \tag{3.10}$$

$$\sum_{k\in\mathbb{Z}} g_{k+2l} g_k = \delta_{l,0} \tag{3.11}$$

$$\sum_{k\in\mathbb{Z}} g_{k+2l} h_k = 0 \tag{3.12}$$

for all $l \in \mathbb{Z}$. It is shown in [16] that if we are given two finite sequences (h_k) and (g_k) satisfying (3.8)-(3.12), and if we assume that ϕ and ψ have compact support, then the refinement equations completely determine ϕ and ψ up to a scalar multiple. So, after we normalize ϕ such that

$$\int_{\mathbb{R}} \phi = 1,$$

we have unique solutions to (3.4) and (3.5).

Example The following example is one of the most famous scaling function wavelet function pairs. It was one of the first constructions of a compactly supported orthogonal wavelet system. The below filters are shown in [16] to generate a multiresolution of $L_2(\mathbb{R})$.

$$\begin{aligned}\left(\begin{array}{cccc} h_0, & h_1, & h_2, & h_3 \end{array} \right) &= \frac{1}{4\sqrt{2}} \left(\begin{array}{cccc} 1+\sqrt{3}, & 3+\sqrt{3}, & 3-\sqrt{3}, & 1-\sqrt{3} \end{array} \right) \\ \left(\begin{array}{cccc} g_0, & g_1, & g_2, & g_3 \end{array} \right) &= \frac{1}{4\sqrt{2}} \left(\begin{array}{cccc} -1+\sqrt{3}, & 3-\sqrt{3}, & -3-\sqrt{3}, & 1+\sqrt{3} \end{array} \right).\end{aligned}$$

These are Daubechies' second scaling and wavelet functions which were first constructed in 1988. The graphs of the scaling and wavelet functions are plotted in Figure 3.1. There are no analytic expressions for the scaling and wavelet functions.

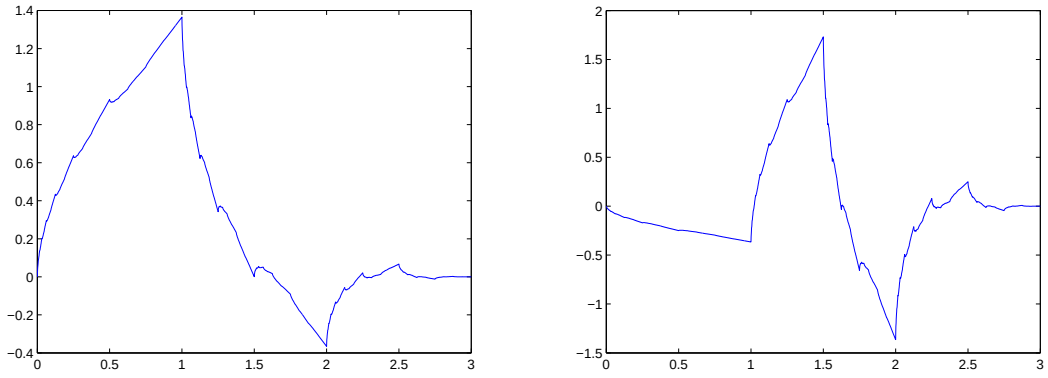

Figure 3.1: Daubechies second scaling function and wavelet function

However, using only the filters, we will see that we can calculate the function value at any dyadic number. If we then assume that the scaling function and wavelet

function are continuous, we can calculate the function values at any real number as accurately as we please.

3.3 The Biorthogonal Case

Orthogonal wavelets do have nice properties, but there is an important generalization of these functions. Biorthogonal wavelets generalize orthogonal wavelets and were first introduced in [1] by Cohen, Daubechies and Feauveau. Instead of one multiresolution analysis, we have two. That is, there are two sequences $\{V_j\}_{j\in\mathbb{Z}}$ and $\{\tilde{V}_j\}_{j\in\mathbb{Z}}$ both satisfying all the properties of a multiresolution as defined in Section 3.1. As well, we can define all the orthogonal complement spaces $W_j, \tilde{W}_j$ exactly the same as before. With these spaces come a pair of scaling functions and a pair of wavelet functions traditionally denoted by $\phi, \tilde{\phi}, \psi, \tilde{\psi}$. The function $\tilde{\phi}$ is sometimes called a dual function of ϕ, and $\tilde{\psi}$ is called a dual function of ψ. All four of them can be shifted and scaled exactly the same as in Definition 3.1.1. Instead of the orthogonality properties (3.1) and (3.2), we now have the biorthogonality properties

$$\langle \phi(\cdot - k), \tilde{\phi}(\cdot) \rangle = \delta_{k,0} \tag{3.13}$$

$$\langle \psi(\cdot - k), \tilde{\psi}(\cdot) \rangle = \delta_{k,0}. \tag{3.14}$$

If we define the projections on to the different levels of the multiresolution

$$P_j : L_2(\mathbb{R}) \to V_j$$

$$\tilde{P}_j : L_2(\mathbb{R}) \to \tilde{V}_j$$

$$Q_j : L_2(\mathbb{R}) \to W_j$$

$$\tilde{Q}_j : L_2(\mathbb{R}) \to \tilde{W}_j$$

by

$$\begin{aligned} P_j &: f \mapsto \sum_{|\lambda|=j} \langle f, \tilde{\phi}_\lambda \rangle \phi_\lambda \\ \tilde{P}_j &: f \mapsto \sum_{|\lambda|=j} \langle f, \phi_\lambda \rangle \tilde{\phi}_\lambda \\ Q_j &: f \mapsto \sum_{|\lambda|=j} \langle f, \tilde{\psi}_\lambda \rangle \psi_\lambda \\ \tilde{Q}_j &: f \mapsto \sum_{|\lambda|=j} \langle f, \psi_\lambda \rangle \tilde{\psi}_\lambda, \end{aligned}$$

then for any $f \in L_2(\mathbb{R})$ and any $j \in \mathbb{Z}$,

$$P_j f + \sum_{k \geq j} Q_k f \to f$$

and

$$\tilde{P}_j f + \sum_{k \geq j} \tilde{Q}_k f \to f.$$

Rearranging and taking norms we have that

$$\begin{aligned} \|P_j f - f\|_{L_2(\mathbb{R})} &= \left\| \sum_{k \geq j} Q_k f \right\|_{L_2(\mathbb{R})} \\ \|\tilde{P}_j f - f\|_{L_2(\mathbb{R})} &= \left\| \sum_{k \geq j} \tilde{Q}_k f \right\|_{L_2(\mathbb{R})} . \end{aligned}$$

So again, as we increase j, we get better and better approximations of f from $P_j f$ or $\tilde{P}_j f$.

We also have refinement equations, but now there are four of them. They are

$$\begin{aligned}
\phi(x) &= \sqrt{2}\sum_{k\in\mathbb{Z}} h_k \phi(2x-k) \\
\psi(x) &= \sqrt{2}\sum_{k\in\mathbb{Z}} g_k \phi(2x-k) \\
\tilde{\phi}(x) &= \sqrt{2}\sum_{k\in\mathbb{Z}} \tilde{h}_k \tilde{\phi}(2x-k) \\
\tilde{\psi}(x) &= \sqrt{2}\sum_{k\in\mathbb{Z}} \tilde{g}_k \tilde{\phi}(2x-k).
\end{aligned}$$

The filters $h_k, g_k, \tilde{h}_k, \tilde{g}_k$ contain all the information necessary to derive the scaling functions and the wavelet functions up to scalar multiplication. However, we normalize everything again such that

$$\begin{aligned}
\int \phi &= 1 \\
\int \tilde{\phi} &= 1.
\end{aligned}$$

Conditions on the four filters are necessary to impose the biorthogonality conditions. The formulations of these conditions are nearly identical to the formulation of the conditions in the orthogonal case. If $h_k, \tilde{h}_k, g_k, \tilde{g}_k$ form scaling and wavelet functions that form a biorthogonal wavelet system, then

$$\begin{aligned}
\sum_{k\in\mathbb{Z}} h_k &= \sqrt{2} \\
\sum_{k\in\mathbb{Z}} \tilde{h}_k &= \sqrt{2} \\
\sum_{k\in\mathbb{Z}} g_k &= 0 \\
\sum_{k\in\mathbb{Z}} \tilde{g}_k &= 0
\end{aligned}$$

$$\begin{aligned}
\sum_{k\in\mathbb{Z}} h_{k+2l}\tilde{h}_k &= \delta_{l,0} \\
\sum_{k\in\mathbb{Z}} \tilde{h}_{k+2l}h_k &= \delta_{l,0} \\
\sum_{k\in\mathbb{Z}} g_{k+2l}\tilde{g}_k &= \delta_{l,0} \\
\sum_{k\in\mathbb{Z}} \tilde{g}_{k+2l}g_k &= \delta_{l,0}
\end{aligned}$$

$$\begin{aligned}
\sum_{k\in\mathbb{Z}} h_{k+2l}\tilde{g}_k &= 0 \\
\sum_{k\in\mathbb{Z}} \tilde{h}_{k+2l}g_k &= 0.
\end{aligned}$$

Example The following example comes from the B-splines. It is the wavelet system having the third B-spline and its third dual function as its scaling functions. The filters are given by [9]

$$\begin{aligned}
&\left(\begin{array}{cccc} h_{-1}, & h_0, & h_1, & h_2 \end{array} \right) = \frac{1}{4\sqrt{2}} \left(\begin{array}{cccc} 1, & 3, & 3, & 1 \end{array} \right) \\
&\left(\begin{array}{cccccccc} \tilde{h}_{-3}, & \tilde{h}_{-2}, & \tilde{h}_{-1}, & \tilde{h}_0, & \tilde{h}_1, & \tilde{h}_2, & \tilde{h}_3, & \tilde{h}_4 \end{array} \right) \\
&= \frac{1}{2^5\sqrt{2}} \left(\begin{array}{cccccccc} 3, & -9, & -7, & 45, & 45, & -7, & -9, & 3 \end{array} \right) \\
&\left(\begin{array}{cccccccc} g_{-3}, & g_{-2}, & g_{-1}, & g_0, & g_1, & g_2, & g_3, & g_4 \end{array} \right) \\
&= \frac{1}{2^5\sqrt{2}} \left(\begin{array}{cccccccc} 3, & 9, & -7, & -45, & 45, & 7, & -9, & -3 \end{array} \right) \\
&\left(\begin{array}{cccc} \tilde{g}_{-1}, & \tilde{g}_0, & \tilde{g}_1, & \tilde{g}_2 \end{array} \right) = \frac{1}{4\sqrt{2}} \left(\begin{array}{cccc} -1, & 3, & -3, & 1 \end{array} \right).
\end{aligned}$$

The scaling and wavelet functions are plotted in Figure 3.2 and the dual scaling and dual wavelet functions are plotted in Figure 3.3.

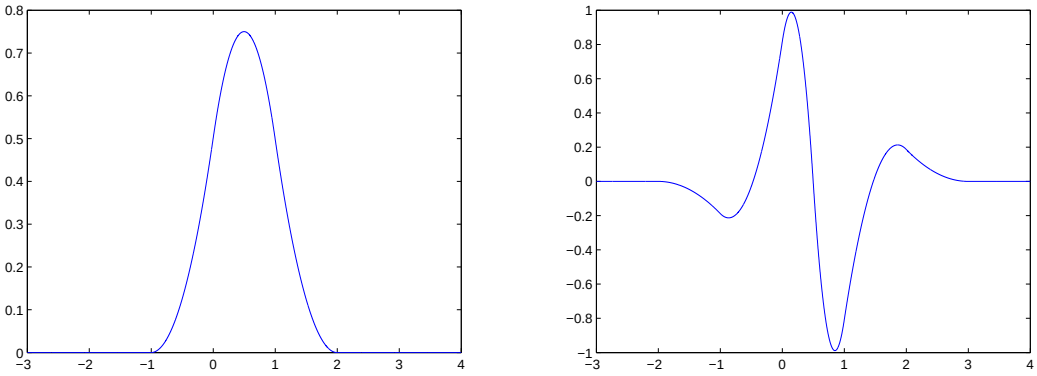

Figure 3.2: Bior3.3 scaling function ϕ and wavelet function ψ

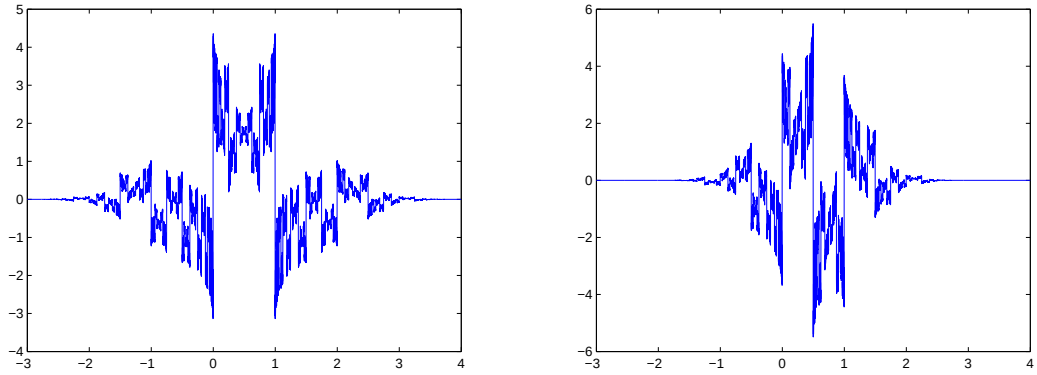

Figure 3.3: Bior3.3 dual scaling function $\tilde{\phi}$ and dual wavelet function $\tilde{\psi}$

In this case, the scaling function ϕ does have an analytic expression. It is the convolution of the characteristic function on $[0, 1)$ with itself three times and then shifted to be centered at $1/2$. That is,

$$\phi(x) = \chi_{[0,1)} * \chi_{[0,1)} * \chi_{[0,1)}(x + 1),$$

where the characteristic function χ_X is defined as

$$\chi_X(x) = \begin{cases} 1 & x \in X \\ 0 & x \notin X \end{cases}.$$

As ϕ has an analytic expression, ψ also has an analytic expression as it is a finite sum of scaled and shifted versions of ϕ. However, the dual functions are again only understood at the dyadic numbers. As seen before though, assuming continuity, this is enough to evaluate the dual functions at any real number as accurate as we please.

3.4 Point Evaluation of ϕ and ψ

We have been mentioning that we can evaluate $\phi(x)$ and $\psi(x)$ for any dyadic value of x. However, we have not yet seen how to actually do this. We are going to assume that ϕ is non-zero only on the interval $[0, n]$. Thus, $h_k \neq 0$ only when $k \geq 0$ and $k \leq n$. Using the refinement equation, if $l \in \mathbb{Z}$ and $l \geq 0$ with $l \leq n-1$, then

$$\begin{aligned} \phi(l) &= \sqrt{2} \sum_{k \in \mathbb{Z}} h_k \phi(2l - k) \\ &= \sqrt{2} \sum_{k \in \mathbb{Z}} h_{2l-k} \phi(k). \end{aligned}$$

This indicates that the vector of integer values

$$\begin{bmatrix} \phi(0) \\ \phi(1) \\ \vdots \\ \phi(n-1) \end{bmatrix}$$

is an eigenvector with eigenvalue one of

$$H_0 = \sqrt{2}\begin{bmatrix} h_0 & 0 & 0 & \cdots & 0 & 0 \\ h_2 & h_1 & h_0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & h_n & h_{n-1} \end{bmatrix}.$$

Then, it can be shown that we must normalize the vector of integer values such that the sum of its values is one. As well, by continuity we have that $\phi(n) = 0$. Next, if $l = i/2$ with i odd and l is a number between 0 and n, then

$$\begin{aligned} \phi(l) &= \sqrt{2}\sum_{k\in\mathbb{Z}} h_k \phi(2l-k) \\ &= \sqrt{2}\sum_{k\in\mathbb{Z}} h_k \phi(i-k) \\ &= \sqrt{2}\sum_{k\in\mathbb{Z}} h_{i-k} \phi(k). \end{aligned}$$

This indicates that

$$\begin{bmatrix} \phi(1/2) \\ \phi(3/2) \\ \vdots \\ \phi(n-1/2) \end{bmatrix} = H_1 \begin{bmatrix} \phi(0) \\ \phi(1) \\ \vdots \\ \phi(n-1) \end{bmatrix},$$

where

$$H_1 = \sqrt{2}\begin{bmatrix} h_1 & h_0 & 0 & 0 & \cdots & 0 & 0 \\ h_3 & h_2 & h_1 & h_0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 0 & h_n \end{bmatrix}.$$

In order to solve for the quarter integers the refinement equation gives us

$$\begin{bmatrix} \phi(1/4) \\ \phi(3/4) \\ \vdots \\ \phi(n-1/4) \end{bmatrix} = H_2 \begin{bmatrix} \phi(1/2) \\ \phi(3/2) \\ \vdots \\ \phi(n-1/2) \end{bmatrix},$$

where H_2 is the matrix

$$H_2 = \sqrt{2} \begin{bmatrix} h_0 & 0 & 0 & \cdots & 0 & 0 \\ h_1 & h_0 & 0 & \cdots & 0 & 0 \\ h_2 & h_1 & h_0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & h_n & h_{n-1} \\ 0 & 0 & 0 & \cdots & 0 & h_n \end{bmatrix}.$$

From here, it is not hard to show that for all $j \geq 3$,

$$\begin{bmatrix} \phi(\frac{1}{2^j}) \\ \phi(\frac{3}{2^j}) \\ \vdots \\ \phi(n-\frac{1}{2^j}) \end{bmatrix} = H_j \begin{bmatrix} \phi(\frac{1}{2^{j-1}}) \\ \phi(\frac{3}{2^{j-1}}) \\ \vdots \\ \phi(n-\frac{1}{2^{j-1}}) \end{bmatrix},$$

where

$$H_j = H_{j-1} \otimes \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Here, $A \otimes B$ denotes the Kronecker tensor product of the matrices A and B. Now that we have a method to evaluate the scaling function at any dyadic value, the refinement equation

$$\psi(x) = \sqrt{2} \sum_{k \in \mathbb{Z}} g_k \phi(2x-k)$$

can be used to evaluate $\psi(x)$ at any dyadic value.

The construction above is a very stable method of evaluating function values of ϕ and ψ at dyadic values. However, Matlab does have a built in function called `wavefun` that does exactly what we have described above.

3.5 Polynomial Regularity

Now that we have described orthogonal multiresolutions and a biorthogonal multiresolutions, we need to look at an important property of multiresolutions that is inherited from properties of the scaling and wavelet functions. As stated earlier, the function 1 is in V_0 for any multiresolution, and thus any constant function is in V_0. Now, it is natural to ask ourselves if there are spaces of higher degree polynomials contained in a multiresolution. It turns out that in many multiresolutions, this is the case.

Definition 3.5.1 *Suppose that $\phi, \psi, \tilde{\phi}, \tilde{\psi}$ generate a biorthogonal wavelet system. The multiresolution V_j is said to have polynomial regularity d if*

$$\int_{\mathbb{R}} x^p \psi(x) dx = 0 \text{ for all } p = 0, \ldots, d. \tag{3.15}$$

In general, we are interested in the maximum polynomial regularity. That is (3.15) holds and

$$\int_{\mathbb{R}} x^{p+1} \psi(x) dx \neq 0.$$

The next theorem ties this definition to the question that we just asked ourselves. However, we first need one more definition.

Definition 3.5.2

$$\Pi_d := \left\{ \sum_{i=0}^{d} a_i x^i \;\middle|\; a_i \in \mathbb{R},\ i = 0, \ldots, d \right\}$$

is the space of real polynomials of degree less than or equal to d.

Theorem 3.5.3 *Let $\phi, \psi, \tilde{\phi}, \tilde{\psi}$ generate a multiresolution. Then, V_j has polynomial regularity d iff for all $f \in \Pi_d$, $P_0 f = f$. In other words, V_j has polynomial regularity d iff $\Pi_d \subset V_0$.*

Proof Suppose $\int_{\mathbb{R}} x^p \psi(x) dx = 0 \quad p = 0, \ldots, d$. Let $f \in \Pi_d$. Then, we know that

$$f = \sum_{|\lambda|=0} \langle f, \tilde{\phi}_\lambda \rangle \phi_\lambda + \sum_{|\lambda| \geq 0} \langle f, \tilde{\psi}_\lambda \rangle \psi_\lambda. \tag{3.16}$$

Looking at the coefficients in the second sum, we have that

$$\begin{aligned} \langle f, \psi_{j,k} \rangle &= \int_{\mathbb{R}} f(x) 2^{j/2} \psi(2^j x - k) dx \\ &= 2^{-j/2} \int_{\mathbb{R}} f\left(\frac{x+k}{2^j} \right) \psi(x) dx. \end{aligned}$$

Since the polynomial $f(\frac{x+k}{2^j})$ has the same degree as $f(x)$, we have that

$$\langle f, \psi_{j,k} \rangle = 0.$$

Then, substituting this into (3.16), we have

$$f = \sum_{|\lambda|=0} \langle f, \tilde{\phi}_\lambda \rangle \phi_\lambda.$$

So,

$$f \in V_0 \Rightarrow P_0 f = f$$

For the converse, suppose $f = \sum_{|\lambda|=0} \langle f, \tilde{\phi}_\lambda \rangle \phi_\lambda$ for all $f \in \Pi_d$. Then, using the fact that a basis of Π_d is

$$\left\{x^0, x^1, \ldots, x^d\right\},$$

we may assume that f is a monomial of degree less than or equal to d. Say $f(x) = x^n$ where $0 \leq n \leq d$. Then,

$$\begin{aligned} & \sum_{|\lambda|\geq 0} \langle f, \tilde{\psi}_\lambda \rangle \psi_\lambda = 0 \\ \Rightarrow \quad & \langle x^n, \tilde{\psi}_\lambda \rangle = 0 \text{ where } |\lambda| = 0 \\ \Rightarrow \quad & \int_{\mathbb{R}} x^p \psi(x) = 0 \quad p = 0, \ldots, d. \end{aligned}$$

So, V_j has polynomial regularity d.

