HMC Senior Theses   HMC Student Scholarship

2008

Wavelets and Wavelet Sets

Sara Gussin
Harvey Mudd College

Recommended Citation

Gussin, Sara, "Wavelets and Wavelet Sets" (2008). *HMC Senior Theses*. 206.
https://scholarship.claremont.edu/hmc_theses/206

Wavelets and Wavelet Sets

Sara Gussin

Jon Jacobsen, Advisor

Darryl Yong, Reader

May, 2008

Abstract

Wavelets are functions that are useful for representing signals and approximating other functions. Wavelets sets are defined in terms of Fourier transforms of certain wavelet functions. In this paper, we provide an introduction to wavelets and wavelets sets, examine the preexisting literature on the subject, and investigate an algorithm for creating wavelet sets. This algorithm creates single wavelets, which can be used to create bases for $L^2(\mathbb{R}^n)$ through dilation and translation. We investigate the convergence properties of the algorithm, and implement the algorithm in Matlab.

Acknowledgments

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In many technical pursuits, it is necessary to have methods of approximating functions. Since the advent of Fourier analysis, such methods have been used in fields from signal processing to image compression. In accordance with this need for accurate and efficient approximation methods, mathematicians have continued investigating alternate ways of meeting the approximation needs of engineers and other applied scientists.

Although Fourier analysis is useful in many cases, it has its disadvantages. Fourier series are most effective at approximating periodic signals, but unfortunately are often inaccurate when used on non-periodic functions. Wavelet functions, however, often decay rapidly, if not vanish, outside a compact set, leading to more accurate approximations for many non-periodic functions.

There are many different wavelet functions, which can be constructed in various ways. Among the most famous are the Haar wavelets, named after Alfred Haar, and the Daubechies wavelets, named after Ingrid Daubechies. The Daubechies wavelets are designed to have compact support and to allow for very efficient compression of information (they have sparse coefficient matrices for many low-order polynomials). Other wavelets have been constructed by other methods, and have varied properties and appearances. Some are fractal-like in nature, while others have simpler geometry like the Shannon wavelet, which is a sinc function. This wide array of wavelets is helpful because different applications require different wavelet properties to maximize their efficiency and accuracy.

One recent method of wavelet construction is through the use of wavelet sets, which were introduced in [4]. These wavelets are characterized by the fact that they are the inverse Fourier transform of the characteristic function

of a set, known as a wavelet set. These wavelet sets are often fractal-like in nature. They also have to tile $\mathbb{R}^n$ by both scaling and translation, which gives them some interesting analytical properties. For this reason, wavelet sets are often studied for their own sake, not just for their applications in approximating functions with wavelets.

There are multiple methods for constructing wavelet sets, and in this paper we focus on the method set forth in [2]. After a background chapter, we provide an introduction to the algorithm itself, followed by an examination of the algorithm and its convergence first in one and then in several dimensions. Finally, we include a chapter which discusses implementation of the algorithm in Matlab.

For the interested reader who wants more information than is given in this thesis, there are a number of accessible papers in the literature that provide more detailed discussions of wavelet sets. The paper entitled "Wavelet Sets in $\mathbb{R}^n$" by Dai, Larson and Speegle [4] is the first paper on wavelet sets, in which their discovery and existence were announced and shown. Another paper by Benedetto and Leon, [3], is similar to the paper used in this thesis except in the case of multiple wavelets rather than single wavelets. The section on wavelet sets in Larson's chapter on "Unitary Systems and Wavelet Sets" in [5] provides a good in-depth technical introduction to wavelet sets. Finally, [9] is another good, relatively accessible, introduction to the theory of wavelet sets.

Chapter 2

Background

2.1 Classical Mathematics: Set Theory and Analysis

First, we review some basic set theory regarding intersections, unions and complements. Recall that if A, B and C are all sets and if $B \subseteq C$, then

$$A \cap B \subseteq A \cap C \quad \text{and} \quad A \cup B \subseteq A \cup C.$$

If $\overline{A}$ denotes the complement of A, De Morgan's laws state that

$$\overline{A \cap B} = \overline{A} \cup \overline{B} \quad \text{and} \quad \overline{A \cup B} = \overline{A} \cap \overline{B}.$$

Using induction, it is clear that these properties extends to countable unions and intersections of sets. In general in this paper, we will be taking complements relative to $\mathbb{R}^n$, but these properties apply regardless of the ambient space. We also know that intersections and unions distribute over each other. That is, if A, B and C are sets, then

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C) \quad \text{and} \quad A \cup (B \cap C) = (A \cup B) \cap (A \cup C).$$

These properties of unions, intersections and complements will be important in the convergence discussion later in this paper. Other important properties are those concerning infinite unions and intersections of sets. Recall that a countable union of sets in $\mathbb{R}^n$ must be nonempty as long as at least one of the sets is nonempty.