□

We proved at the start of this section that every multiresolution has polynomial regularity 0. We now have a theorem that tells us exactly when we have higher polynomial regularity. However, in order to apply this theorem, we have to evaluate $\int x^p \psi(x) dx$. We will see in Corollary 4.3.3 in the next chapter that we can evaluate this integral with no error using only the filters $h, \tilde{h}, g, \tilde{g}$. The notion of polynomial regularity will become important when we define the polynomial regularity of a quasi-interpolation scheme.

Example Table 3.1 gives the maximum polynomial regularity of different orthogonal and biorthogonal wavelet systems. First, we examine Daubechies' N^{th} multiresolution, and then the multiresolution given by the 3^{rd} B-spline with two different dual functions. Finally, we look at the polynomial regularity of the Coiflets. The

Coiflet multiresolution was designed specifically to give the multiresolution maximal polynomial regularity based on the size of the filter.

Wavelet	Multiresolution	Polynomial Regularity
db2	Primal	1
db3	Primal	2
dbN	Primal	N-1
Bior3.1	Primal	2
Bior3.1	Dual	0
Bior3.7	Dual	7
Coif1	Primal	1
Coif3	Primal	5
CoifN	Primal	2N-1

Table 3.1: Polynomial regularity of different multiresolutions

3.6 Classical Setting

The notation in this section makes some of the results that we will prove later easier to read and understand. It also gives a nice summary of the wavelet setting. Some of the results from this section are proven in [20] and others were proved earlier in this chapter.

Let $\phi, \tilde{\phi}, \psi, \tilde{\psi}$ be scaling functions and wavelet functions generating a biorthogonal wavelet system. The filter for the scaling functions will be indexed by

$$\begin{aligned} h_{l,k} &= h_{l-2k} \\ \tilde{h}_{l,k} &= \tilde{h}_{l-2k}. \end{aligned}$$

Then, the biorthogonality conditions of h indicate that

$$\sum_{k\in\mathbb{Z}} h_{l_1,k}\tilde{h}_{l_2,k} = \delta_{l_1,l_2}.$$

For the wavelet functions, we have

$$\begin{aligned} g_{l,k} &= g_{l-2k} \\ \tilde{g}_{l,k} &= \tilde{g}_{l-2k}. \end{aligned}$$

Then, the remaining biorthogonality conditions give us that

$$\sum_{k\in\mathbb{Z}} g_{l_1,k}\tilde{g}_{l_2,k} = \delta_{l_1,l_2},$$

and

$$\begin{aligned} \sum_{k\in\mathbb{Z}} h_{l_1,k}\tilde{g}_{l_2,k} &= 0 \\ \sum_{k\in\mathbb{Z}} \tilde{h}_{l_1,k}g_{l_2,k} &= 0. \end{aligned}$$

In fact, it can be shown [5] that the biorthogonality condition tells us exactly what the dual filters are. More specifically, it is shown that

$$\begin{aligned} \tilde{h}_k &= (-1)^k g_{1-k} \\ \tilde{g}_k &= -(-1)^k h_{1-k}. \end{aligned}$$

The properties of the filters in the orthogonal setting are also summarized by these equations. Since $h = \tilde{h}$ and $g = \tilde{g}$ in the orthogonal case, we have that

$$g_k = -(-1)^k h_{1-k}.$$

3.7 Approximation Properties

As discussed earlier, some function spaces arise naturally in wavelet analysis. To see this, suppose we have a biorthogonal wavelet system. Then, for any $j_0 \in \mathbb{Z}$ and any $f \in L_2(\mathbb{R})$, we have that

$$f = \sum_{|\lambda|=j_0} \langle f, \tilde{\phi}_\lambda \rangle \phi_\lambda + \sum_{|\lambda| \geq j_0} \langle f, \tilde{\psi}_\lambda \rangle \psi_\lambda.$$

For ease of notation, define

$$\begin{aligned} c_\lambda &:= \langle f, \tilde{\phi}_\lambda \rangle \\ d_\lambda &:= \langle f, \tilde{\psi}_\lambda \rangle \\ \tilde{c}_\lambda &:= \langle f, \phi_\lambda \rangle \\ \tilde{d}_\lambda &:= \langle f, \psi_\lambda \rangle. \end{aligned}$$

Then, we have that

$$\begin{aligned} f &= \sum_{|\lambda|=j_0} c_\lambda \phi_\lambda + \sum_{|\lambda| \geq j_0} d_\lambda \psi_\lambda \quad \text{and} \\ f &= \sum_{|\lambda|=j_0} \tilde{c}_\lambda \tilde{\phi}_\lambda + \sum_{|\lambda| \geq j_0} \tilde{d}_\lambda \tilde{\psi}_\lambda. \end{aligned}$$

It turns out that we can not get strict inequality in approximations, but we can get inequalities that are nearly strict.

Definition 3.7.1 *We say that a is less than or similar to b if a is less than or equal to some constant k times b, where k is independent of a and b. We write*

$$a \lesssim b.$$

If we have that $a \lesssim b$ and $b \lesssim a$, then we write

$$a \sim b.$$

IN this sense, we can say that two norms $\|\cdot\|$ *and* $\|\cdot\|^t$ *(or semi-norms) on a linear space* X *are said to be equivalent if for all* $x \in X$*, we have that*

$$\|x\| \sim \|x\|^t.$$

It is a well-known [6] that if we have two equivalent norms on a linear space, then they induce the same topology. This means that a sequence converges with respect to $\|\cdot\|$ *iff the same sequence converges with respect to* $\|\cdot\|^t$*.*

Now, it is known [5] that if $f \in H^s(\mathbb{R})$ for some $s > 1/2$ and V_0 has polynomial regularity p for some $p \geq s - 1$, then

$$|d_\lambda| \lesssim 2^{-|\lambda|s} \|f\|_{H^s(\text{supp } \psi_\lambda)}. \tag{3.17}$$

Above, we are taking the standard definition of the support of f as

$$\text{supp } f := \{x \mid f(x) \neq 0\}.$$

From (3.17), it is shown that

$$\|f - P_j f\|_{L_2(\mathbb{R})} \lesssim 2^{-js} \|f\|_{H^s(\mathbb{R})}. \tag{3.18}$$

These inequalities will not be heavily used in this thesis, but they are playing a major role in the background. More specifically, we are going to be trying to make the coefficients d_λ less than some tolerance. Equation (3.17) indicates that this is equivalent to forcing the local Sobolev norms of f to zero. As well, (3.18) says that forcing d_λ to zero for all λ will force the $L_2(\mathbb{R})$ error of $P_j f - f$ to zero.

A more important norm relation for this thesis relates the Besov norm of f to a series of scaled norms of the coefficients d_λ. It is shown in [7] that

$$\|f\|^q_{B^s_q(L_p(\mathbb{R}))} \sim \sum_j 2^{jq(s+1/2-1/p)} \left(\sum_{|\lambda|=j} |d_\lambda|^p \right)^{q/p}. \tag{3.19}$$

As we are considering the case $p = 2$, we have

$$\begin{aligned} \|f\|^q_{B^s_q(L_2(\mathbb{R}))} &\sim \sum_j 2^{jqs} \left(\sum_{|\lambda|=j} |d_\lambda|^2 \right)^{q/2} \\ &= \sum_j 2^{jqs} \left(\|(d_\lambda)_{|\lambda|=j}\|^q_{l^2(\mathbb{R})} \right). \end{aligned}$$

However, it will become important that we retain the more general equivalence (3.19).

Now that we have motivated the importance of estimating the coefficients c_λ and d_λ, we need to look at how to approximate them. This is the task of the next chapter.

Chapter 4

Quasi-Interpolants

The goal of this thesis is to use the scaling functions

$$\{\phi_\lambda \mid \lambda \in \Lambda\},$$

where Λ is some finite index set, in order to represent some function f accurately in an efficient manner. In order to construct this representation, we have to evaluate the inner product $\langle f, \tilde{\phi}_\lambda \rangle$. There are a few special cases when the inner product can be evaluated exactly, however this is generally impossible. So, a method of approximation is in store. One of the most promising methods of approximating this is using what is called a quasi-interpolation scheme. The quasi-interpolants that we will be considering are in fact Gaussian quadrature formulae. A lot of work has been done in using quasi-interpolants in the wavelet setting, and they are what we will be using for the remainder of this thesis. However, this is by far not the only method of approximating c_λ. For example, there is the following technique.

4.1 Design of Fröhlich and Schneider

The design of Jochen Fröhlich and Kai Schneider is discussed in [11]. Their method of approximating $\langle f, \tilde{\phi}_\lambda \rangle$ is based on using a cardinal Lagrange function $S_J \in V_J$. S_J

must satisfy

$$S_J\left(\frac{i}{2^J}\right) = \delta_{i,0} \tag{4.1}$$

$$V_J = \overline{\text{span}}\left\{ S_{J,k} = S_J\left(x - \frac{k}{2^J}\right) \;\middle|\; k \in \mathbb{Z} \right\}. \tag{4.2}$$

If we are in the case of an interpolating scaling function, that is

$$\phi(i) = \delta_{i,0} \text{ for all } i \in \mathbb{Z},$$

then the scaling function ϕ will satisfy (4.1) and (4.2). However, the idea is to find a function that is easier to work with than ϕ and exists for any multiresolution. Fröhlich and Schneider show that another cardinal Lagrange function can be defined by

$$\hat{S}(\omega) = \frac{(\hat{b}(\omega))^2}{\hat{b}(2\omega)},$$

where $\{b(x-k)\}_{k\in\mathbb{Z}}$ is any Riesz basis of V_0. Here, $\hat{b}$ denotes the Fourier transform of b. Once a cardinal Lagrange function $S_J \in V_J$ is fabricated, an approximation $f_J(x) := P_J f(x)$ can be written as

$$f_J(x) = \sum_{k\in\mathbb{Z}} f\left(\frac{k}{2^J}\right) S_J\left(x - \frac{k}{2^J}\right).$$

Once we have this representation, we would like to calculate the coefficients necessary to write f_J as

$$f_J = \sum_{|\lambda|=J} c_\lambda \phi_\lambda.$$

To establish the coefficient c_λ, we need

$$\begin{aligned} \langle f_J, \tilde{\phi}_{J,l} \rangle &= \sum_{k\in\mathbb{Z}} f\left(\frac{k}{2^J}\right) \left\langle S_J\left(\cdot - \frac{k}{2^J}\right), \tilde{\phi}_{J,l} \right\rangle \\ &= \sum_{k\in\mathbb{Z}} f\left(\frac{k}{2^J}\right) \langle S_J(\cdot), \tilde{\phi}_{J,l-k} \rangle. \end{aligned}$$

This indicates that

$$f_J = \sum_{l\in\mathbb{Z}} \sum_{k\in\mathbb{Z}} f\left(\frac{k}{2^J}\right) \langle S_J, \tilde{\phi}_{J,l-k}\rangle \phi_{J,l}$$

The problem with this method are that the cardinal Lagrange functions may be very difficult to compute as numerous Fourier and inverse Fourier transforms must be computed. Ideally, we would like a design that is not only accurate, but easy to compute.

4.2 Definition of a Quasi-Interpolant

Before we can define and construct a quasi-interpolant, we need some general definitions concerning linear operators and linear functionals.

Definition 4.2.1 *(a) A linear operator $L : L_p(\Omega) \to L_p(\Omega)$ is said to be local if there exists a compact set M such that*

$$f|_{x+M} = 0 \Rightarrow Lf(x) = 0.$$

for all $x \in \Omega$.

(b) A linear functional $c : L_p(\Omega) \to \mathbb{R}$ is said to be local if there exists a compact set M such that

$$f|_M = 0 \Rightarrow c(f) = 0.$$

In this case, we call M the support of c.

In Definition 4.2.1, $f|_M$ indicates the restricted function

$$f|_M(x) = \begin{cases} f(x) & x \in M \\ 0 & x \notin M \end{cases}.$$

Then, Definition 4.2.1 indicates that after operating a local linear operator on a function, evaluating this new function at any point x only requires points near x. Consequently, local linear operators are computationally easy to work with.

Definition 4.2.2 *Let Q be some linear operator that is defined on polynomials. Then, Q is said to have polynomial regularity m if all polynomials of degree less than or equal to m are fixed points of Q, but polynomials of degree $m+1$ are not fixed by Q.*

Recalling that for a multiresolution, we have that $V_j \subset L_2(\mathbb{R})$ for all $j \in \mathbb{Z}$, we are prepared to define a quasi-interpolation scheme.

Definition 4.2.3 *Let Q_j be a collection of linear operators satisfying:*
(a) $Q_j : L_p(\Omega) \to V_j$ for all $j \in \mathbb{Z}$.
(b) Q_j preserves polynomials of degree less than or equal to m for some $m \in \mathbb{N}$. That is, Q_j has polynomial regularity m.
(c) Q_j is local as in Definition 4.2.1 and is uniformly bounded in the sense that there is a constant C_Q and a compact set K such that for all $U \subseteq \Omega$,

$$\|Q_j f\|_{L_p(U)} \leq C_Q \|f\|_{L_p(U+2^{-j}K)}.$$

Then the collection of linear operators Q_j is said to form a quasi-interpolation scheme.

It is shown in [3] that the locality of Q_j assures us that for all $x \in \Omega$, $Q_j f(x)$ depends only on the values $f(y)$ where $|x-y| \lesssim 2^{-j}$. It is easy to show that the constant in this inequality is in fact the size of the support of ϕ.

Since Q_j is a linear operator on to V_j, and

$$\{\phi_\lambda \mid |\lambda| = j\}$$

spans V_j, there are unique linear functionals q_λ such that

$$Q_j f = \sum_{|\lambda|=j} q_\lambda(f)\phi_\lambda.$$

It is shown in [3] that if the linear functionals q_λ are local and uniformly bounded for all $|\lambda| = j$, then we are guaranteed that Q_j is a local uniformly bounded linear operator. So, a quasi-interpolation scheme is a sequence of uniformly bounded linear operators that are guaranteed to approximate a class of polynomials exactly, and they only require a few points to do so. As wavelet approximations are uniformly bounded and have polynomial exactness, this indicates that quasi-interpolants give accurate approximations to

$$P_j f = \sum_{|\lambda|=j} \langle f, \tilde{\phi}_\lambda \rangle \phi_\lambda,$$

and that this approximation is computationally easy to construct. In fact, it is shown in [3] that the quasi-interpolant gives the same order of local error as the nearest polynomial of degree less than or equal to m, where m is the polynomial regularity of the quasi-interpolant.

We are going to consider quasi-interpolants whose linear functionals can all be derived from the single quasi-interpolant

$$q(f) = \sum_i w_i f(x_i) \approx \langle f, \tilde{\phi} \rangle, \tag{4.3}$$

where $w_i \in \mathbb{R}$ are weights and $x_i \in \mathbb{R}$ are nodes or sample points. Here, we are letting $q(f)$ be a Gaussian quadrature formulae. Now, recall that the task of $q_\lambda(f)$

is to approximate $\langle f, \tilde{\phi}_\lambda \rangle$. However, we can approximate this with a single quasi-interpolant by noting that

$$\begin{aligned} c_\lambda = \langle f, \tilde{\phi}_\lambda \rangle &= \int_{\mathbb{R}} f(x) 2^{j/2} \tilde{\phi}(2^j x - k) dx \\ &= \int_{\mathbb{R}} 2^{-j/2} f\left(\frac{x+k}{2^j}\right) \tilde{\phi}(x) dx \\ &= \langle f_\lambda, \tilde{\phi} \rangle, \end{aligned}$$

where

$$f_\lambda(x) := 2^{-j/2} f\left(\frac{x+k}{2^j}\right).$$

Then the quasi-interpolation scheme that we are going to use to approximate $P_j f$ is

$$Q_j f = \sum_{|\lambda|=j} q_\lambda(f) \phi_\lambda = \sum_{|\lambda|=j} q(f_\lambda) \phi_\lambda. \tag{4.4}$$

Since f has compact support, this sum is guaranteed to be finite and thus, if all the weights w_i are finite, we are guaranteed that the quasi-interpolants are uniformly bounded, at least with respect to $\| \cdot \|_{L_\infty(\mathbb{R})}$. Then, the only remaining difficulty is finding appropriate weights and sample points.

Below are two constructions of quasi-interpolation schemes of the form (4.4). The first is probably the most popular and well-known quasi-interpolant. The second example is not so well-known, but it does have some nice properties.

4.3 Design of Piessens and Sweldens

The design of Robert Piessens and Wim Sweldens is described in detail in [18]. The idea is to guarantee ourselves polynomial regularity while using as few sample points as possible. Writing Equation (4.3) with $x_i = i$ gives a quadrature formulae of the

form

$$q(f) = \sum_i w_i f(i).$$

The goal is to show that we can construct q such that we form a quasi-interpolation scheme.

Assuming that f has enough derivatives, we can approximate f by a truncated Taylor series expansion about zero. That is,

$$f(x) \approx \sum_{i=0}^{N} \frac{f^{(i)}(0)}{i!} x^i$$

for some $N \in \mathbb{N}$. Then, we can approximate the coefficients c_λ by

$$c_\lambda = \langle f, \tilde{\phi} \rangle \approx \sum_{i=0}^{N} \frac{f^{(i)}(0)}{i!} \langle x^i, \tilde{\phi} \rangle.$$

This suggests that a possible quasi-interpolant would be designed if we imposed that

$$q(x^i) = \langle x^i, \tilde{\phi} \rangle \tag{4.5}$$

for all $i = 0, \ldots, N$. Then, $Q_{|\lambda|}$ will have polynomial regularity N by brute force construction. As discussed above, if the linear functionals q_λ are local and uniformly bounded, this is enough to guarantee that $Q_{|\lambda|}$ is local and uniformly bounded. First, q_λ will be uniformly bounded if the weights w_i are bounded and we will see that this is the case. Secondly, we will see that q_λ is local and that the support of q_λ is given by the support of $\tilde{\phi}_\lambda$.

Before we can prove that we do in fact have quasi-interpolation scheme, we need to show that we can evaluate the right hand side of (4.5). This defines the continuous moments which can be computed exactly using the two-scale relationship.

Definition 4.3.1 *For any scaling function ϕ with filter $(h_k)_{k\in\mathbb{Z}}$ and for any $n \in \mathbb{N}$,*

$$\begin{aligned} M_n &:= \int_{\mathbb{R}} x^n \phi(x) dx \\ m_n &:= \sum_{k\in\mathbb{Z}} k^n h_k \end{aligned}$$

are the continuous moments of ϕ and discrete moments of $(h_k)_{k\in\mathbb{Z}}$ respectively. Similar definitions hold for $\tilde{M}_n$ and $\tilde{m}_n$.

Theorem 4.3.2 *M_n can be computed with no error.*

Proof We will prove this by strong induction. The base case is true since

$$M_0 = \int \phi(x) dx = 1.$$

Suppose that we have computed the values of $M_0, \ldots, M_{n-1}$ with no error. Then,

$$\begin{aligned} M_n &= \int x^n \phi(x) dx \\ &= \sqrt{2} \int x^n \sum_k h_k \phi(2x-k) dx \\ &= \frac{\sqrt{2}}{2} \int \sum_k \left(\frac{x+k}{2}\right)^n h_k \phi(x) dx \\ &= \frac{\sqrt{2}}{2^{n+1}} \int \sum_k \sum_{l=0}^{n} \binom{n}{l} x^l k^{n-l} h_k \phi(x) dx \\ &= \frac{\sqrt{2}}{2^{n+1}} \sum_{l=0}^{n} \binom{n}{l} m_{n-l} M_l \\ &= \frac{\sqrt{2}}{2^{n+1}} \left(\sum_{l=0}^{n-1} \binom{n}{l} m_{n-l} M_l + m_0 M_n \right) \end{aligned}$$

Rearranging for M_n, we have that

$$M_n = \left(1 - \frac{\sqrt{2}}{2^{n+1}} m_0\right)^{-1} \frac{\sqrt{2}}{2^{n+1}} \sum_{l=0}^{n-1} \binom{n}{l} m_{n-l} M_l. \tag{4.6}$$

Now, assuming compact support of ϕ, m_n will be a finite sequence. Thus, m_n can be computed exactly for all $n \in \mathbb{N}$, and then the result is proved. $\square$

Theorem 4.3.2 tells us that we can evaluate the continuous moments, and the proof gives a recursive formula to evaluate M_n for all $n \in \mathbb{N}$. The Theorem also indicates that as many continuous moments as we desire can be evaluated using only the discrete moments m_n.

On a side note, recall that in the Chapter 3 we were interested in computing $\int x^p \psi(x)$ in order to calculate the polynomial regularity of a multiresolution.

Corollary 4.3.3 *The values of $\int x^p \psi(x)$ can be computed exactly.*

Proof

$$\begin{aligned}
\int x^p \psi(x)dx &= \int x^p \sqrt{2} \sum_{k\in\mathbb{Z}} g_k \phi(2x-k)dx \\
&= \sqrt{2} \sum_{k\in\mathbb{Z}} g_k \int x^p \phi(2x-k)dx \\
&= \frac{\sqrt{2}}{2} \sum_{k\in\mathbb{Z}} g_k \int \left(\frac{x+k}{2}\right)^p \phi(x)dx \\
&= \frac{\sqrt{2}}{2^{p+1}} \sum_{k\in\mathbb{Z}} g_k \int \sum_{l=0}^{p} \binom{p}{l} x^l k^{p-l} \phi(x)dx \\
&= \frac{\sqrt{2}}{2^{p+1}} \sum_{l=0}^{p} \sum_{k\in\mathbb{Z}} \binom{p}{l} k^{p-l} g_k \int x^l \phi(x)dx.
\end{aligned}$$

Defining the discrete moments of g_k by

$$n_i := \sum_{k\in\mathbb{Z}} k^i g_k,$$

we have that

$$\int x^p \psi(x) dx = \frac{\sqrt{2}}{2^{p+1}} \sum_{l=0}^{p} \binom{p}{l} n_{p-l} M_l. \tag{4.7}$$

By the previous theorem, and the fact that g_k has finitely many non-zero terms, we can evaluate (4.7) with no error. Thus, we can calculate the polynomial regularity of any multiresolution using only the filters h and g.

□

Now that we have the moments of $\tilde{\phi}$, in order to solve for the weights of the linear functional q, we have to solve the system

$$q(x^n) = \sum_i i^n w_i = \tilde{M}_n \tag{4.8}$$

where $n = 0, \ldots, N$. If supp $\tilde{\phi} = [a, b]$, we can force q to be local with support $[a, b]$ if all sample points are inside $[a, b]$. Thus we have a linear system of $N+1$ equations and $b - a + 1$ unknowns. So, to have any chance at having a unique solution, we should let $N = b - a$. That is, we have the system

$$w_a + w_{a+1} + \cdots + w_{b-1} + w_b = \tilde{M}_0$$

$$a w_a + (a+1) w_{a+1} + \cdots + (b-1) w_{b-1} + b w_b = \tilde{M}_1$$

$$\vdots \qquad\qquad \vdots$$

$$a^N w_a + (a+1)^N w_{a+1} + \cdots + (b-1)^N w_{b-1} + b^N w_b = \tilde{M}_N.$$

This is a Vandermonde system, and it is shown [12] that a Vandermonde system has a unique solution if and only if the coefficients in the second row are unique. This is the case in the above construction as our second row contains all the integers from

a to b exactly once. Thus we are guaranteed a solution to (4.5). This also indicates that the linear functional q is uniformly bounded if $\tilde{M}_n$ is finite for all n. So, the design of Sweldens and Piessens is in fact a quasi-interpolation scheme if we can show that $Q_j : L_p(\Omega) \to V_j$. However, this is clear as

$$Q_j f = \sum_{|\lambda|=j} q(f_\lambda)\phi_\lambda,$$

and $\phi_\lambda \in V_j$ for all λ with $|\lambda| = j$.

However, conditioning is a problem. In general, as the size of the support of the scaling function increases, so does the condition number of the linear system. Consider the case where the support of $\tilde{\phi}$ starts at 0. Then, in order to establish polynomial regularity $N-1$, we need to invert the $N \times N$ matrix

$$\begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 0 & 1 & 2 & \dots & N-1 \\ 0 & 1 & 4 & \dots & (N-1)^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 1 & 2^{N-1} & \dots & (N-1)^{N-1} \end{bmatrix}.$$

This is the Vandermonde matrix $(a_{i,j})_{i=1,\dots,N-1}^{j=1,\dots,N-1}$, where

$$\begin{aligned} a_{i,j} &= (j-1)^{i-1} \\ 0^0 &:= 1. \end{aligned}$$

The condition numbers of this matrix for the first few N are given in Table 4.1. Since the condition number of the system increases rapidly as N increases, we see that finding the solution becomes very difficult as N increases.

N	Condition Number
3	2.618
4	13.193
5	154.457
6	2592.886

Table 4.1: Condition numbers for certain Vandermonde systems

Example Consider computing the quasi-interpolation scheme of Piessens and Sweldens for Daubechies' second scaling function which is given by the filter

$$\left(\begin{array}{cccc} h_0, & h_1, & h_2, & h_3 \end{array} \right) = \frac{1}{4\sqrt{2}} \left(\begin{array}{cccc} 1+\sqrt{3}, & 3+\sqrt{3}, & 3-\sqrt{3}, & 1-\sqrt{3} \end{array} \right).$$

Then, in order to establish locality, the quasi-interpolant scheme is based on the linear functional

$$q(f) = \sum_{i=0}^{3} w_i f(i).$$

Using Equation (4.6), the first four continuous moments are

$$\begin{bmatrix} M_0 \\ M_1 \\ M_2 \\ M_3 \end{bmatrix} = \begin{bmatrix} 1 \\ \frac{1}{2}(3-\sqrt{3}) \\ \frac{3}{2}(2-\sqrt{3}) \\ \frac{1}{28}(189-107\sqrt{3}) \end{bmatrix}.$$

Then, the weights of our linear functionals satisfy the linear equation

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 2 & 3 \\ 0 & 1 & 4 & 9 \\ 0 & 1 & 8 & 27 \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} M_0 \\ M_1 \\ M_2 \\ M_3 \end{bmatrix}.$$

Solving this system,

$$\begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{bmatrix} = \frac{1}{56} \begin{bmatrix} 7+3\sqrt{3} \\ 21+19\sqrt{3} \\ 21-19\sqrt{3} \\ 7-3\sqrt{3} \end{bmatrix}.$$

To illustrate the accuracy of this quasi-interpolant, we know that

$$\langle \phi, \phi_{0,k} \rangle = \delta_{k,0}$$

since ϕ is orthogonal. Using the quasi-interpolant, we have that

$$q(\phi_{0,1}(x)) = q(\phi(x-1)) = \sum_{i=0}^{3} w_i \phi(i-1) \approx 0.0778.$$

This quasi-interpolant works well if the size of the masks are not too big, and if we are going to be approximating functions that can locally be approximated well by polynomials of low degree.

Now, suppose that we are operating this quasi-interpolation scheme on a function f, and it turns out that $f \in V_j$ for some $j \in \mathbb{Z}$. Unless it was the case that f was also a polynomial of degree less than or equal to the polynomial regularity of the quasi-interpolant, the above design would not pick up the fact that $f \in V_j$. However, as we are projecting on to V_j, it would be nice if our quasi-interpolant Q_j was exact for all elements of V_j. This motivates the quasi-interpolant introduced in the next section.

4.4 Design of Ware

The quasi-interpolation scheme designed by Antony Ware is the quasi-interpolant we will be using when we look at adaptive methods of approximating functions. A

more detailed construction of the quasi-interpolant is described in [20].

4.4.1 Design

Definition 4.4.1 *Fix some j and define a linear operator on to level j of the multiresolution by*

$$\hat{P}_j f := \sum_{|\lambda|=j} \theta_\lambda(f)\phi_\lambda, \tag{4.9}$$

where θ_λ are linear functionals to be determined.

Consider a Gaussian quadrature formula that is sampled at the integers and half integers. That is, the linear functional is of the form

$$\theta(f) = \sum_i w_i f(i/2),$$

and then as before,

$$\theta_\lambda(f) = \theta(f_\lambda).$$

Bearing in mind the biorthogonality condition

$$\langle \phi(\cdot), \tilde{\phi}(\cdot - k)\rangle = \delta_{0,k},$$

a good choice for the linear functional θ would be one satisfying

$$\theta(\phi(\cdot - k)) = \sum_i w_i \phi(i/2 - k) = \delta_{0,k}. \tag{4.10}$$

Then, the family of linear functionals is

$$\theta_{j,k}(f) = 2^{-j/2} \sum_i w_i f(2^{-j}(i/2 + k)). \tag{4.11}$$

Now, we need to show that solutions of (4.10) exist and that these solutions are finite. This will be discussed when we look at the construction of the linear

functional θ. As well, we need the linear operators to form a quasi-interpolation scheme as described in Definition 4.2.3.

Claim 4.4.2 *The collection of linear operators $\hat{P}_j$, as defined by (4.9) and (4.11), form a quasi-interpolation scheme.*

Proof We have three condition in Definition 4.2.3 that need to be satisfied. Condition (a) is that for each j, $\hat{P}_j$ maps to level j of the multiresolution. This is obvious for all $f \in L_2(\mathbb{R})$ since

$$\hat{P}_j f = \sum_{|\lambda|=j} \theta_\lambda(f)\phi_\lambda,$$

and $\phi_\lambda \in V_j$ for all $|\lambda| = j$.

To establish that we have polynomial regularity for some $m \in \mathbb{N}$, recall that multiresolutions always come equipped with some level of polynomial regularity. That is, for any multiresolution, there is a positive integer m such that for all $f \in \Pi_m$

$$P_0 f = f. \tag{4.12}$$

Now, pick some λ' with $|\lambda'| = 0$. Then, operating $\theta_{\lambda'}$ on both sides of (4.12) indicates that

$$\begin{aligned} \theta_{\lambda'}(f) &= \sum_{|\lambda|=0} \langle f, \tilde{\phi}_\lambda \rangle \theta_{\lambda'}(\phi_\lambda) \\ &= \langle f, \tilde{\phi}_{\lambda'} \rangle, \end{aligned}$$

where the last equality comes from the design of $\theta_{\lambda'}$. So, if $f \in V_0$,

$$\begin{aligned} \hat{P}_0 f &= \sum_{|\lambda|=0} \theta_\lambda(f)\phi_\lambda \\ &= \sum_{|\lambda|=0} \langle f, \tilde{\phi}_\lambda \rangle \phi_\lambda \\ &= P_0 f = f. \end{aligned}$$

So the collection of linear operators $\hat{P}_j$ have polynomial regularity, and their polynomial regularity is the same as the maximum polynomial regularity of the multiresolution.

To establish a uniform bound and locality of the linear operators $\hat{P}_j$, again we only require that θ_λ is local and uniformly bounded. As in the last section, it is sufficient to show that the weights are bounded and that the sum is finite. We will see this is the case when we look at the construction of θ_λ in the subsequent section.

□

Now, $\hat{P}_j$ actually has a property that is quite a bit stronger than polynomial regularity. We have that

$$\hat{P}_j f = P_j f$$

for all $f \in V_j$. The design of Piessens and Sweldens only had that

$$\hat{P}_j f = P_j f$$

for all $f \in V_j \cap \Pi_m$ for some $m \in \mathbb{N}$. This result concerning the design of Ware will be crucial when we consider adaptive representations. Moreover, this quasi-interpolant gives a tighter upper bound than the quasi-interpolant of Sweldens and Piessens. The design of Ware bounds the error of $\hat{P}_j f - f$ by the nearest element in V_j where as the design of Piessens and Sweldens bounds the error in terms of the nearest element of Π_m.

4.4.2 Construction

We now know what properties we want θ to satisfy, and we have some results that follow from these properties. However, we have not yet seen how to construct θ. Two

questions regarding the design are whether there are finite solutions and if solutions do exist, whether they are unique. To answer the first question, we will show exactly how to solve for the weights w_i so that they satisfy (4.10). At this point, it will become apparent that if a solution does exist, it is not unique.

Definition 4.4.3 *Suppose that there is a sequence* (w_i) *satisfying (4.10). Then, we call* (w_i) *a discrete dual of* ϕ.

First, note that (4.10) contains infinitely many equations, one for each integer k. However, each equation has a finite number of variables w_i as the support of the scaling function is finite. Suppose that our scaling function ϕ satisfies

$$|\text{supp } \phi| = N,$$

where N is some positive integer with $N \geq 2$. It is shown in [20] that if a discrete dual (w_i) exists, then it has $2N - 3$ non-zero entries. We will see by example that calculating the weights is equivalent to inverting a very special kind of matrix. Below, we will construct discrete duals of the third B-spline.

Example Consider the third B-spline defined by the the filter

$$\begin{pmatrix} h_{-1}, & h_0, & h_1, & h_2 \end{pmatrix} = \frac{1}{4\sqrt{2}} \begin{pmatrix} 1, & 3, & 3, & 1 \end{pmatrix}.$$

Using the refinement equation, we can find the function values of the scaling function

at the integer and half integer values. They are

$$\begin{bmatrix} \phi(-1) \\ \phi(-1/2) \\ \phi(0) \\ \phi(1/2) \\ \phi(1) \\ \phi(3/2) \end{bmatrix} = \begin{bmatrix} 0 \\ 1/8 \\ 1/2 \\ 3/4 \\ 1/2 \\ 1/8 \end{bmatrix}.$$

As $N = 3$, we are looking for three non-zero weights. Equation (4.10) indicates that three different discrete duals are solutions to

$$\begin{aligned} W \begin{bmatrix} w_{-2} \\ w_{-1} \\ w_0 \end{bmatrix} &= \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \\ W \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix} &= \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \\ W \begin{bmatrix} w_2 \\ w_3 \\ w_4 \end{bmatrix} &= \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \end{aligned}$$

where

$$W = \begin{bmatrix} \phi(1) & \phi(3/2) & 0 \\ \phi(0) & \phi(1/2) & \phi(1) \\ \phi(-1) & \phi(-1/2) & \phi(0) \end{bmatrix}.$$

So, we have that any column of the inverse matrix of W is a solution to (4.10). As

$\phi(-1) = 0$, we see that

$$W = \begin{bmatrix} \phi(1) & \phi(3/2) & 0 \\ \phi(0) & \phi(1/2) & \phi(1) \\ 0 & \phi(-1/2) & \phi(0) \end{bmatrix}.$$

By doing one row echelon step, namely a column swap, we can define the matrix

$$\tilde{W} = \begin{bmatrix} \phi(1) & 0 & \phi(3/2) \\ \phi(0) & \phi(1) & \phi(1/2) \\ 0 & \phi(0) & \phi(-1/2) \end{bmatrix}.$$

Now, define the two polynomials

$$p^e(z) = \phi(1)z + \phi(0)z = \sum_i \phi(i)z^i \tag{4.13}$$

$$p^o(z) = \phi(3/2)z + \phi(1/2) + \phi(-1/2)z^{-1} = \sum_i \phi(i+1/2)z^i. \tag{4.14}$$

Definition 4.4.4 *Let p, q be two polynomials. Say that*

$$\begin{aligned} p(z) &= \sum_{i=0}^{n_1} p_i z^i \\ q(z) &= \sum_{i=0}^{n_2} q_i z^i. \end{aligned}$$

First, $p(z)$ and $q(z)$ can be written as

$$\begin{aligned} p(z) &= p_{n_1} \prod_{i=1}^{n_1} (z - \alpha_1) \\ q(z) &= q_{n_2} \prod_{i=1}^{n_2} (z - \beta_1), \end{aligned}$$

where $\alpha_i, \beta_j \in \mathbb{C}$ for all $i = 1, \cdots, n_1$ and $j = 1, \cdots, n_2$. Then, the resultant $\rho(p, q)$ is defined to be

$$\rho(p, q) = p_{n_1}^{n_2} q_{n_2}^{n_1} \Pi_{i=1}^{n_1} \Pi_{i=1}^{n_2} (\alpha_i - \beta_j).$$

As well, the Sylvester matrix of p and q is defined to be the $(n_1 + n_2) \times (n_1 + n_2)$ matrix given by

$$S = \begin{bmatrix} p_{n_1} & p_{n_1-1} & \cdots & p_0 & 0 & 0 & \cdots & 0 \\ 0 & p_{n_1} & p_{n_1-1} & \cdots & p_0 & 0 & \cdots & 0 \\ \vdots & & \vdots & & \vdots & & \vdots & \\ 0 & 0 & \cdots & 0 & p_{n_1} & p_{n_1-1} & \cdots & p_0 \\ q_{n_2} & q_{n_2-1} & \cdots & q_0 & 0 & 0 & \cdots & 0 \\ 0 & q_{n_2} & q_{n_2-1} & \cdots & q_0 & 0 & \cdots & 0 \\ \vdots & & \vdots & & \vdots & & \vdots & \\ 0 & 0 & \cdots & 0 & q_{n_2} & q_{n_2-1} & \cdots & q_0 \end{bmatrix}.$$

It is well-known that [2] the determinant of the Sylvester matrix is equal to the resultant of the two polynomials. So, the Sylvester matrix is invertible if and only if p and q have no common zeros.

Now, note that if we multiply (4.14) by z, then the Sylvester matrix of the polynomial given in (4.13) and this new polynomial is exactly the transpose of $\tilde{W}$.

The fact that we can find the discrete dual by inverting a Sylvester matrix turns out to always be the case. That is, we have the following algorithm for constructing discrete duals.

1. Find the function values of the scaling function at integer and half integer values.

2. Construct the two Laurent polynomials.

$$\begin{aligned} p^e(z) &:= \sum_i \phi(i) z^i \\ p^o(z) &:= \sum_i \phi(i+1/2) z^i. \end{aligned}$$

3. Construct the transpose of the Sylvester matrix of p^e and p^o. Call it $\tilde{W}$.

4. If possible, invert $\tilde{W}$.

5. Choose any column $\tilde{W}^{-1}$.

6. Interlace the top half and the bottom half of the column to form the discrete dual in the correct order.

7. Decide the index of the first significant weight w_i.

In Step 6, the columns are interlaced using the fact that the top half of the columns contain the weights at the even indices in sequential order and the bottom half contain the weights at the odd indices in sequential order. This essentially is undoing the row echelon step that carried W to $\tilde{W}$. Although this algorithm is very easy to program, there are two major obstacles.

First, we have the problem of deciding which one of the columns to select as the discrete dual. It was proposed in [20] that the discrete dual with the smallest l^2 norm should be used. This will hopefully in turn help control the error in $|\theta_\lambda(f) - c_\lambda|$.

The second problem comes from Step 4 in the algorithm. This matrix could not be invertible or could be quite ill-conditioned. From experience though, if $\tilde{W}$ is invertible, for scaling functions with relatively small support, the condition number is not too large.

One last thing to note is that in Step 3, we are implicitly assuming that $\phi(x) = 0$ if $x < 0$. This is of course not the case in general. If it happens to be the case that $\phi(x) \neq 0$ for some $x < 0$, then $p^e(z)$ and $p^o(z)$ are no longer polynomials but rather Laurent polynomials. However, we can simply multiply $p^e(z)$ and $p^o(z)$ by a large enough power of z. This will create two polynomials with only positive powers. Then, after proceeding through the remainder of the algorithm, this shift can be corrected in Step 7.

In the previous example, the three discrete duals are

$$\begin{bmatrix} w_2 \\ w_3 \\ w_4 \end{bmatrix} = \begin{bmatrix} 5/2 \\ -2 \\ 1/2 \end{bmatrix} \quad \begin{bmatrix} w_0 \\ w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} -1/2 \\ 2 \\ -1/2 \end{bmatrix} \quad \begin{bmatrix} w_{-2} \\ w_{-1} \\ w_0 \end{bmatrix} = \begin{bmatrix} 1/2 \\ -2 \\ 5/2 \end{bmatrix}.$$

So, using the decision method described above in order to select a discrete dual, we would use the linear functionals

$$\theta_{j,k}(f) = \sum_{i=0}^{2} w_i f(2^{-j}(i/2 + k)),$$

where

$$(w_0, w_1, w_2) = \left(\frac{-1}{2}, 2, \frac{-1}{2}\right).$$

So, if a discrete dual does exist, we see by construction that we are guaranteed that the linear functionals are local, as the sum is finite, and uniformly bounded, as the weights are finite.

Concerning the existence of a solution remains an open problem. As we have seen, the existence of the solution is equivalent to the resultant of two polynomials being non-zero, and the resultant of two polynomials is zero if and only if they share a common root. So, we would be guaranteed a solution if we could show that $p^e(z)$

and $p^o(z)$ have no common zeros. Attempts were made in [21] to show that this is the case for all scaling functions. Although existence is not guaranteed, for all the examples we will be considering, discrete duals do exist.

4.4.3 Lifting

Lifting was first introduced by Wim Sweldens and Ingrid Daubechies in [10]. It is a method of constructing new wavelets from existing ones. It is shown in [20] that new discrete duals can be formed out of old discrete duals in a similar manner. Let

$$\begin{aligned} p^e(z) &= \sum_i \phi(i) z^i \\ p^o(z) &= \sum_i \phi(i+1/2) z^i. \end{aligned}$$

Then, satisfying (4.10) is in fact equivalent to finding Laurent polynomials $w^e(z)$ and $w^o(z)$ such that

$$w^e(1/z)p^e(z) + w^o(1/z)p^o(z) = 1. \tag{4.15}$$

It turns out that Laurent polynomials $w^e(z)$ and $w^o(z)$ can be formed to satisfy (4.15) if and only if $p^e(z)$ and $p^o(z)$ share no common zeros. This verifies are claim that the Sylvester matrix of $p^e(z)$ and $p^o(z)$ begin invertible is enough to guarantee the existence of a discrete dual. Once (4.15) is satisfied, the discrete dual can be read off as the coefficients of the Laurent polynomial

$$\begin{aligned} w(z) &= w^e(z^2) + z w^o(z^2) \text{ as} \\ w(z) &= \sum_i w_i z^i. \end{aligned}$$

Finally, define

$$\begin{aligned} p(z) &= p^e(z^2) + zp^o(z^2) \\ &= \sum_i \phi(i/2)z^i. \end{aligned}$$

Then, the lifting step is that for any Laurent polynomial s, the coefficients of $\tilde{w}(z)$ form a new discrete dual where

$$\tilde{w}(z) = w(z) + zs(z^2)p(-1/z). \tag{4.16}$$

Example Consider the third B-spline. We have that

$$\begin{aligned} p^e(z) &= 1/2 + 1/2z \\ p^o(z) &= 1/8z^{-1} + 3/4 + 1/8z. \end{aligned}$$

A discrete dual is given by the Laurent polynomial

$$w(z) = 5/2z^2 - 2z^3 + 1/2z^4.$$

Now, let $s(z) = 4z$. Then, using s to construct a new discrete dual as in (4.16), we have that

$$\tilde{w}(z) = 5/2z^2 - 2z^3 + 1/2z^4 + z4z^2(-1/8z + 1/2 - 3/4z^{-1} + 1/2z^{-2} - 1/8z^{-3}).$$

After expanding and simplifying, we get the new discrete dual

$$\tilde{w}(z) = -1/2 + 2z - 1/2z^2.$$

To see that lifting does yield a new discrete dual, we will assume that Equation (4.15) is enough to ensure that we have a discrete dual. Using (4.16) and the definitions of

$w(z)$ and $p(z)$,

$$\begin{aligned}\tilde{w}(z) &= w^e(z^2) + zw^o(z^2) + zs(z^2)(p^e(1/z^2) - z^{-1}p^o(1/z^2)) \\ &= (w^e(z^2) - s(z^2)p^o(1/z^2)) + z(w^o(z^2) + s(z^2)p^e(1/z^2)) \\ &= \tilde{w}^e(z^2) + z\tilde{w}^o(z^2).\end{aligned}$$

Thus,

$$\begin{aligned}\tilde{w}^e(z) &= w^e(z) - s(z)p^o(1/z) \\ \tilde{w}^o(z) &= w^o(z) + s(z)p^e(1/z).\end{aligned}$$

Substituting the above equations into (4.15), we get

$$\begin{aligned}&\tilde{w}^e(1/z)p^e(z) + \tilde{w}^o(1/z)p^o(z) \\ =\ &(w^e(1/z) - s(1/z)p^o(z))p^e(z) + (w^o(1/z) + s(1/z)p^e(1/z))p^o(z) \\ =\ &w^e(1/z)p^e(z) + w^o(1/z)p^o(z) = 1.\end{aligned}$$

Thus, the coefficients of $\tilde{w}(z)$ form a discrete dual.

4.4.4 Error Bounds

The discrete dual gives us a quasi-interpolation scheme with some very nice error bounds. We will examine some of them here. First, we need a method to project on to the wavelet spaces W_j in a similar manner as our projection on to V_j. Consider a multiresolution with the filters $h, \tilde{h}, g, \tilde{g}$. Working backwards, the goal is to find a linear functional satisfying the "biorthogonality" condition

$$\xi_{j,k}(\psi_{j,l}) = \delta_{k,l}.$$

To establish this linear functional, we will use the notation of the classical setting. That is,

$$\begin{aligned}
\delta_{k,l} &= \sum_{l'} g_{l',l}\tilde{g}_{l',k} \\
&= \sum_{l',k'} g_{l',l}\tilde{g}_{k',k}\delta_{k',l'} \\
&= \sum_{l',k'} g_{l',l}\tilde{g}_{k',k}\theta_{j+1,k'}(\phi_{j+1,l'}) \\
&= \sum_{l',k'} \tilde{g}_{k',k}\theta_{j+1,k'}(g_{l',l}\phi_{j+1,l'}) \\
&= \sum_{k'} \tilde{g}_{k',k}\theta_{j+1,k'}\left(\sum_{l'} g_{l'-2l}\phi_{j+1,l'}\right) \\
&= \sum_{k'} \tilde{g}_{k',k}\theta_{j+1,k'}\left(\sum_{l'} g_{l'}\phi_{j+1,l'+2l}\right) \\
&= \sum_{k'} \tilde{g}_{k',k}\theta_{j+1,k'}(\psi_{j,l}) \\
&= \xi_{j,k}(\psi_{j,l}).
\end{aligned}$$

So, a projection on to W_j can be defined as

$$\hat{Q}_j f = \sum_{|\lambda|=j} \xi_\lambda(f)\psi_\lambda$$

where

$$\xi_{j,k}(f) = \sum_{k'} \tilde{g}_{k',k}\theta_{j+1,k'}(f).$$

Now, recall that we denoted the true projection coefficients by

$$\begin{aligned}
c_\lambda &= \langle f, \tilde{\phi}_\lambda \rangle \\
d_\lambda &= \langle f, \tilde{\psi}_\lambda \rangle.
\end{aligned}$$

Then, for any $f \in L_2(\mathbb{R})$,

$$f = \sum_{|\lambda|=j} c_\lambda \phi_\lambda + \sum_{|\lambda|\geq j} d_\lambda \psi_\lambda.$$

If $|\lambda| = j$, then

$$\begin{aligned}\theta_\lambda(f) &= \sum_{|\lambda'|=j} c_{\lambda'}\theta_\lambda(\phi_{\lambda'}) + \sum_{|\lambda'|\geq j} d_{\lambda'}\theta_\lambda(\psi_{\lambda'}) \\ &= c_\lambda + \sum_{|\lambda'|\geq j} d_{\lambda'}\theta_\lambda(\psi_{\lambda'}). \end{aligned} \tag{4.17}$$

It can also shown in a similar manner as above that if $|\lambda| \geq j$, then

$$\xi_\lambda(f) = d_\lambda + \sum_{|\lambda'|>|\lambda|} d_{\lambda'}\xi_\lambda(\psi_{\lambda'}). \tag{4.18}$$

So the error between the exact coefficient and the approximated coefficient can be expanded in terms of wavelet coefficients at higher levels.

When we look at adaptive projections, we are going to be considering the l^p norm

$$\left(\sum_{\lambda\in\Upsilon_j} |d_\lambda|^p\right)^{1/p} \tag{4.19}$$

where $\Upsilon_j \subset \{\lambda \;\big|\; |\lambda| = j\}$ is some finite index set. Equation (4.18) suggests that if the wavelet coefficients of f at levels greater than j decay rapidly, then a good approximation to (4.19) is given by

$$\left(\sum_{\lambda\in\Upsilon_j} |\xi_\lambda(f)|^p\right)^{1/p}.$$

We will look at this in more detail when we design methods to produce adaptive representations of approximations.

4.5 Aliasing

Aliasing is a problem that arises when performing any kind of approximation using discrete samples. It is unavoidable, but the risk of its occurrence can be reduced in

certain circumstances. Aliasing occurs because if γ is any linear functional based on discrete samples, and $f(x_i) = g(x_i)$ for all sample points x_i of γ, then

$$\gamma(f) = \gamma(g).$$

Example Consider the function

$$f(x) = \begin{cases} 100x & x \in [0, 1/4) \\ 50 - 100x & x \in [1/4, 1/2) \\ 0 & x \notin [0, 1/2) \end{cases}$$

A plot of the function is given in Figure 4.1. Consider the scaling function given by

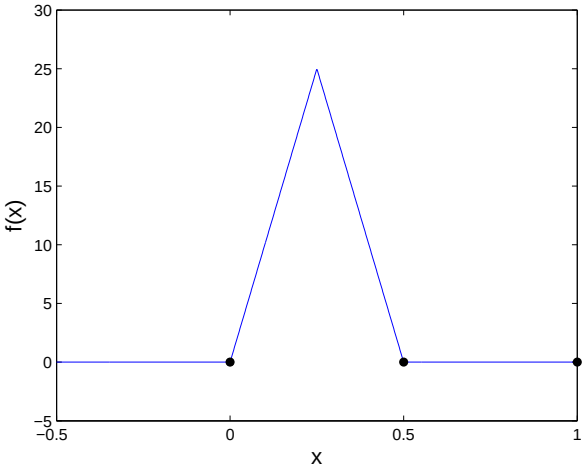

Figure 4.1: $f(x)$ with the sample points

the third B-spline. Then, the true value of c_λ, where $\lambda = (0, 0)$, is

$$\int_{\mathbb{R}} \phi(x) f(x) dx \approx 4.2318.$$

However, if we use a discrete dual,

$$\begin{aligned}\theta_{0,0}(f) &= w_0 f(0) + w_1 f(1/2) + w_2 f(1) \\ &= -1/2 \cdot 0 + 2 \cdot 0 - 1/2 \cdot 0 \\ &= 0.\end{aligned}$$

This occurred because $f(x_i) = 0$ at all the sampling points and thus

$$\theta_\lambda(f) = \theta_\lambda(0) = 0.$$

Recall that (4.17) gives us that

$$\theta_\lambda(f) - c_\lambda = \sum_{|\lambda'| \geq j} d_{\lambda'} \theta_\lambda(\psi_{\lambda'}).$$

So, in the above example, we have that

$$\sum_{|\lambda'| \geq j} d_{\lambda'} \theta_\lambda(\psi_{\lambda'}) = 4.2318.$$

That is, the true coefficient c_λ actually lives entirely in approximations higher than level 0 .

One way to reduce the risk of aliasing is to know where there may be aliasing error, and then adjust the problem appropriately. For instance, if we had known of the spike before hand in the above example, we could of refined the grid around the spike to be sure that the linear functional sampled at a non-zero point.

The phenomenon of aliasing can also occur in Fourier analysis.

Definition 4.5.1 *Let f be some function with compact support on $[0, 1]$. The Fourier*

transform and the discrete Fourier transform are given by

$$\hat{f}(k) = \int_0^1 f(x)e^{-2\pi\imath kx}dx$$
$$\hat{f}_k = \frac{1}{N}\sum_{j=0}^{N-1} f(x_j)e^{-2\pi\imath kx_j}$$

respectively.

The discrete Fourier transform can be thought of as a method of approximating the continuous Fourier transform. In fact, it is shown in [13] that for an $N \in \mathbb{N}$ we have that

$$\hat{f}_k = \sum_{j\in\mathbb{Z}} \hat{f}(jN+k) = \hat{f}(k) + \sum_{j\neq 0} \hat{f}(jN+k). \tag{4.20}$$

Example

$$f(x) = \begin{cases} \sin(10\pi x) & x \in [0,1] \\ 0 & x \notin [0,1] \end{cases}.$$

The plots of the real and imaginary parts of $\hat{f}(x)$ are plotted in Figure 4.2. However,

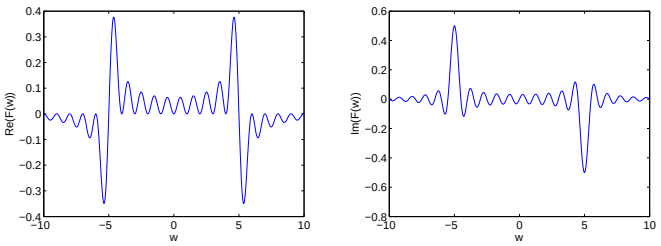

Figure 4.2: Real and Imaginary axis of the Fourier transform

if we consider the discrete Fourier transform of f with

$$x_j = \frac{j}{10} \text{ where } j = 0, \ldots, 10,$$

we get

$$\begin{aligned}\hat{f}_k &= \frac{1}{11}\sum_{j=0}^{11}\sin(\pi j)e^{-2\pi\imath kj/10} \\ &= 0.\end{aligned}$$

The plot of $f(x)$ along with the sample points are in Figure 4.3. Now, letting $N = 11$

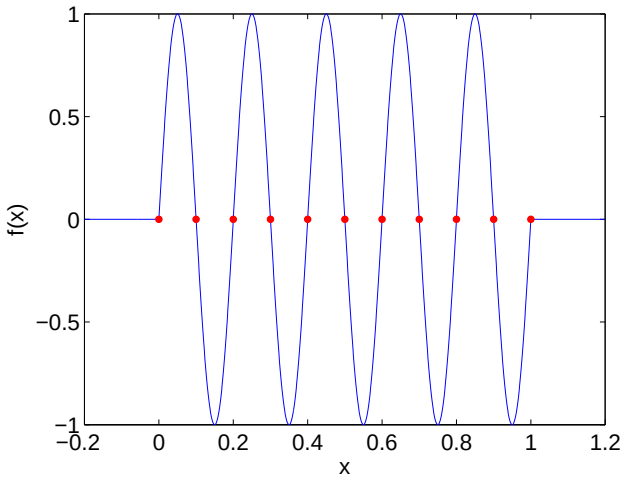

Figure 4.3: $\sin(10\pi x)$ with the sample points

in (4.20), we have that

$$\hat{f}_k = \hat{f}(k) + \sum_{j\neq 0}\hat{f}(k+11j).$$

Since $\hat{f}_k = 0$, we have that the entire Fourier transform lives in levels different then the one given by the sample $x_j = \frac{j}{10}$.

4.6 Design of Verlinden and Haegemans

This final design is not a quasi-interpolation scheme, but rather a technique to decrease the error between $Q_j f$ and $P_j f$ where Q_j is any quasi-interpolation scheme.

In [19], Verlinden and Haegemans discuss how to construct a sequence of approximations that converge rapidly to the inner product we are trying to approximate. This design requires us having a quasi-interpolation scheme before hand. Suppose we have a multiresolution defined by $\phi, \psi, \tilde{\phi}, \tilde{\psi}$ and (h_k) is the filter of ϕ. Then, ϕ is the unique compactly supported fixed point of the linear operator

$$(Tg)(x) = \sqrt{2} \sum_k h_k g(2x - k),$$

with ϕ normalized by $\int \phi = 1$. Next, suppose we have any quasi-interpolation scheme

$$q(f) = \sum_i w_i f(x_i)$$

which approximates c_λ with some polynomial regularity m. The design of Sweldens and Piessens, or the design of Ware would both be a suitable choice for q. Then, they show in [19] that the sequence

$$q\left((T^*)^n f\right)$$

converges to c_λ. Above, $(T^*)^n f$ denotes the adjoint operator of T operated n times on f. It is shown that the convergence satisfies

$$q\left((T^*)^n f\right) - \langle f, \phi \rangle \lesssim \frac{1}{2^{(m+1)n}}.$$

This technique works quite well, and they even discuss how to increase the convergence using Richardson extrapolation. The adjoint operator T^* is relatively simple, however it gets to be expensive to work with, especially when we want to compute $(T^*)^n f$ for large n.

Chapter 5

Adaptive Representations

We have now seen how we can accurately approximate c_λ and d_λ using a quasi-interpolation scheme. Thus, we can approximate

$$P_j f = \sum_{|\lambda|=j} c_\lambda \phi_\lambda \quad \text{by} \quad \sum_{|\lambda|=j} q_\lambda(f)\phi_\lambda.$$

However, this may not be a very practical method of approximation. Suppose that we are trying to approximate a function with different types of qualitative properties in different regions. In the wavelet setting, this amounts to sections of the function needing to be represented in different levels of the multiresolution. That is, some sections can be represented in V_{j_1} with some tolerance ϵ, where as another section may need to represented in V_{j_2} where $j_2 > j_1$ in order to ensure that the error in this second section is also of order ϵ. The sections requiring this finer grid could include spikes, discontinuities, regions of high oscillation or even unwanted noise in the function. We will be classifying a region as needing to be refined if the local Besov norm in this region is large.

First we need to form a partition of the domain of our function that distributes the local errors evenly across the domain. Once we have a partition of the domain, we have to discuss how to project on to an adaptive, or non-uniformly spaced grid.

5.1 Local Scaling Function Representations

The structure of the multiresolution makes wavelet analysis an appealing method of approximating functions. By moving up the multiresolution, we can in theory construct an approximation of any square-integrable function as accurately as we please. However, there are different methods to representing a real-valued function in a wavelet setting.

The most common method is to use the discrete wavelet transform introduced by Mallat in 1989 in [14]. This algorithm uses the scaling function at a base level, and wavelet functions at higher levels to project a function on to some level of the multiresolution higher than the base level.

Suppose we have some square-integrable function f. Pick a base level j_0 of the multiresolution and compute $P_{j_0} f \in V_{j_0}$. If the error between the projection and the true function is too large, then compute $Q_{j_0} f \in W_{j_0}$. Then, the projection $P_{j_0} + Q_{j_0}$ is a projection on to

$$V_{j_0} \oplus W_{j_0}.$$

Since

$$W_{j_0} \perp V_{j_0} \text{ and}$$

$$V_{j_0} \oplus W_{j_0} = V_{j_0+1},$$

$P_{j_0} + Q_{j_0}$ is the projection on to V_{j_0+1}. That is, we now have the projection

$$P_{j_0+1} f = \sum_{|\lambda|=j_0+1} \langle f, \tilde{\phi}_\lambda \rangle \phi_\lambda,$$

written as

$$\sum_{|\lambda|=j_0} \langle f, \tilde{\phi}_\lambda \rangle \phi_\lambda + \sum_{|\lambda|=j_0} \langle f, \tilde{\psi}_\lambda \rangle \psi_\lambda.$$

Continuing in this manner, we can form the projection operator

$$P_{j_0} + Q_{j_0} + Q_{j_0+1} + \cdots + Q_{J-2} + Q_{J-1}$$

which projects f on to

$$V_{j_0} \oplus \bigoplus_{j=j_0}^{J-1} W_j.$$

By the same argument as above, this is simply a projection on to V_J.

Another way to see this is by noting that $Q_j = P_{j+1} - P_j$. Then,

$$\begin{aligned} P_{j_0} f + \sum_{j=j_0}^{J-1} Q_j f &= P_{j_0} f + \sum_{j=j_0}^{J-1} (P_{j+1} f - P_j f) \\ &= P_{j_0} f + P_{j_0+1} f - P_{j_0} f + \cdots + P_J f - P_{J-1} f \\ &= P_J f. \end{aligned}$$

Now, we have two representations

$$\begin{aligned} P_J f &= \sum_{|\lambda|=J} c_\lambda \phi_\lambda \\ P_J f &= \sum_{|\lambda|=j_0} c_\lambda \phi_\lambda + \sum_{j=j_0}^{J-1} \sum_{|\lambda|=j} d_\lambda \psi_\lambda. \end{aligned}$$

In order to change between the two representations, we can use the discrete wavelet transform. This is an invertible transformation that calculates the coefficients of the scaling functions at level J using scaling function coefficients at the base level, and wavelet coefficients at levels between j_0 and $J-1$. The transformation relies on the refinement equation along with the orthogonality of the integer translates. A more

detailed outline of the transformation can be found in [4].

At this point, the classical method of making the approximation adaptive is to let insignificant values d_λ be 0. That is, we would pick some tolerance $\epsilon > 0$ and if $d_\lambda > \epsilon$, we would leave d_λ unchanged and if $d_\lambda \leq \epsilon$, we would let $d_\lambda = 0$.

This kind of approximation is built on a multi-layer grid. That is, if we wanted to evaluate the approximation at a single point x, we would have to evaluate

$$P_{j_0} f(x), Q_{j_0} f(x), \ldots, Q_{J-1} f(x).$$

An alternative method is to partition the domain and project each region on to an appropriate level of the multiresolution. This is a nearly single-layer method of approximation that does not use the wavelet functions in the representation. It is only nearly single-layer as near the boundary points of the partition, there may be interaction with scaling functions at several different levels.

Consider the general algorithm

1. Project f on to some base level j_0 of the multiresolution to get an approximation $P_{j_0} f$. Let $P = P_{j_0}$.

2. Find all regions where there is large error in $Pf - f$ and refine the grid to one level higher in these regions.

3. Project the refined area on to one level higher of the multiresolution.

4. Bring these two projections together appropriately to build a new projection on to a direct sum of elements of the multiresolution. Call this new projection P.

5. Goto Step 2.

In order to make this algorithm terminate, we build two stops into the algorithm. One is if the error in the representation is less than some predetermined tolerance ϵ. The second is if we have performed some predetermined maximum number of iterations denoted by `maxiter`. The result, Pf, is some representation of f using only the scaling function on an adaptive grid. That is, we have an adaptive local scaling function representation.

In order to simplify this algorithm, for all $\epsilon > 0$ we would like to be able to determine a partition Λ_ϵ of the domain Ω before constructing P such that we are guaranteed that

$$\|Pf - f\|_{L_2(\Omega)} \lesssim \epsilon$$

where P_Λ is some projection. Then, we do not have to repetitively look for subsets of Ω that contain large error in $Pf - f$. Creating this partition will be the role of the prediction.

Now, since $V_j \subset V_{j+1}$, we have that there is a $J \in \mathbb{Z}$ such that $Pf \in V_J$. However, we have constructed P such that restrictions of Pf are in subspaces of V_J. That is, there are subsets I of Ω such that

$$Pf\Big|_I \in V_j\Big|_I$$

for some $j < J$. The fact that there are restrictions of Pf that are in a subspace of V_J is what makes the algorithm more practical than if we were simply to compute

$$P_J f = \sum_{|\lambda|=J} \langle f, \tilde{\phi}_\lambda \rangle \phi_\lambda.$$

The approximation described above is very similar to the classical approximation of f expressed as

$$P_{j_0} f + \sum_{j=j_0}^{J-1} Q_j f.$$

With help from the discrete wavelet transform, both algorithms result in an approximation Pf that can be written as a sum of scalar multiples of scaling functions at level J. As the next algorithm constructs a local scaling function representation, it is the representation we will be considering. Then, if desired, the discrete wavelet transform could be implemented in order to write Pf as an element of

$$V_{j_0} \oplus \bigoplus_{j=j_0}^{J-1} W_j.$$

5.2 Prediction

Let f be some compactly supported function with support Ω. The role of the prediction is to determine a partition of Ω such that

$$\|P_\Lambda f - f\|_{L_2(\Omega)} \lesssim \epsilon,$$

where P_Λ is some projection operator to be determined. The first step to predicting the partition is finding a method to determine where error may exist in a local scaling function representation. Following [7], we make use of local error functions.

5.2.1 Local Error Functions

A local error function E maps subsets of Ω to the positive real numbers. Error functions must satisfy

$$\square_1 \subseteq \square_2 \Rightarrow E(\square_1) \leq E(\square_2)$$

and

$$E(\square_1) \to 0 \text{ as } \operatorname{diam}(\square_1) \to 0$$

whenever $\square_1, \square_2 \subseteq \Omega$ are subsets of Ω. We want an error function that will capture the qualitative properties of our function in subsets of Ω. That is, the error function E maps a subset $\square_\lambda$ of Ω to a real number that corresponds to the error in $\square_\lambda$ of a local scaling function representation. Here, $\square_\lambda$ denotes the interval

$$\square_\lambda = \square_{(j,k)} = 2^{-j}(\square + k)$$

where $\square$ is the support of the mother scaling function ϕ.

Then, given $\epsilon > 0$, the role of the error function is to form a partition consisting of subsets $\square_\lambda$, indexed by $\lambda \in \Lambda_\epsilon$, such that

$$\begin{aligned} \Omega &= \bigcup_{\lambda \in \Lambda_\epsilon} \square_\lambda, \\ |\square_\lambda \cap \square_\mu| &= 0 \text{ for all } \lambda, \mu \in \Lambda_\epsilon,\ \lambda \neq \mu, \text{ and} \\ \epsilon_p(\Lambda_\epsilon) &:= \left(\sum_{\lambda \in \Lambda_\epsilon} E(\square_\lambda)^p \right)^{1/p} \lesssim \epsilon. \end{aligned} \tag{5.1}$$

Note that satisfying (5.1) means that the l^p norm of the sequence $E(\square_\lambda)$, where λ runs through Λ_ϵ, is of order less than ϵ.

Now, suppose that $\tau > (s + 1/p)^{-1}$ for some $1 \leq p \leq \infty$, $s > 1/2$ and $f \in B^s_q(L_\tau(\Omega))$ for some $q \leq \tau$. The assumption that $s > 1/2$ is standard to assume when performing point evaluation and this is crucial as we are working with discrete samples. Then, it is shown in [7] that

$$E(f, \square_\lambda) := 2^{-|\lambda|(s+1/p-1/\tau)} \|f\|_{B^s_q(L_\tau(\square^*_\lambda))} \tag{5.2}$$

defines an error function. It is also shown that, for all $\epsilon > 0$, there is a partition Λ_ϵ such that $E(f, \square_\lambda)$ satisfies (5.1). Above, $\square^*_\lambda$ is the interval $\square_\lambda$ with some additional

points near the boundary of $\square_\lambda$. We will need these extra points because when we project a function on to $\square_\lambda$, the support of the approximation will generally extend past $\square_\lambda$. This overspill of $\square_\lambda$ will introduce error that needs to be accounted for. In fact, it is shown in [3] that $\square_\lambda^*$ can be any interval satisfying

$$\operatorname{diam}(\square_\lambda^*) \lesssim 2^{-|\lambda|}$$

$$\square_\lambda \subset \square_\lambda^*.$$

What we want now is a local scaling function representation whose L_p error in Ω has the same order as the l^p norm of the sequence

$$(E(\square_\lambda))_{\lambda \in \Lambda_\epsilon}.$$

Also, we want to do this in such a way that that the number of points in the partition is not too large.

To prove that such a representation exists, we can use a result from [7] that states that, with the above choice of error function, and for any $N \in \mathbb{N}$, there is a partition set Λ of Ω such that $|\Lambda| = N$ and

$$\epsilon_p(\Lambda) \leq CN^{-s}\|f\|_{B_q^s(L_\tau(\Omega))}, \tag{5.3}$$

where C is independent of N and f. Above, $|\Lambda|$ denotes the counting measure of Λ. This means that, given any tolerance $\epsilon > 0$, we can find a finite partition of the domain such that

$$\epsilon_p(\Lambda) \lesssim \epsilon.$$

As well, (5.3) indicates that to decrease the error we should increase N. This verifies our intuition that increasing the number of points in the partition decreases the size

of $\epsilon_p(\Lambda)$.

Definition 5.2.1 *For any partition set Λ, and any map P_Λ which maps a subspace L_p on to some level of a multiresolution, P_Λ is said to be E-admissible if*

$$\|P_\Lambda f - f\|_{L_p(\square_\lambda)} \lesssim E(\square_\lambda), \text{ for all } \lambda \in \Lambda.$$

Now, if we construct our partition such that $\epsilon_p(\Lambda) \lesssim \epsilon$, and if P_Λ is an E-admissible operator, we have that

$$\begin{aligned} \|P_\Lambda f - f\|^p_{L_p(\Omega)} &= \sum_{\lambda\in\Lambda} \|P_\Lambda f - f\|^p_{L_p(\square_\lambda)} \\ &\lesssim \sum_{\lambda\in\Lambda} E(\square_\lambda)^p. \end{aligned}$$

Then, taking each side to the power of $1/p$, we get

$$\|P_\Lambda f - f\|_{L_p(\Omega)} \lesssim \left(\sum_{\lambda\in\Lambda} E(\square_\lambda)^p\right)^{1/p} \lesssim \epsilon.$$

This indicates that an E-admissible operator, along with a properly partitioned grid, gives a good adaptive local scaling function representation in the sense that we can make it as accurate as we desire. However, we would like P_Λ to possess similar conditions to those of the non-adaptive projections. More specifically, we would like P_Λ to be local in some sense and preserve a class of polynomials.

Definition 5.2.2 *Let P_Λ be an adaptive projection operator. Then, P_Λ is said to be a quasi-interpolant if*

(a) P_Λ has polynomial regularity m where $m \in \mathbb{N}$ in the sense that it preserves all polynomials of degree less than m and

(b) P_Λ is local with respect to Λ. That is, if U is contained in

$$\bigcup_{\substack{\lambda\in\Lambda\\ |\lambda|=j}} \square_\lambda,$$

for some j, then

$$P_\Lambda f|_U = P_\Lambda(\chi_{U+2^{-j}K} f)|_U,$$

where K is some compact set. We will see later that we can make P_Λ local with K being the compact set making P_j local.

The preservation of polynomials is important as it gives us similar error bounds of

$$\|P_\Lambda f - f\|_{L_2(\Omega)}$$

as in the non-adaptive case. That is, local errors are bounded in terms of the nearest local polynomial. The locality will make P_Λ computationally easy to work with in the same sense as the non-adaptive case. Thus, our goal is to construct an operator P_Λ which is an E-admissible adaptive quasi-interpolation operator. This will be the role of the iteration.

5.2.2 Oracles

We have discussed how we want the errors in our partition to be distributed. However, we have not formally shown how to construct a partition using the error function. That is, we need to examine how to form Λ such that it satisfies (5.3). This is the job of the oracle. In general, it performs the following steps.

1. Choose an initial partition of the domain.

2. Search the partition for regions that may contain large local error using the local error function.

3. Refine the partition in these regions.

4. Goto Step 2.

Recall that our error function measures the error in terms of scaled local Besov norms. Thus, we are creating a partition that controls the local Besov norms. Since we are in the wavelet setting, it is convenient to partition the domain on a dyadic grid. So, the oracle that we will be considering performs Step 3 by adding a new point to the partition exactly in the middle of the region under consideration. Then, if our initial partition only contains dyadic points, we are guaranteed that our final partition will only contain dyadic points.

Step 2 is where the local error function comes into play. If we are considering some region $\square_\lambda$, then we can determine if $E(f, \square_\lambda) < \epsilon$ or if $E(f, \square_\lambda) > \epsilon$. By design of the multiresolution, we can run the oracle until we have a partition such that every region has error less than ϵ. We have also introduced the additional parameter that we called `maxiter`. As the oracle can be computationally expensive, it makes sure that the oracle does no more than `maxiter` iterations.

Recall that we actually required that

$$\|P_\Lambda f - f\|_{L_p(\Omega)} \lesssim \epsilon_p(\Lambda) \lesssim \epsilon \tag{5.4}$$

where $\epsilon_p(\Lambda)$ is defined in (5.1). However, we have imposed that

$$E(\square_\lambda) < \epsilon \text{ for all } \lambda \in \Lambda.$$

This would be sufficient to satisfy (5.4) if $p = \infty$, but for $p \neq \infty$, a more complicated oracle would have to be designed. However, we do have the one way implication

$$\epsilon_P(\Lambda) \lesssim \epsilon \Rightarrow E(\square_\lambda) \lesssim \epsilon \text{ for all } \lambda \in \Lambda,$$

and we are guaranteed that a finite partition satisfying (5.4) does exist, and thus the difference between the statements

$$\left(\sum_{\lambda \in \Lambda} E(\square_\lambda)^p \right)^{1/p} \lesssim \epsilon$$

and

$$E(\square_\lambda) < \epsilon \text{ for all } \lambda \in \Lambda$$

shall be ignored. If this does pose a problem, we can simply restrict ourselves to functions whose domain can be partitioned such that

$$\begin{aligned} E(\square_\lambda) &\lesssim \epsilon \text{ for all } \lambda \in \Lambda \text{ and} \\ |\Lambda| &\leq M \end{aligned}$$

for some $M \in \mathbb{N}$. Then, we would have the two way implication

$$\epsilon_P(\Lambda) \lesssim \epsilon \Leftrightarrow E(\square_\lambda) \lesssim \epsilon \text{ for all } \lambda \in \Lambda,$$

However, it should be noted that it is possible to implement (5.4). We could continuously refine the partition in the interval containing the largest value in the local error function and then check if (5.4) is satisfied. However, this is expensive and thus we will not be implementing it. Now, in order to calculate an accurate approximation to $E(f, \square_\lambda)$, we need some norm equivalences.

5.2.3 Norm Equivalences

As Besov norms are computationally difficult to evaluate, we need to take advantage of some norm equivalences discussed earlier. Recall that we have the equivalence from [7]

$$\|f\|^q_{B^s_q(L_p)} \sim \sum_j 2^{jq(s+1/2-1/p)} \left(\sum_{|\lambda|=j} |d_\lambda|^p \right)^{q/p} \tag{5.5}$$

where $q \leq p$ and we will assume that $s > 1/2$. In order to look at this norm equivalence at a local level, we need to introduce another index set.

Definition 5.2.3 *Let $\square^*_\lambda$ be the extended support $\square_\lambda$ of ϕ_λ. Then the set of significant wavelet coefficients is*

$$\zeta^j_\lambda := \left\{ \mu \middle| |\mu| = j \text{ and } |w_\mu \cap \square^*_\lambda| \neq 0 \right\}$$

where $j \in \mathbb{Z}$.

Then, restricting ourselves to $\square^*_\lambda$, (5.5) indicates that

$$\|f\|^q_{B^s_q(L_p(\square^*_\lambda))} \lesssim \sum_j 2^{jq(s+1/2-1/p)} \left(\sum_{\mu \in \zeta^j_\lambda} |d_\mu|^p \right)^{q/p}. \tag{5.6}$$

Now, we are interested in bounds of the error function $E(f, \square_\lambda)$ where

$$E(f, \square_\lambda)^q = 2^{-|\lambda| q(s+1/p-1/\tau)} \|f\|^q_{B^s_q(L_p(\square^*_\lambda))}.$$

Considering the case $\tau = p$, if $s > 0$ (which we have assumed), we have that

$$\begin{aligned} E(f, \square_\lambda)^q &\lesssim 2^{-|\lambda|qs} \sum_j 2^{jq(s+1/2-1/p)} \left(\sum_{\mu \in \zeta_\lambda^j} |d_\mu|^p \right)^{q/p} \\ &= 2^{-|\lambda|qs} 2^{|\lambda|q(s+1/2-1/p)} \left(\sum_{\mu \in \zeta_\lambda^{|\lambda|}} |d_\mu|^p \right)^{q/p} + \sum_{j \neq |\lambda|} 2^{-|\lambda|qs} 2^{jq(s+1/2-1/p)} \left(\sum_{\mu \in \zeta_\lambda^j} |d_\mu|^p \right)^{q/p} \\ &= 2^{|\lambda|q(1/2-1/p)} \left(\sum_{\mu \in \zeta_\lambda^{|\lambda|}} |d_\mu|^p \right)^{q/p} + \sum_{j \neq |\lambda|} 2^{qs(j-|\lambda|)} 2^{jq(1/2-1/p)} \left(\sum_{\mu \in \zeta_\lambda^j} |d_\mu|^p \right)^{q/p}. \end{aligned}$$

Now, if we assume that the error is almost all local to level $|\lambda|$ in the sense that

$$\sum_{j \neq |\lambda|} 2^{qs(j-|\lambda|)} 2^{jq(1/2-1/p)} \left(\sum_{\mu \in \zeta_\lambda^j} |d_\mu|^p \right)^{q/p} \lesssim 2^{|\lambda|q(1/2-1/p)} \left(\sum_{\mu \in \zeta_\lambda^{|\lambda|}} |d_\mu|^p \right)^{q/p},$$

then

$$E(f, \square_\lambda)^q \lesssim 2^{|\lambda|q(1/2-1/p)} \left(\sum_{\mu \in \zeta_\lambda^{|\lambda|}} |d_\mu|^p \right)^{q/p}.$$

Thus, a bound for our error function is

$$E(f, \square_\lambda) \lesssim 2^{|\lambda|(1/2-1/p)} \left(\sum_{\mu \in \zeta_\lambda^{|\lambda|}} |d_\mu|^p \right)^{1/p}. \tag{5.7}$$

The assumption that the error is local to the current level will follow if our function behaves well in the sense that the wavelet coefficients decay appropriately and thus the local Besov norms behave appropriately. This may not always be the case as in the scenario we looked at earlier when we considered aliasing error. If we did want to decrease our risk of aliasing error, we could use a bound such as

$$E(f, \square_\lambda)^q \lesssim \sum_{j \in \Gamma} 2^{jq(s+1/2-1/p)} \left(\sum_{\mu \in \zeta_\lambda^j} |d_\mu|^p \right)^{q/p},$$

where Γ is some appropriate finite subset of $\mathbb{Z}$. This is the idea of a halo as we are considering wavelet coefficients at levels near the current level in order to be sure that aliasing is avoided. However, this would make running the oracle a lot more expensive. Thus, we will assume that aliasing will not introduce too much error and so an appropriate bound for $E(f, \square_\lambda)$ is given by (5.7). As well, the upper bound in (5.7) will be called $E(f, \square_\lambda)$. That is, our new error function is defined as

$$E(f, \square_\lambda) := 2^{|\lambda|(1/2-1/p)} \left(\sum_{\mu \in \zeta_\lambda^{|\lambda|}} |d_\mu|^p \right)^{1/p}.$$

To see that E is an error function, we first have to assume that $p = 1$ or $p = 2$ to assure ourselves that $(1/2 - 1/p) \leq 0$. This does not pose a problem as we are assuming that $p = 2$ throughout this thesis. Now, if $\square_{\lambda_1} \subset \square_{\lambda_2}$, then $|\lambda_1| > |\lambda_2|$. As well, the significant wavelet coefficients of $\square_{\lambda_1}$ will be smaller (about half the size) of the significant wavelet coefficients of $\square_{\lambda_2}$. Thus,

$$\begin{aligned} E(f, \square_{\lambda_1}) &= 2^{|\lambda_1|}(1/2 - 1/p) \left(\sum_{\mu \in \zeta_{\lambda_1}^{|\lambda_1|}} |d_\mu|^p \right)^{1/p} \\ &\leq 2^{|\lambda_2|}(1/2 - 1/p) \left(\sum_{\mu \in \zeta_{\lambda_2}^{|\lambda_2|}} |d_\mu|^p \right)^{1/p} \\ &= E(f, \square_{\lambda_2}). \end{aligned}$$

This is still not quite the error function that we are going to be using. Since we can not compute d_μ exactly, we will approximate it by $\xi_\mu(f)$. That is, we will use the error function

$$E^*(f, \square_\lambda) := 2^{|\lambda|(1/2-1/p)} \left(\sum_{\mu \in \zeta_\lambda^{|\lambda|}} |\xi_\mu(f)|^p \right)^{1/p}.$$

It will not always be the case that E^* is an error function as it depends on discrete samples of the function. However, $\xi_\mu(f)$ is the only approximation that we have for d_μ. As well, based on numerical experiments, using E^* has always picked up areas of large local Besov norm if the aliasing error is small. So, we will treat E^* as an error function. That is, we will use E^* as the essential function in the oracle.

5.2.4 Oracle Error

Now, we would like to justify our substitution of $E(f, \square_\lambda)$ with $E^*(f, \square_\lambda)$. Recall that we have the aliasing condition (4.18)

$$\xi_\lambda(f) = d_\lambda + \sum_{|\lambda'|>|\lambda|} d_{\lambda'}\xi_\lambda(\psi_{\lambda'}).$$

So,

$$\begin{aligned} |\xi_\lambda(f) - d_\lambda| &= \left| \sum_{|\lambda'|>|\lambda|} d_{\lambda'}\xi_\lambda(\psi_{\lambda'}) \right| \\ &\leq \sum_{j>|\lambda|} \left| \sum_{|\lambda'|=j} d_{\lambda'}\xi_\lambda(\psi_{\lambda'}) \right| \\ &\leq \sum_{j>|\lambda|} \left(\sum_{|\lambda'|=j} |d_{\lambda'}|^p \right)^{1/p} \left(\sum_{|\lambda'|=j} |\xi_\lambda(\psi_{\lambda'})|^{p'} \right)^{1/p'} \end{aligned} \tag{5.8}$$

where p and p' satisfy $1/p + 1/p' = 1$. The last inequality is Holder's inequality. Now, the term $\xi_\lambda(\psi_{\lambda'})$ involves a fixed number of non-zero entries at any level $|\lambda'|$ of the multiresolution since it is a discrete sample of a compactly supported function. It is shown in [20] that this number is bounded independently of $|\lambda|$ and $|\lambda'|$, and the entry itself is also independent of $|\lambda|$ and $|\lambda'|$ apart from a scaling factor of $2^{\frac{|\lambda'|-|\lambda|}{2}}$. Also, we can consider both sums as finite as there are only finitely many λ' such that

$\xi_\lambda(\psi_{\lambda'}) \neq 0$. Thus, the above argument and (5.8) gives us

$$|\xi_\lambda(f) - d_\lambda| \lesssim \sum_{j>|\lambda|} 2^{\frac{j-|\lambda|}{2}} \left(\sum_{\substack{|\mu|=j \\ \xi_\lambda(\psi_\mu)\neq 0}} |d_\mu|^p \right)^{1/p}. \tag{5.9}$$

Then, it turns out that

$$\left\{ \mu \;\middle|\; |\mu| = j \text{ and } \xi_\lambda(\psi_\mu) \neq 0 \right\} \subseteq \zeta_\lambda^j,$$

and thus we can write (5.9) as

$$|\xi_\lambda(f) - d_\lambda| \lesssim \sum_{j>|\lambda|} 2^{\frac{j-|\lambda|}{2}} \left(\sum_{\mu\in\zeta_\lambda^j} |d_\mu|^p \right)^{1/p}. \tag{5.10}$$

One assumption we have to make is that the upper bound we have for the local Besov norms in (5.6) is in fact a norm equivalence. That is

$$\|f\|^q_{B^s_q(L_p(\square^*_\lambda))} \sim \sum_j 2^{jq(s+1/2-1/p)} \left(\sum_{\mu\in\zeta_\lambda^j} |d_\mu|^p \right)^{q/p}. \tag{5.11}$$

This assumption is valid for well-behaved functions as the only time it may not hold is if the wavelet coefficients do not decay fast enough near the boundary of $\square^*_\lambda$. However, we have at some level assumed that this is not the case by demanding that

$$f \in B^s_q(L_p(\square^*_\lambda)).$$

Then, with the assumption (5.11), we can deduce that

$$\sum_{j>|\lambda|} 2^{jq(s+1/2-1/p)} \left(\sum_{\mu\in\zeta_\lambda^j} |d_\mu|^p \right)^{q/p} \lesssim \|f\|^q_{B^s_q(L_p(\square^*_\lambda))}. \tag{5.12}$$

Now, recall that we are working in the case $p = 2$, however $p = 1$ would work in the construction below. First,

$$\begin{aligned} |E(f, \square_\lambda) - E^*(f, \square_\lambda)| &= \left| 2^{|\lambda|(1/2-1/p)} \left(\|(d_\mu)_{\mu \in \zeta_\lambda^{|\lambda|}}\|_{l^p} - \|(\xi_\mu(f))_{\mu \in \zeta_\lambda^{|\lambda|}}\|_{l^p} \right) \right| \\ &\leq 2^{|\lambda|(1/2-1/p)} \|(d_\mu - \xi_\mu(f))_{\mu \in \zeta_\lambda^{|\lambda|}}\|_{l^p}. \end{aligned} \tag{5.13}$$

by the triangle inequality. Then, rewriting (5.13) as a sum and substituting (5.10) into the upper bound, we have

$$\begin{aligned} &= 2^{|\lambda|(1/2-1/p)} \left(\sum_{\mu \in \zeta_\lambda^{|\lambda|}} |d_\mu - \xi_\mu(f)|^p \right)^{1/p} \\ &\lesssim 2^{|\lambda|(1/2-1/p)} \left(\sum_{\mu \in \zeta_\lambda^{|\lambda|}} \left(\sum_{j > |\mu|} 2^{\frac{j-|\mu|}{2}} \left(\sum_{\gamma \in \zeta_\mu^j} |d_\gamma|^p \right)^{1/p} \right)^p \right)^{1/p}. \end{aligned} \tag{5.14}$$

As $|\mu| = |\lambda|$ for all the sums, we can write (5.14) as

$$\begin{aligned} &= 2^{|\lambda|(1/2-1/p)} 2^{-|\lambda|/2} \left(\sum_{\mu \in \zeta_\lambda^{|\lambda|}} \left(\sum_{j > |\mu|} 2^{j/2} \left(\sum_{\gamma \in \zeta_\mu^j} |d_\gamma|^p \right)^{1/p} \right)^p \right)^{1/p} \\ &= 2^{-|\lambda|/p} \left(\sum_{\mu \in \zeta_\lambda^{|\lambda|}} \left(\sum_{j > |\mu|} 2^{j/2} \left(\sum_{\gamma \in \zeta_\mu^j} |d_\gamma|^p \right)^{1/p} \right)^p \right)^{1/p}. \end{aligned}$$

Thus, we have that

$$|E(f, \square_\lambda) - E^*(f, \square_\lambda)|^p \lesssim 2^{-|\lambda|} \sum_{\mu \in \zeta_\lambda^{|\lambda|}} \left(\sum_{j > |\mu|} 2^{j/2} \left(\sum_{\gamma \in \zeta_\mu^j} |d_\gamma|^p \right)^{1/p} \right)^p. \tag{5.15}$$

Now, consider the term

$$\begin{aligned}
&\sum_{j>|\mu|} 2^{j/2} \left(\sum_{\gamma \in \zeta_\mu^j} |d_\gamma|^p \right)^{1/p} \\
= &\sum_{j>|\mu|} 2^{-j(s-1/p)} 2^{j(s+1/2-1/p)} \left(\sum_{\gamma \in \zeta_\mu^j} |d_\gamma|^p \right)^{1/p} \\
\leq &\left(\sum_{j>|\mu|} 2^{-jq'(s-1/p)} \right)^{1/q'} \left(\sum_{j>|\mu|} 2^{jq(s+1/2-1/p)} \left(\sum_{\gamma \in \zeta_\mu^j} |d_\gamma|^p \right)^{q/p} \right)^{1/q}
\end{aligned}$$

where $1/q + 1/q' = 1$. Then, using (5.12), we have

$$\sum_{j>|\mu|} 2^{j/2} \left(\sum_{\gamma \in \zeta_\mu^j} |d_\gamma|^p \right)^{1/p} \lesssim \|f\|_{B_q^s(L_p(\square_\lambda^*))} \left(\sum_{j>|\mu|} 2^{-jq'(s-1/p)} \right)^{1/q'}. \tag{5.16}$$

Now, consider the term

$$\sum_{j>|\mu|} 2^{-jq'(s-1/p)}.$$

This is a geometric series and is guaranteed to converge if $q' > 0$ and $s > 1/p$. We are usually only interested in the case $p = 2$; in this case the condition reduces to $s > 1/2$ which we assumed earlier. Now, the series converges to

$$\frac{2^{-(s-1/p)q'(|\mu|+1)} 2^{q'(s-1/p)}}{2^{q'(s-1/p)} - 1} = \frac{2^{-q'(s-1/p)|\mu|}}{2^{q'(s-1/p)} - 1}. \tag{5.17}$$

Then, substituting (5.17) into (5.16), we have

$$\sum_{j>|\lambda|} 2^{j/2} \left(\sum_{\gamma \in \zeta_\mu^j} |d_\gamma|^p \right)^{1/p} \lesssim \|f\|_{B_q^s(L_p(\square_\lambda^*))} \left(\frac{2^{-q'(s-1/p)|\mu|}}{2^{q'(s-1/p)} - 1} \right)^{1/q'}. \tag{5.18}$$

Then, substituting (5.18) into (5.15)

$$\begin{aligned} |E(f,\square_\lambda) - E^*(f,\square_\lambda)|^p &\lesssim 2^{-|\lambda|} \sum_{\mu\in\zeta_\lambda^{|\lambda|}} \left(\frac{2^{-|\mu|(s-1/p)}}{(2^{q'(s-1/p)}-1)^{1/q'}} \right)^p \|f\|^p_{B^s_q(L_p(\square^*_\lambda))} \\ &= 2^{-|\lambda|} \sum_{\mu\in\zeta_\lambda^{|\lambda|}} \frac{2^{-|\lambda|p(s-1/p)}}{(2^{q'(s-1/p)}-1)^{p/q'}} \|f\|^p_{B^s_q(L_p(\square^*_\lambda))}. \end{aligned}$$

The last equality holds as $|\mu| = |\lambda|$ for all the terms in the sum. As the sum is independent of μ and $\zeta_\lambda^{|\lambda|}$ is a finite set where the size of the set depends only on the size of the scaling function, we have

$$|E(f,\square_\lambda) - E^*(f,\square_\lambda)|^p \lesssim \frac{2^{-|\lambda|}2^{-|\lambda|p(s-1/p)}}{(2^{q'(s-1/p)}-1)^{p/q'}} \|f\|^p_{B^s_q(L_p(\square^*_\lambda))}.$$

Taking each side to the power of $1/p$, we have

$$|E(f,\square_\lambda) - E^*(f,\square_\lambda)| \lesssim \frac{2^{-|\lambda|s}}{(2^{q'(s-1/p)}-1)^{1/q'}} \|f\|_{B^s_q(L_p(\square^*_\lambda))}.$$

As the denominator only depends on p, q, s, i.e. it is independent of f and λ, we have that

$$|E(f,\square_\lambda) - E^*(f,\square_\lambda)| \lesssim C2^{-|\lambda|s} \|f\|_{B^s_q(L_p(\square^*_\lambda))}$$

where C is independent of f and λ. That is

$$|E(f,\square_\lambda) - E^*(f,\square_\lambda)| \lesssim 2^{-|\lambda|s} \|f\|_{B^s_q(L_p(\square^*_\lambda))}.$$

This indicates that the error in substituting the error function $E(f,\square_\lambda)$ with $E^*(f,\square_\lambda)$ depends on the local Besov norm of f and decreases as $|\lambda|$ increases. This justifies the substitution if we are dealing with functions with moderate Besov norms, or if $|\lambda|$ is large.

Thus from now on, when we compute partitions of domains, we will use the error function $E^*(f,\square_\lambda)$.

5.3 The DSX Algorithm

The DSX algorithm can be used to construct an adaptive local scaling function representation. It was first introduced by Wolfgang Dahmen, Reinhold Schneider and Yuesheng Xu in [7]. They designed the algorithm to project non-linear compositions of functions on to an adaptive grid. The major step that the authors did not consider was how to actually compute the local scaling function representation. i.e. They only looked at the algorithm at a theoretical point of view and never actually computed the representation. Thus, we are going to look in detail at how to construct the adaptive local scaling function representation. The algorithm has four main steps, but we only require two of them: the prediction and the iteration. We have discussed the prediction as well as some necessary assumptions and norm equivalences that we require. Now, we are ready to consider the iteration and look at its simplifications.

5.3.1 The Iteration

Suppose that we have predicted a partition set Λ of the domain Ω. The following notation needs to be introduced.

$$j_0 := \min_{\lambda \in \Lambda} |\lambda| \text{ and } J := \max_{\lambda \in \Lambda} |\lambda|$$

are the coarsest and finest levels respectively of the partition set Λ.

$$\Lambda_j := \{\lambda \mid |\lambda| = j\}$$

is the index set of the intervals partitioned at level j.

$$\Omega_j := \bigcup_{\lambda \in \Lambda_j} \square_\lambda$$

is the union of all the intervals from level j. The set of significant indices is given by

$$\Upsilon_j := \{\lambda \;\Big|\; |\lambda| = j \text{ and } |w_\lambda \cap \Omega_j| \neq 0\}$$

where

$$w_\lambda := \text{supp } \phi_\lambda.$$

Now, define the projection on to level j as

$$P_j f := \sum_{\lambda \in \Upsilon_j} q_\lambda(f)\phi_\lambda, \tag{5.19}$$

where q is some non-adaptive quasi-interpolation scheme with polynomial regularity m. Earlier, we denoted by P_j the map

$$\begin{aligned} P_j &: L_2(\mathbb{R}) \to V_j \\ P_j &: f \mapsto \sum_{|\lambda|=j} \langle f, \tilde{\phi}_\lambda \rangle \phi_\lambda. \end{aligned}$$

However, P_j will be defined as in (5.19) from now on. We should note that Υ_j is the smallest set such that for all $x \in \Omega_j$, we have

$$\sum_{\lambda \in \Upsilon_j} = q_\lambda(f)\phi_\lambda(x) = \sum_{|\lambda|=j} q_\lambda(f)\phi_\lambda(x).$$

Recall that we are trying to build an adaptive quasi-interpolation scheme in a recursive manner. Let $P_{\Lambda,j_0} := P_{j_0}$ be the initial projection and $P_\Lambda := P_{\Lambda,J}$ be the final projection. The recursive formula is

$$P_{\Lambda,j}(f) := P_{\Lambda,j-1} f + P_j f - \sum_{\lambda \in \Upsilon_j} \langle P_{\Lambda,j-1}, \tilde{\phi}_\lambda \rangle \phi_\lambda. \tag{5.20}$$

This iteration is designed by simply adding up the non-adaptive projections at all the levels between j_0 and J, and then subtracting off the overlap of the levels which

in some sense we have counted twice.

Now, let m be the polynomial regularity of the non-adaptive quasi-interpolant P_j. Then, it is shown in [3] that P_Λ is E-admissible, has polynomial regularity m and is local in the sense that if $U \subset \Omega_j$, then

$$P_\Lambda f|_U = P_\Lambda(\chi_{U+2^{-j}+K} f)|_U$$

where K is the compact set making P_j local as in Definition 4.2.1(a). So, P_Λ is exactly what we were looking for and thus P_Λ also satisfies [7]

$$\|P_\Lambda f - f\|_{L_2(\square_\lambda)} \lesssim \inf_{p \in \Pi_m} \|p - f\|_{L_2(\square^*_\lambda)}.$$

The difficulty in performing the iterative step (5.20) is that it is very expensive to compute

$$\langle P_{\Lambda,j-1} f, \tilde{\phi}_\lambda \rangle.$$

To see that this correction term can be evaluated exactly, first note that

$$P_{\Lambda,j-1} f \in V_{j-1}.$$

So, there is an expansion of $P_{\Lambda,j-1} f$ of the form

$$P_{\Lambda,j-1} f = \sum_{|\mu|=j-1} c_\mu \phi_\mu.$$

Then,

$$\langle P_{\Lambda,j-1} f, \tilde{\phi}_\lambda \rangle = \sum_{|\mu|=j-1} c_\mu \langle \phi_\mu, \tilde{\phi}_\lambda \rangle.$$

Thus, we really only need to be able to evaluate

$$\int \phi(x) \tilde{\phi}(2x - k) dx.$$

Using the refinement equation, we have

$$\begin{aligned}\int \phi(x)\tilde{\phi}(2x-k)dx &= \int \sqrt{2}\sum_l h_l \phi(2x-l)\tilde{\phi}(2x-k)dx \\ &= \frac{\sqrt{2}}{2}\sum_l h_l \int \phi(x-l)\tilde{\phi}(x-k)dx \\ &= \frac{\sqrt{2}}{2}h_k.\end{aligned}$$

Even though implementing the refinement and duality equations is relatively simple, performing the change of basis requires applying the refinement equation multiple times. Thus, we would like a method to avoid having to implement this change of basis. As we will see, we can do this if we carefully select our quasi-interpolation scheme.

5.3.2 The DSX Algorithm with the Discrete Dual

It turns out that the discrete dual introduced by Ware can evaluate the correction term efficiently. Suppose we use a discrete dual in the quasi-interpolant for the DSX algorithm. That is,

$$P_j f = \sum_{\lambda \in \Upsilon_j} \theta_\lambda(f)\phi_\lambda.$$

Recall that if $f \in V_j$, then $\langle f, \tilde{\phi}_\lambda \rangle = \theta_\lambda(f)$. Now, the problematic term is

$$\sum_{\lambda \in \Upsilon_j} \langle P_{\Lambda,j-1} f, \tilde{\phi}_\lambda \rangle \phi_\lambda.$$

As $P_{\Lambda,j-1}f$ is in the space

$$V_{j_0} \oplus \ldots \oplus V_{j-1} \subset V_j,$$

we can use the discrete dual in order to compute the correction term. In other words, if $|\lambda| = j$, then

$$\theta_\lambda(P_{\Lambda,j-1}f) = \langle P_{\Lambda,j-1}f, \tilde{\phi}_\lambda \rangle.$$

So, the DSX algorithm when using the quasi-interpolation scheme of Ware simplifies as

$$\begin{aligned} P_j f &= \sum_{\lambda \in \Upsilon_j} \theta_\lambda(f)\phi_\lambda \\ P_{\Lambda,j_0} &= P_{j_0} \\ P_\Lambda &= P_{\Lambda,J} \\ P_{\Lambda,j}f &= P_{\Lambda,j-1}f + P_j f - P_j(P_{\Lambda,j-1}f). \end{aligned} \tag{5.21}$$

Then, writing (5.21) in the summation notation, we have

$$\begin{aligned} P_{\Lambda,j}f &= P_{\Lambda,j-1}f + \sum_{\lambda \in \Upsilon_j} \theta_\lambda(f)\phi_\lambda - \sum_{\lambda \in \Upsilon_j} \theta_\lambda(P_{\Lambda,j-1}f)\phi_\lambda \\ &= P_{\Lambda,j-1}f + \sum_{\lambda \in \Upsilon_j} \theta_\lambda(f - P_{\Lambda,j-1}f)\phi_\lambda. \end{aligned} \tag{5.22}$$

The iterative step (5.22) is also easy to program and we no longer require a change of basis.

In fact, if we use the discrete dual, the operators $P_{\Lambda,j}$ take on an interesting form. For the sake of simplicity, suppose that $j_0 = 0$. The more general case can easily be seen from this case. From (5.21), we have that

$$P_{\Lambda,j}f = P_{\Lambda,j-1}f + P_j(f - P_{\Lambda,j-1}f).$$

Thus,

$$\begin{aligned} P_{\Lambda,1}f &= P_{\Lambda,0}f + P_1(f - P_{\Lambda,0}f) \\ &= P_0 f + P_1(f - P_0 f) \\ &= (P_1 + P_0)f - P_1 P_0 f. \end{aligned}$$

Going up one more level, it is not hard to see that

$$P_{\Lambda,2}f = (P_2 + P_1 + P_0)f - (P_1P_0 + P_2P_0 + P_2P_1)f + P_2P_1P_0 f.$$

Then, we can predict the form of $P_{\Lambda,j}$ where j is an integer between j_0 and J.

Theorem 5.3.1

$$P_{\Lambda,j}f = \sum_{n=1}^{j+1}(-1)^{n+1} \sum_{i_n > \ldots > i_1} (P_{i_n} \ldots P_{i_1})f,$$

where $i_{j+1}, \ldots, i_1$ *are all integers between* 0 *and* j.

Proof Throughout the proof, we will always assume that $i_{j+1}, \ldots, i_1$ are integers between 0 and j. The result will be shown by induction. The base case is done above. Next, suppose that

$$P_{\Lambda,j}f = \sum_{n=1}^{j+1}(-1)^{n+1} \sum_{i_n > \ldots > i_1} (P_{i_n} \ldots P_{i_1})f.$$

Then,

$$\begin{aligned} P_{\Lambda,j+1} &= P_{\Lambda,j} + P_{j+1}(f - P_{\Lambda,j}) \\ &= \sum_{n=1}^{j+1}(-1)^{n+1} \sum_{i_n > \ldots > i_1} (P_{i_n} \ldots P_{i_1})f + \\ &\quad P_{j+1}f - \sum_{n=1}^{j+1}(-1)^{n+1} \sum_{i_n > \ldots > i_1} (P_{j+1}P_{i_n} \ldots P_{i_1})f. \end{aligned}$$

Realizing that the last sum introduces all the new combinations of the projections, we have that

$$P_{\Lambda,j+1}f = \sum_{n=1}^{j+2}(-1)^{n+1}\sum_{i_n>\ldots>i_1}(P_{i_n}\ldots P_{i_1})f.$$

□

This form of the adaptive local scaling function representation will play a role when we consider point evaluation of the representation.

5.4 Numerical Experiment

Now that we have seen results of the local scaling function representation, we can look at some numerical examples. First, we will simply look at some partitions formed by different values of ϵ and `maxiter` using the discrete dual. Then, we will use the discrete dual to construct the actual adaptive local scaling function representation and consider the error in the representation.

Example Consider the function

$$f(x) = \begin{cases} \sin(\pi x) & x \in [0,1]\cup[5,6] \\ 0 & x \notin [0,1]\cup[5,6] \end{cases}.$$

At $x = 0, 1, 5, 6$ the first derivative has a jump discontinuity. Thus, we expect the local Besov norms in the intervals containing these points to be large. This would increase the value of the error function, forcing the oracle to refine the grid in these regions. A plot of the function is in Figure 5.1.

Using different values for ϵ and `maxiter`, the oracle was iterated on $[-1, 7]$ initially partitioned at the half integers. The results are summarized along with the number

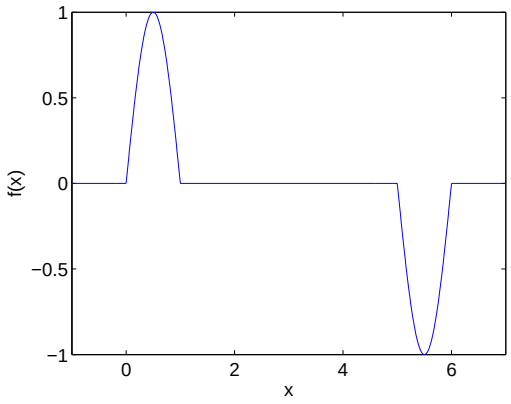

Figure 5.1: A function with large local Besov norms

ϵ	`maxiter`	$\lvert\Lambda\rvert$
$1e^{-1}$	4	24
$1e^{-1}$	8	24
$1e^{-3}$	4	76
$1e^{-3}$	8	116

Table 5.1: Different partitions of the domain of the function

of points in the partition in Table 5.1. Figures 5.2 and 5.3 contain the plots of the different partitions. Then Figure 5.4 is the same plots as Figure 5.3, just zoomed in to illustrate that more points have been added around $x = 0$.

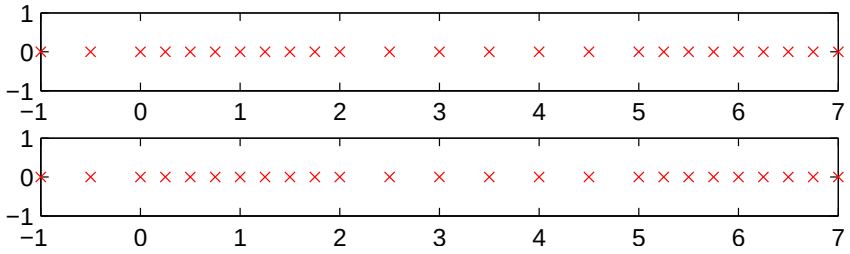

Figure 5.2: $\epsilon = 1e^{-1}$ and `maxiter`=4 (top) `maxiter` =8 (bottom)

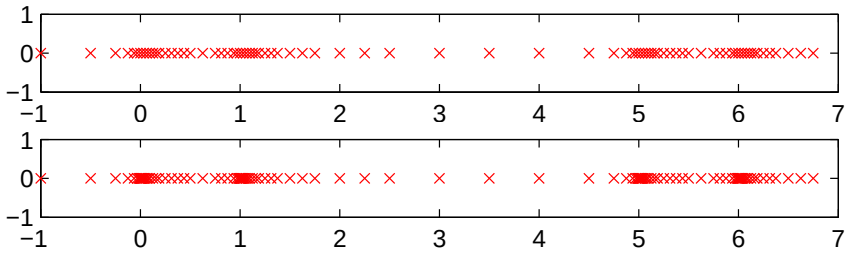

Figure 5.3: $\epsilon = 1e^{-3}$ and `maxiter`=4 (top) `maxiter` =8 (bottom)

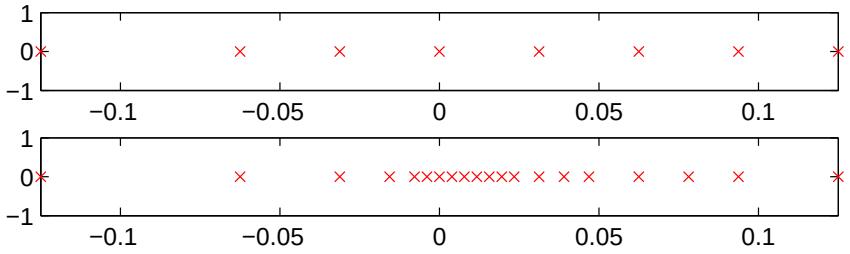

Figure 5.4: A close up of $\epsilon = 1e^{-3}$ and `maxiter`=4 (top) `maxiter`=8 (bottom)

Note that for the case of $\epsilon = 1e^{-1}$, no new points were added to the partition when `maxiter` was increased from 4 to 8. This is because the error function in each subset of $[-1, 7]$ after four iterations was already less than ϵ. However, for the case

of $\epsilon = 1e^{-3}$, more points were added to the partition, and as expected they were concentrated around the points $x = 0, 1, 5, 6$. In the above example, a discrete dual of the third B-spline was used.

Finally, we should note that this could have had a large contribution of aliasing error. If the initial partition had been at only integer values, the oracle would have done nothing as the function values at all the partition points would have been zero. Thus, it was crucial that the initial partition was at the half integers.

Now that we have seen some partitions of the domain, we are ready to form the actual adaptive representation.

Example Again, consider the example

$$f(x) = \begin{cases} \sin(\pi x) & x \in [0,1] \cup [5,6] \\ 0 & x \notin [0,1] \cup [5,6] \end{cases}$$

with the domain

$$\Omega = [-1, 7].$$

After forming the partition for different values of ϵ and `maxiter` and then calculating the adaptive local scaling function representation using the DSX algorithm with the discrete dual, the resulting approximations along with the errors are plotted in Figures 5.5, 5.6, 5.7 and 5.8.

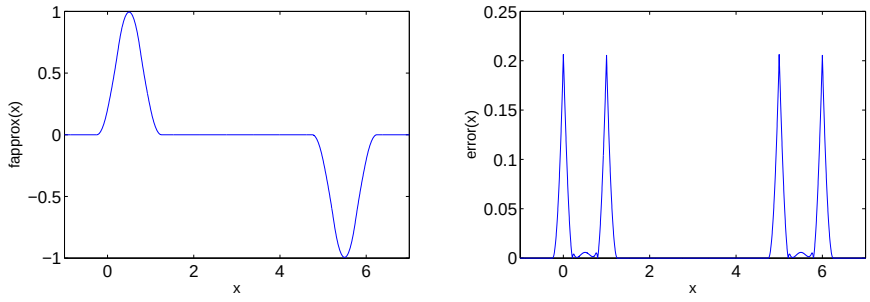

Figure 5.5: `fapprox(x)` and `error(x)` for $\epsilon = 1e^{-1}$ and `maxiter`=4

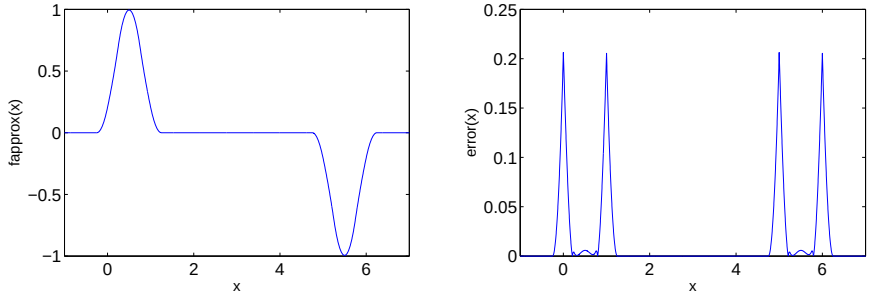

Figure 5.6: `fapprox(x)` and `error(x)` for $\epsilon = 1e^{-1}$ and `maxiter`=8

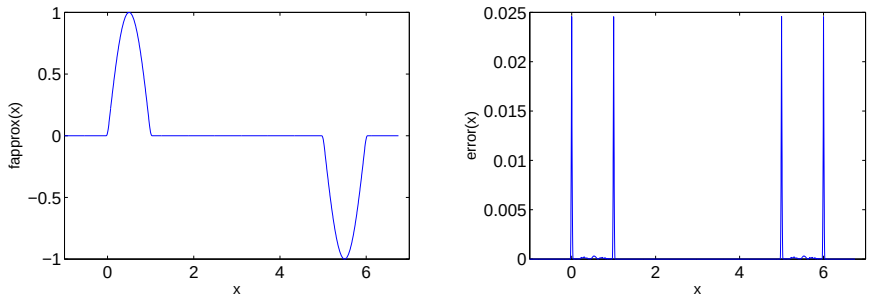

Figure 5.7: `fapprox(x)` and `error(x)` for $\epsilon = 1e^{-3}$ and `maxiter`=4

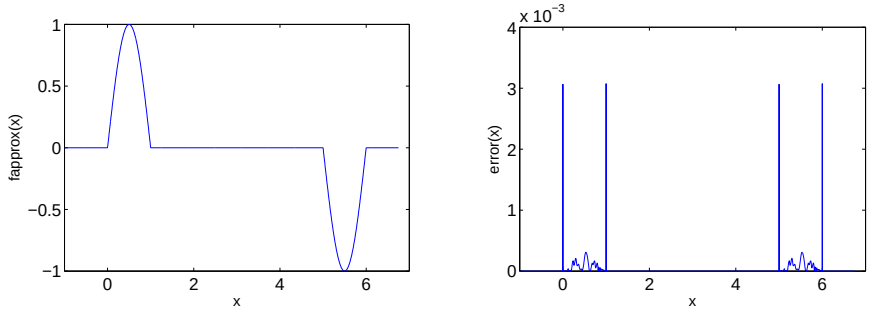

Figure 5.8: `fapprox(x)` and `error(x)` for $\epsilon = 1e^{-3}$ and `maxiter`=8

As before when $\epsilon = 1e^{-1}$, the sufficient partition is determined by level four. This is why the error between `maxiter`=4 and `maxiter`=8 has not changed. Also, recall the definition of an E-admissible operator. The idea is to force all the local errors to be of order less than ϵ. When looking at the plots of the error, it can be seen that this is the case. Also, as expected, the error as well as the partition points are concentrated around the singularities $x = 0, 1, 5, 6$.

Table 5.2 is a summary of numerical estimates of the $L_2(\Omega)$ error of the approximations for the different values of ϵ and `maxiter`.

ϵ	`maxiter`	$L_2(\Omega)$
$1e^{-1}$	4	$5.9939e^{-3}$
$1e^{-1}$	8	$5.9339e^{-3}$
$1e^{-3}$	4	$2.4951e^{-5}$
$1e^{-3}$	8	$1.4841e^{-5}$

Table 5.2: $L_2(\Omega)$ error of some different representations

Recalling the bound from (5.3), it should be the case that $|\Gamma|^{-s}$ is related to the $L_2(\Omega)$ error. We can see that this is the case as

$$\begin{aligned} 5.9939e^{-3} &< 1/24 \approx 0.0417 \\ 2.4951e^{-5} &< 1/76 \approx 0.0132 \\ 1.4841e^{-5} &< 1/116 \approx 0.0086. \end{aligned}$$

Chapter 6

Applications

There are many applications from areas such as Physics and Engineering that local scaling function representations can be applied to. In general, performing even the simplest tasks on a complicated function can be computationally difficult. An adaptive local scaling function representation gives us an alternative representation of complicated functions. Since scaling functions have some nice properties, these can be taken advantage of when performing these tasks.

As the DSX algorithm along with the discrete dual is computationally simple, Matlab code has been written in order to form all of the plots that will be considered in this chapter.

6.1 Point Evaluation

In order to look at almost any property of a function, we need to be able to perform point evaluation. In our case, we are interested in doing point evaluation efficiently. In other words, if we only want to evaluate our approximation at one point, we do not want to have to build the entire approximation.

Let $\Omega \subseteq \mathbb{R}$ and $f : \Omega \to \mathbb{R}$. Let Λ be a partition set of Ω, and let j_0 and J be the coarsest and finest levels of Λ respectively. Then, we have the single-level projections given by

$$P_j f := \sum_{\lambda \in \Upsilon_j} \theta_\lambda(f)\phi_\lambda.$$

Recall that the iterative step of the DSX algorithm simplifies as

$$\begin{aligned} P_{\Lambda,j_0}f &= P_{j_0}f \\ P_{\Lambda,j}f &= P_{\Lambda,j-1}f + \sum_{\lambda\in\Upsilon_j}\theta_\lambda(f - P_{\Lambda,j-1}f)\phi_\lambda \\ P_\Lambda f &= P_{\Lambda,J}f. \end{aligned}$$

Now, let $x \in \Omega$ be some point at which we are interested in evaluating

$$P_\Lambda f(x).$$

As Λ is a partition set of Ω, we know that there is an integer j between j_0 and J such that $x \in \Omega_j$. Consider the following algorithm.

$lev := j$

for $i = j, \ldots, J$

$S_{x,i} := \left\{ \lambda \in \Upsilon_i \;\middle|\; x \in w_\lambda \right\}$

if $S_{x,i} \neq \emptyset$

$lev \leftarrow i$

end

end

Note that the algorithm returns an integer lev with $lev \leq J$. Then, we have the following result.

Theorem 6.1.1 $P_\Lambda f(x) = P_{\Lambda,lev}f(x)$.

Proof If $x \notin \omega_\lambda$, then $\phi_\lambda(x) = 0$, so if $x \notin \omega_\lambda$ for all $\lambda \in \Upsilon_j$, then $\phi_\lambda(x) = 0$ for all $\lambda \in \Upsilon_j$, so that if this is the case,

$$\sum_{\lambda\in\Upsilon_j}\theta_\lambda(f - P_{\Lambda,j-1}f)\phi_\lambda(x) = 0.$$

Then,

$$\begin{aligned} P_{\Lambda,j}f(x) &= P_{\Lambda,j-1}f(x) + \sum_{\lambda\in\Upsilon_j} \theta_\lambda(f - P_{\Lambda,j-1}f)\phi_\lambda(x) \\ &= P_{\Lambda,j-1}f(x). \end{aligned} \tag{6.1}$$

The design of the above algorithm picks the largest level i such that

$$\left\{\lambda \in \Upsilon_i \;\middle|\; x \in \omega_\lambda\right\} \neq \emptyset,$$

and calls it `lev`. Thus, (6.1) indicates that

$$P_\Lambda f(x) = P_{\Lambda,J-1}f(x) = \ldots = P_{\Lambda,lev}f(x).$$

□

The importance of Theorem 6.1.1 is that we do not need to build the entire iteration in order to evaluate the approximation at a point. This would be extremely important if the point of evaluation x only requires us to go up a few levels of a rather large iteration.

Now, there are restrictions we can impose on our partition in order to simplify point evaluation. First, suppose that our partition is formed so that if $\lambda, \mu \in \Lambda$ form two neighboring intervals, then

$$\big||\lambda| - |\mu|\big| \leq 1.$$

That is, two neighboring intervals never differ by more than one level. Now, suppose that for all $x \in \Omega$,

$$S_{x,i} = \emptyset \text{ for all } i < j-1 \text{ and for all } i > j+1, \tag{6.2}$$

where $x \in \Omega_j$. Then, the single-level projections $P_i f$ for all $i < j-1$ and $i > j+1$ satisfy

$$P_i f(x) = 0.$$

In other words, if the condition given in (6.2) is imposed, the only projections that affect the value of $P_\Lambda f(x)$ are $P_{j-1}f, P_j f, P_{j+1}f$. So, using Theorem 5.3.1 and Theorem 6.1.1, we have that

$$\begin{aligned} P_\Lambda f(x) &= P_{\Lambda,lev} f(x) \\ &= P_{\Lambda,j+1} f(x) \\ &= (P_{j+1} + P_j + P_{j-1}) f(x) - (P_j P_{j-1} + P_{j+1} P_{j-1} \\ &\quad + P_{j+1} P_j) f(x) + P_{j+1} P_j P_{j-1} f(x). \end{aligned} \tag{6.3}$$

Now, in order to impose (6.2), we need to impose that for all j and for all $\lambda \in \Upsilon_j$

$$|w_\lambda \cap \Omega_k| = 0 \text{ for all } k \neq j-1, j, j+1.$$

This can be achieved by making sure that, for any j, there are enough adjacent intervals at level j before there is a jump in levels. The number of adjacent intervals, as we will see, depends only on the support of the scaling function. This condition is described in [7], and a partition that satisfies it is called M-graded where M is some positive integer. i.e. A partition is said to be M-graded if after there is a jump between two levels, there are M intervals at this new level before the next jump.

Now, we have constructed partitions that are not necessarily M-graded and may not even jump only one level when changing levels. However, a program could be written that takes any partition set of a domain and maps it to the smallest M-graded partition set containing the original partition set. In fact, it is shown in [7]

that for any partition set Λ and any $M \in \mathbb{N}$, there is an M-graded partition set $\hat{\Lambda}$ such that

$$\Lambda \subseteq \hat{\Lambda} \text{ and } |\Lambda| \lesssim |\hat{\Lambda}|.$$

Thus, we can always refine our partition to an M-graded partition without adding too many new points to the partition.

In order to choose a value of M to guarantee ourselves that an M-graded partition will simplify point evaluation as described in (6.3), we need the following theorem.

Theorem 6.1.2 *If $M \geq 2(n-1)$ where n is the size of the support of the scaling function ϕ, then we are guaranteed that the point evaluation simplifies as in (6.3).*

Proof We will only prove the result in the simplest case. The more general case will become apparent from this case. Suppose that our scaling function has support $[0, n]$, and thus

$$\text{supp } \phi_{j,k} = [2^{-j}k, 2^{-j}(n+k)].$$

Suppose that we have an interval at level j of the form $[0, 2^{-j}]$ and furthermore, suppose we want to move to level $j+1$ in the next interval. Now, the values k in Υ_j that intersect with the interval $[0, 2^{-j}]$ on a set not of measure zero are

$$k \in \{-(n-1), \ldots, 0\}.$$

Thus, the support of the approximation at this level will extend to the right until the point $2^{-j}n$. Thus, we need level $j+1$ to cover the entire interval

$$[2^{-j}, 2^{-j}n].$$

As the measure of this interval is $2^{-j}(n-1)$, and the length of intervals at level $j+1$ are $2^{-(j+1)}$, a counting argument indicates that the number of intervals of level $j+1$ before another jump must be at least

$$\frac{2^{-j}(n-1)}{2^{-(j+1)}} = 2(n-1).$$

□

Thus, for Daubechies' second scaling function, in order to simplify the point evaluation as described above, our partition must be M-graded where $M \geq 2(3-1) = 4$.

6.2 Approximating a Derivative

Now that we can represent subsets of $L_2(\mathbb{R})$ as a local scaling function representation, we can look at derivatives. More specifically, we can approximate the derivative of f by looking at the derivative of a local scaling function representation of f. Suppose that we have some function f in an appropriate Besov space. Representing f as a local scaling function, we have an approximation of the form

$$P_\Lambda f = \sum_{j=j_0}^{J} \sum_{|\lambda|=j} c_\lambda \phi_\lambda.$$

Let x be some point where we are interested in the derivative of f. Then, assuming that ϕ_λ has a derivative at x for all significant λ, we can estimate $Df(x)$ by

$$DP_\Lambda f(x) = \sum_{j=j_0}^{J} \sum_{|\lambda|=j} c_\lambda D\phi_\lambda(x),$$

where D is the derivative operator. To ease notation, consider using the variables j and k. That is,

$$\begin{aligned} DP_\Lambda(x)f &= \sum_{j=j_0}^{J}\sum_{k\in\mathbb{Z}} c_{j,k} D\phi_{j,k}(x) \\ &= \sum_{j=j_0}^{J}\sum_{k\in\mathbb{Z}} c_{j,k} D(2^{j/2}\phi(2^j x-k)) \\ &= \sum_{j=j_0}^{J}\sum_{k\in\mathbb{Z}} c_{j,k} 2^{3j/2}\phi'(2^j x-k) \\ &= \sum_{j=j_0}^{J} 2^{3j/2}\sum_{k\in\mathbb{Z}} c_{j,k}\phi'(2^j x-k). \end{aligned}$$

Example A scaling function that has one continuous derivative is the third B-spline. Suppose we are trying to differentiate the function

$$f(x) = \begin{cases} \sin(\pi x) & x \in [0,1]\cup[5,6] \\ 0 & x \notin [0,1]\cup[5,6] \end{cases} \tag{6.4}$$

at the point $x = 1/4$. Then, using the parameters $\epsilon = .01$ and `maxiter`=3, we get an approximation of f

$$P_\Lambda f = \sum_{j=3}^{4} 2^{j/2}\sum_{k\in\mathbb{Z}} c_{j,k}\phi(2^j x-k).$$

Taking the derivatives of each side and using the fact that all the sums are finite, we have that

$$DP_\Lambda f = \sum_{j=3}^{4} 2^{3j/2}\sum_{k\in\mathbb{Z}} c_{j,k} D\phi(2^j x-k).$$

Letting x=1/4, we get that

$$DP_\Lambda f(1/4) = 1.2814.$$

Evaluating the true derivative, we get that

$$f'(1/4) = 2.2214.$$

So, in this example, we see quit a bit of error. This is because we only found a representation in V_4. The error could be decreased by moving to a higher multiresolution. However, this becomes computationally expensive and cumbersome.

A more practical method of approximating a derivative would be to use a numerical technique of differentiation on a local scaling function representation $P_\Lambda f$ of f. The Matlab code is able to numerically differentiate the local scaling function representation using:

The forward-difference formula,

$$f'(x_0) \approx \frac{f(x_0 + h) - f(x_0)}{h}.$$

The backward-difference formula,

$$f'(x_0) \approx \frac{f(x_0) - f(x_0 - h)}{h}.$$

The three-point formula,

$$f'(x_0) \approx \frac{f(x_0 + h) - f(x_0 - h)}{2h}.$$

The five-point formula,

$$f'(x_0) \approx \frac{1}{12h}(f(x_0 - 2h) - 8f(x_0 - h) + 8f(x_0 + h) - f(x_0 + 2h).$$

Numerical differentiation relies on being able to do point evaluation. However, we have already looked at how to do this in an efficient manner in a previous section.

Below are plots of the derivative of

$$f(x) = \begin{cases} \sin(\pi x) & x \in [0,1] \cup [5,6] \\ 0 & x \notin [0,1] \cup [5,6] \end{cases} .$$

using the forward-difference formula, three-point formula and five-point formula. As well, there is a plot of the true derivative of f. Below, I am denoting the adaptive local scaling function representation of f by Pf.

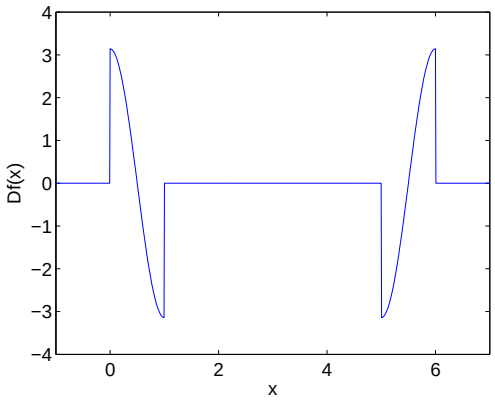

Figure 6.1: Derivative of $f(x)$.

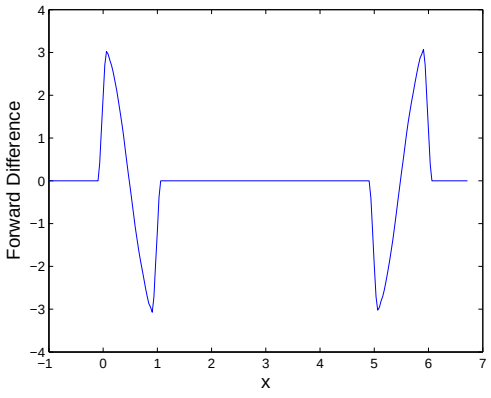

Figure 6.2: Forward-difference formula acting on Pf.

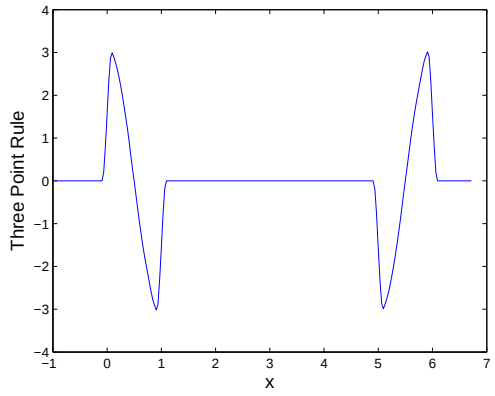

Figure 6.3: Three-point formula acting on Pf.

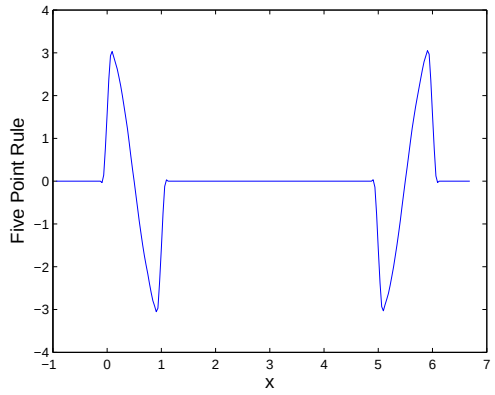

Figure 6.4: Five-point formula acting on Pf.

Now, we have not really taken full advantage of the adaptivity when considering numerical differentiation. In order to do this, we should not be partitioning the domain any finer than the oracle does (or at least only partitioning when it is absolutely necessary). Then, numerical differentiation at some point x would only depend on the interval $\square_\lambda$ containing x and the step size h would be the measure of $\square_\lambda$. At this point, the differentiation is truly adaptive.

6.3 Approximating an Integral

Consider some compactly supported function f with appropriate Besov norms. Let Ω be the support of f. Suppose we are interested in approximating

$$\int_{\mathbb{R}} f.$$

Instead of integrating f, we can integrate $P_\Lambda f$. That is, we simply try to approximate

$$\int_{\mathbb{R}} f \text{ with } \int_{\mathbb{R}} P_\Lambda f.$$

Now, $P_\Lambda f$ is a representation of f in $V_{j_0} \oplus \cdots \oplus V_J$. Writing $P_\Lambda f$ as

$$P_\Lambda f = \sum_{|\lambda|=j_0}^{J} c_\lambda \phi_\lambda \tag{6.5}$$

we have

$$\int_{\mathbb{R}} P_\Lambda f = \int_{\mathbb{R}} \sum_{|\lambda|=j_0}^{J} c_\lambda \phi_\lambda.$$

Since everything has compact support, the sum is guaranteed to be finite. So, we can interchange the integral and the summation to get

$$\int_{\mathbb{R}} P_\Lambda f = \sum_{|\lambda|=j_0}^{J} c_\lambda \int_{\mathbb{R}} \phi_\lambda. \tag{6.6}$$

Recall that the scaling function has the property

$$\int_{\mathbb{R}} \phi(x)dx = 1.$$

Thus, we have that

$$\begin{aligned} \int_{\mathbb{R}} \phi_{j,k}(x) &= \int_{\mathbb{R}} 2^{j/2}\phi(2^j x - k)dx \\ &= 2^{j/2}2^{-j} \int_{\mathbb{R}} \phi(x)dx \\ &= 2^{-j/2}. \end{aligned}$$

This means that for any λ,

$$\int_{\mathbb{R}} \phi_\lambda = 2^{-|\lambda|/2}.$$

Substituting this into (6.6), we have that

$$\int_{\mathbb{R}} P_\Lambda f = \sum_{j=j_0}^{J} 2^{-j/2} \sum_{|\lambda|=j} c_\lambda. \tag{6.7}$$

That is, the integral of $P_\lambda f$ is simply a scaled sum of the coefficients in the expansion given by (6.5).

Example Consider the function

$$f(x) = \begin{cases} \sin(\pi x) & x \in [0,1] \\ 0 & x \notin [0,1] \end{cases}.$$

In this example, we are using Daubechies' second scaling function on the domain $[-1,2]$. To keep things simple, the case of $\epsilon = 1e^{-1}$ and `maxiter`=2 with an initial partition at the half integers is considered. The non-zero coefficients at the two different levels are given in Table 6.1. Summing up the coefficient from level 2 and level 3, we get

$$\begin{aligned} \sum_{|\lambda|=2} c_\lambda &= 1.2603 \\ \sum_{|\lambda|=3} c_\lambda &= 0.0136. \end{aligned}$$

Then, scaling these values and summing them according to (6.7), we get that

$$\int_{\mathbb{R}} P_\Lambda f \approx 0.6350.$$

The actual value of the integral is

$$\int_0^1 \sin(\pi x) dx = \frac{2}{\pi} \approx 0.6366.$$

k	$c_{2,k}$
0	0.2319
1	0.4652
2	0.4260
3	0.1372

k	$c_{3,k}$
0	-0.0227
1	0.0178
2	0.0268
3	-0.0100
4	0.0286
5	-0.0153
6	0.0142
7	-0.0117
8	-0.0313
9	0.0166
10	0.00036

Table 6.1: Coefficients in the adaptive local scaling function representation

Here we have only considered approximating the integral of f over its entire domain. If we wanted to look at integrals over subsets of the domain of f, things become increasingly more difficult. Even though scaling functions are local, there would be scaling functions that would straddle the integration limits. If this is the case, we would need a method to integrate a scaling function over only part of its domain.

Another technique would be to use numerical integration. As we know how to do point evaluation, techniques such as Simpson's rule or the trapezoid rule are easily implemented.

Suppose we have some function f whose definite integrals we are interested in. Now, define the function

$$F(t) := \int_{-\infty}^{t} f(x)dx.$$

F is good enough for evaluating any definite integral as

$$\int_{a}^{b} f(x)dx = F(b) - F(a).$$

From a numerical point of view, we first have to pick some step size Δt. Then, we can use the recursive formula

$$\begin{aligned}\int_{-\infty}^{t+\Delta t} f(x)dx &= \int_{-\infty}^{t} f(x)dx + \int_{t}^{t+\Delta t} f(x)dx \\ &\approx \int_{-\infty}^{t} f(x)dx + \frac{\Delta t}{2}(f(t+\Delta t) + f(t)).\end{aligned}$$

Here, we have used the trapezoid rule to numerically integrate. Then, using the notation $F(t)$, we have

$$F(t+\Delta t) \approx F(t) + \frac{\Delta t}{2}(f(t+\Delta t) + f(t)) \tag{6.8}$$

$$F(a) = 0 \tag{6.9}$$

where a is the left boundary of the support of f. We should note that all the error in this algorithm is accumulated as we iterate. Thus, one should take caution when observing results for this iteration.

Now, all we have to do is replace f with some adaptive local scaling function representation Pf of f. That is, we define

$$G(t) := \int_{0}^{t} Pf(x)dx,$$

and use the recursive formula

$$\begin{aligned}G(a) &= 0 \\ G(t+\Delta t) &= G(t) + \frac{\Delta t}{2}(Pf(t+\Delta t) + Pf(t))\end{aligned}$$

Example Consider the function

$$f(x) = \begin{cases} \sin(\pi x) & x \in [0,1] \cup [5,6] \\ 0 & x \notin [0,1] \cup [5,6] \end{cases}.$$

In this example, as $f(x) = 0$ for all $x \leq 0$, we do not have to integrate all the way to $-\infty$. That is, we define F and G as

$$\begin{aligned} F(t) &= \int_0^t f(x)dx \\ G(t) &= \int_0^t Pf(x)dx. \end{aligned}$$

It is easy to see that F can be written as

$$F(t) = \int_0^t f(x)dx = \begin{cases} \frac{1-\cos(\pi x)}{\pi} & x \in [0,1] \cup [5,6] \\ \frac{2}{\pi} & x \in (1,5) \\ 0 & x \in (6,\infty) \end{cases}.$$

Below, Daubechies' second scaling function with $\epsilon = 1e^{-4}$ and `maxiter`=6 is used to form an adaptive local scaling function representation of f. After iterating $G(t)$ in the recursive formula (6.8) with initial condition (6.9) and step size $\Delta t = 2^{-7}$, a plot of G(t) was formed and is given in Figure 6.5.

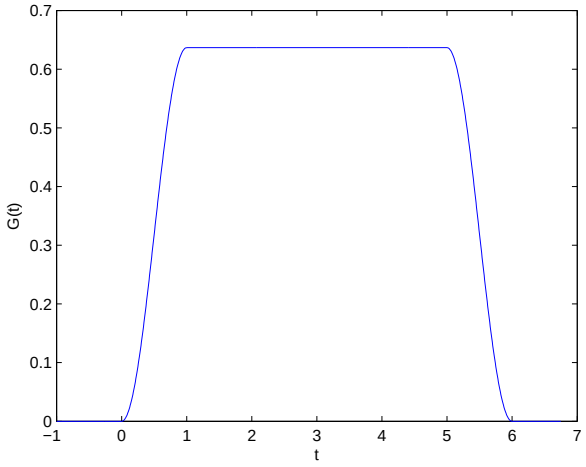

Figure 6.5: Approximation of an integral

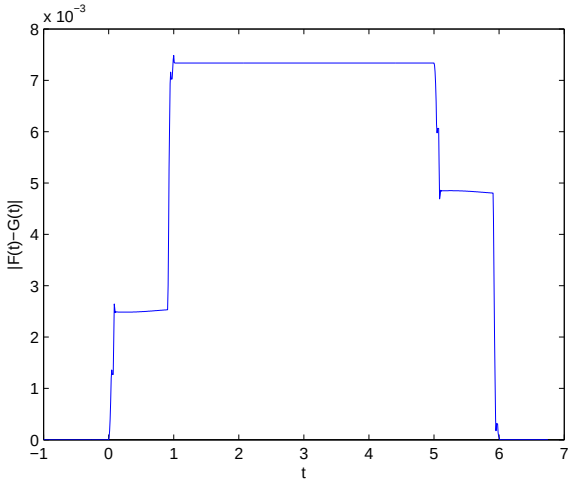

Figure 6.6: Error in an integral

The error in G is given in Figure 6.6. Using the l^2 error to approximate the L_2 error, we also have that

$$\|F(t) - G(t)\|_{L_2(\mathbb{R})} \approx 0.1779.$$

Again, we have not taken full advantage of the adaptivity in the representation. Ideally, we should approximate $\int_a^b f(x)dx$ as follows. First, write each interval $\square_\lambda$ as

$$\square_\lambda = [a_\lambda, b_\lambda]$$

Now, the interval of interest $[a, b]$ can be written as

$$[a, b] = [a, a_{\lambda_1}] \cup [a_{\lambda_1}, b_{\lambda_1}] \cup \cdots \cup [a_{\lambda_n}, b_{\lambda_n}] \cup [b_{\lambda_n}, b]$$

where $\square_{\lambda_i}$ with $i = 1, \cdots, n$ are all the intervals from the partition entirely contained in $[a, b]$. Then, we approximate $\int_a^b f(x)dx$ by

$$\begin{aligned} \int_a^b f(x)dx &\approx \int_a^b P_\Lambda f(x)dx \\ &= \int_a^{a_1} P_\Lambda f(x)dx + \int_{b_n}^b P_\Lambda f(x)dx + \sum_{i=1}^n \int_{a_i}^{b_i} P_\Lambda f(x)dx \end{aligned}$$

where we use some type of numerical integration. In this case, I am going to use the trapezoid rule so that

$$\begin{aligned}\int_a^b f(x)dx &\approx \frac{a_1-a}{2}(P_\Lambda f(a_1)+P_\Lambda f(a))+\frac{b-b_n}{2}(P_\Lambda f(b)+P_\Lambda f(b_n))\\ &\quad+\sum_{i=1}^{n}\frac{b_i-a_i}{2}(P_\Lambda f(b_i)+P_\Lambda f(a_i))\\ &= \frac{a_1-a}{2}(P_\Lambda f(a_1)+P_\Lambda f(a))+\frac{b-b_n}{2}(P_\Lambda f(b)+P_\Lambda f(b_n))\\ &\quad+\sum_{i=1}^{n}\frac{|\square_{\lambda_i}|}{2}(P_\Lambda f(b_i)+P_\Lambda f(a_i)). \qquad (6.10)\end{aligned}$$

Example Consider the same example as above, but this time we are going to use the adaptivity to evaluate

$$\int_0^1 f(x)dx$$

where

$$f(x)=\begin{cases}\sin(\pi x) & x\in[0,1]\\ 0 & x\notin[0,1]\end{cases}.$$

Now, using Daubechies' second scaling function, $\epsilon = 1e^{-4}$ and `maxiter=6`, we get a local scaling function representation of f. Then, approximating the integral according to (6.10) we get

$$\int_0^1 f(x)dx \approx 0.6376.$$

Again, the true value of the integral was

$$\int_0^1 f(x)dx = \frac{2}{\pi} \approx 0.6366.$$

6.4 Other Applications

Here we will briefly consider some of the classical applications of wavelet analysis. Another brief summary is given in [15].

6.4.1 Compression

The goal of compression is to store information while using as little data as possible. Say that we have information stored as a real-valued function f. Then, a non-linear approximation approach is to decompose f into some basis elements, and then disregard all the coefficients except for the largest N. Since we have eliminated some coefficients, this step loses some of the information stored in f. So, perfect reconstruction is not possible.

In the context of local scaling function representations, the oracle decides on a basis based on local Besov norms of f. Once we have all the coefficients, we store the N largest coefficients and the scale and shift of the coefficients. Then, our compressed signal becomes

$$P_N f = \sum_{i=1}^{N} c_{\lambda_i} \phi_{\lambda_i}$$

6.4.2 Denoising

Denoising is very similar to compression. In general, a compression scheme is performed, and then the compressed signal is reconstructed. The algorithm works if the noise is poorly approximated by wavelets.

In our setting, we have to assume that noise is poorly approximated by coarse scaling functions or that the noise is quite high in the multiresolution. Then, after we decompose a noisy signal f as an adaptive local scaling function representation, we simply pick the largest N coefficients, and let the rest of them be 0. Then, we reconstruct a new signal with these new coefficients and hope that some of the noise has been eliminated while not losing the general shape of the function.

6.4.3 Differential Equations

Techniques of wavelet analysis have been used in differential equations to study different types of problems. Wavelets are a good technique for studying differential equations that exhibit qualitative properties such as spikes, jumps or corners. As we have only been considering real valued functions, we have to restrict ourselves to one-dimensional differential equations. That is, we can only consider ordinary differential equations.

Consider a one-dimensional boundary value problem on the interval $[0, 1]$ given by

$$\begin{aligned} u''(t) &= f(t, u, u') \\ u(0) &= u(1) = 0. \end{aligned} \tag{6.11}$$

A Galerkin method assumes that we can find an approximation to the solution

$$u(t) \approx v(t) = \sum_{i=1}^{N} c_i v_i(t),$$

where $(v_i(t))_{i=1}^{N}$ are a linearly independent set whose span is close to the solution of (6.11). In our setting, a good choice would be a collection of scaling functions from multiple levels of a multiresolution. Then, the task is to minimize

$$v''(t) = f(t, v, v').$$

Minimizing this function is another problem in itself. We would require some method such as a least squares method to minimize the error. However, minimizing the error function

$$E(v) := v''(t) - f(t, v, v')$$

relies on efficient point evaluation.

A more complicated algorithm that one might be interested in is a one dimensional time dependent partial differential equation. For example, we may have a partial differential equation of the form

$$u_t = f(u, u_x)$$
$$u(x, 0) = g(x).$$

After discretizing the partial differential equation, one might be interested in moving up one time step. That is, given $u(x, t_0)$, find $u(x, t_0 + \Delta t)$. This task is not easy in the local scaling function representation setting, at least without constructing a scaling function representation from scratch each time we move up one time step. Even the task of finding a sufficient partition of $u(x, t_0 + \Delta t)$ based on the partition of $u(x, t_0)$ is very difficult.

The task of looking at differential equations in the wavelet setting is currently being researched by many experts. However, we will not be examining any further methods of using scaling functions to approximate solutions to differential equations.

6.4.4 Working at the Compressed Level

There are instances where if a function is represented at a compressed level, then necessary manipulations to the data can be performed at this compressed level. This becomes important when we are working with a large amount of data.

Consider the possibility that we could compress the images of an animation as local scaling function representations. Then, we could compress the images such that they are $1/n$ times the size of the uncompressed image. If we could perform some

necessary manipulation at this compressed level, we could perform it n times as fast. By Moore's Law, this is equivalent to waiting $\log_2(n)$ years for the computer speed to increase n fold.

6.5 Generalizations

There are many generalizations of all these results that could be looked at.

6.5.1 Higher Dimensions

We have only considered real-valued functions $f : \mathbb{R} \to \mathbb{R}$. A generalization could be built if we built the theory with $f : \mathbb{R}^n \to \mathbb{R}$. This requires scaling functions in higher dimensions. One of the simplest ways of constructing these is to take tensor products of the multiresolution with itself n times. Scaling functions and wavelet functions can be defined for this new multiresolution using the tensor product of the scaling and wavelet functions. Then, we are ready to do all the work we have considered in this thesis in higher dimensional spaces.

There are other ways to construct multiresolutions of $L_2(\mathbb{R}^n)$. Cohen discusses in [5] how to use a method of triangulation of $\mathbb{R}^n$ in order to form a multiresolution of $L_2(\mathbb{R}^n)$.

An important application comes from multiresolutions of $L_2(\mathbb{R}^2)$. An image can be considered as a compactly supported function $f : \mathbb{R}^2 \to \mathbb{R}$. That is, it is a map which takes a position (x, y) and maps it to a number associated to the colour bit at (x, y). Wavelet analysis is particularly useful in this setting as pictures may exhibit rapid changes in colour which could create large local Besov norms of f in this region.

6.5.2 Non Shift-Invariant Spaces

Wavelet analysis has many of its nice properties because in order to understand the entire multiresolution, we really only need to understand one function in V_0. This comes from the shifting and scaling properties of the multiresolution. However, there are ways to decompose $L_2(\mathbb{R})$ in a non shift-invariant setting. This leads to the question of how the above theory could be generalized in this setting. Except for acknowledging that it is a problem, it shall not be discussed any further in this thesis.

Chapter 7

Conclusions

Wavelet analysis is one of many methods to approximate a large class of functions to any desired tolerance. The locality of scaling and wavelet functions allows one to control the error at a local level. Thus, we can distribute the error evenly in an adaptive local scaling function representation.

The first obstacle in creating a local scaling function representation was calculating the scaling and wavelet function coefficients. It is known that in general, these coefficients can not be evaluated exactly. We have looked at some different techniques of approximating the coefficients as well as a method to help increase the convergence of the approximation to the true coefficients.

Once we were able to accurately approximate the necessary coefficients, we needed a method to create an adaptive projection. We have used the DSX algorithm in order to create these adaptive representations. The DSX algorithm had two major steps that we had to consider.

First, there was the problem of constructing an appropriate partition of the domain of the function we were representing. An error function that was defined in terms of scaled local Besov norms was used to determine this partition. Once a partition was constructed, we needed to calculate the actual projection. The iteration in the DSX algorithm is designed for this task, except that it is computationally difficult to work with. However, we saw that if we pick the proper quasi-interpolant, the iteration was greatly simplified.

Once we had the adaptive local scaling function representation, we considered different applications of this representation. We found that point evaluation was reasonably simple, and that if we had appropriate conditions on the partition, point evaluation became extremely simple. Once point evaluation was considered, we looked at numerical differentiation and numerical integration. A brief description of some of the classical applications of wavelet analysis was also given. As well, a few generalizations of the ideas discussed in this thesis were mentioned.

In summary, we have seen how to accurately create an adaptive representation of any appropriate function with little effort. The overall algorithm that we have described is self-sufficient in the sense that the algorithm determines everything from the grid to the projection. Many applications and generalizations regarding these representations exist, and thus there are a lot of areas ripe for further investigation.

Bibliography

[1] J.C. Feauveau A. Cohen, I. Daubechies. Biorthogonal bases of compactly supported wavelets. *Comm. Pure and Appl. Math.*, 45:485–560, 1992.

[2] B. Beckermann and G. Labahn. A fast and numerically stable euclidean-like algorithm for detecting relatively prime numerical polynomials, Jan 1998.

[3] K. Bittner and K. Urban. Adaptive wavelet methods using semiorthogonal spline wavelets: Sparse evaluation of nonlinear functions. Preprint, Universität Ulm, 2004.

[4] C.K. Chui. *An Introduction to Wavelets*. Academic Press, San Diego, 1992.

[5] Albert Cohen. *Numerical Analysis of Wavelet Methods*. North-Holland, 2003.

[6] John Conway. *A Course in Functional Analysis*. Springer, 1990.

[7] Wolfgang Dahmen, Reinhold Schneider, and Yuesheng Xu. Nonlinear functionals of wavelet expansions—adaptive reconstruction and fast evaluation. *Numer. Math.*, 86:49–101, 2000.

[8] Ingrid Daubechies. Orthonormal bases of compactly supported wavelets. *Comm. Pure and Appl. Math.*, 41:909–996, 1988.

[9] Ingrid Daubechies. *Ten Lectures on Wavelets*, volume 61 of *CBMS-NSF Refional Conference Series in Applied Mathematics*. Society for Industrial and Applied Mathematics, Philadelphia, 1992.

[10] Ingrid Daubechies and Wim Sweldens. Factoring wavelet transforms into lifting steps. Technical report, Bell Laboratories, Lucent Technologies, 1996.

[11] Jochen Fröhlich and Kai Schneider. An adaptive wavelet-vaguelette algorithm for the solution of PDEs. *Journal of Computational Physics*, 140(2):174–190, 1997.

[12] Charles F. Van Loan Gene H. Golub. *Matrix Computations*. The John Hopkins University Press, 1996.

[13] Christopher Jekeli. Spherical harmonic analysis, aliasing, and filtering. *Journal of Geodesy*, 70:214–223, 1996.

[14] Stephane G. Mallat. Multiresolution approximations and wavelet orthonormal bases of $L^2(\mathbf{R})$, September 1989.

[15] María Cristina Pereyra and Martin J. Mohlenkamp. Wavelets, their friends, and what they can do for you. http://www.math.ohiou.edu/ mjm/20044/PASIII/waveletPASIII.pdf, 2004.

[16] H. Resnikoff and Jr. R. Wells. *Wavelet Analysis. The Scalable Structure of Information*. Springer-Verlag New York Inc., 1998.

[17] Gilbert Strang. Wavelet transforms versus fourier tranforms. *Bulletin of the American Mathematical Society*, 28:208–305, April 1993.

[18] Wim Sweldens and Robert Piessens. Quadrature formulae and asymptotic error expansions for wavelet approximations of smooth functions. *SIAM J. Numer. Anal.*, 31(4):1240–1264, 1994.

[19] Pierre Verlinden and Ann Haegemans. An asymptotic expansion in wavelet analysis and its application to accurate numerical wavelet decomposition. *Numer. Algorithms*, 2(3-4):287–298, 1992.

[20] Antony Ware. Finite discrete projections onto wavelet spaces. Working paper.

[21] Antony Ware. Proving that discrete duals can be constructed ... Working paper.