We will use the notation $A \setminus B$ to denote set subtraction, that is, $A \setminus B = A \cap \overline{B}$. Given an expression of the form

$$((A \setminus B_0) \setminus \ldots \setminus B_n) \setminus \ldots,$$

we know that the countable subtraction will be nonempty if

$$\mu(A) > \sum_{i=0}^{\infty} \mu(B_i),$$

where $\mu(X)$ denotes the measure of the set X.

These set theory ideas are important to the analysis regarding wavelet sets later in this paper. Before we can discuss wavelet sets themselves, however, we must review some important ideas from analysis.

The motivating factor behind wavelet analysis is the need to approximate functions using other functions. That is, we are trying to create bases for some particular function space. An important space for applications is the space $L^2(\Omega)$, for $\Omega \subseteq \mathbb{R}^n$, the set of all square integrable functions $f : \Omega \to \mathbb{R}$. Recall that $f \in L^2(\Omega)$ if

$$\int_{\Omega} f^2 \, dx < \infty.$$

It is also useful to recall the L^2 inner product, which is defined as the following.

Definition 2.1. *The L^2 inner product of two functions $f, g \in L^2(\Omega)$ is defined to be*

$$(f, g) = \int_{\Omega} f(x)g(x) \, dx.$$

We can use this inner product to find the coefficients of a function with respect to any orthogonal basis of $L^2(\Omega)$. Many such bases are possible, including the wavelet functions we will discuss later in this paper. The most commonly known, however, is the Fourier basis. The Fourier basis is a collection of sine and cosine functions of varying frequencies, together with $\frac{1}{2}$, which form an orthogonal basis for $L^2(-\pi, \pi)$. These functions are commonly used in signal analysis, as well as in the study of differential equations.

Nonetheless, there are downsides to the Fourier basis. Their approximations of periodic functions are generally very accurate, especially when those functions do not have any discontinuities. However, when it comes to approximations of nonperiodic functions, discontinuous functions, and highly nonlinear functions, Fourier functions are not always very accurate. In the case of discontinuities, we often encounter Gibbs phenomena, in which there is a significant overshoot in the approximation on a discontinuous function. Gibbs phenomena also have a large effect on the convergence of a Fourier series.

(a) Scaling Function ϕ (b) Wavelet Function ψ
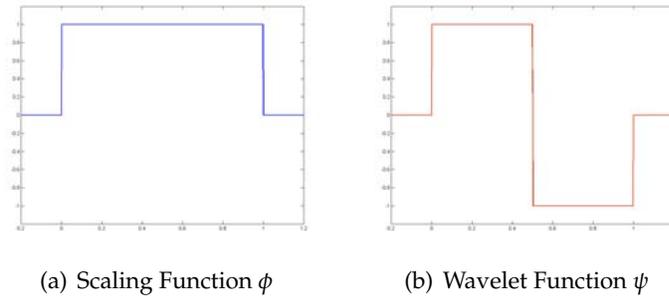
Figure 2.1: The Haar functions.

2.2   Introduction to Wavelets

To avoid many of these difficulties, we can use wavelet functions as basis functions instead of sinusoids. There are many different types of wavelet functions, whose dilates and translates form basis functions. To illustrate the dilation and translation transformations we will examine the Haar wavelet functions. There are two Haar functions, a scaling function and a wavelet function, both shown in Figure 2.1.

These two functions are known as the Haar scaling function, φ, and the Haar wavelet function, ψ. They are defined by

$$\phi(x) = \begin{cases} 1 & x \in (0,1) \\ 0 & \text{otherwise,} \end{cases}$$

and

$$\psi(x) = \begin{cases} 1 & x \in (0, \frac{1}{2}) \\ -1 & x \in (\frac{1}{2}, 1) \\ 0 & \text{otherwise.} \end{cases}$$

To create an orthonormal basis using the functions φ and ψ, we use two transformations, called T and D_a. T is a translation operator, which translates a function to the right by one, that is, $(Tf)(x) = f(x-1)$ for any function $f(x)$. The dilation operator D_a dilates a function by a factor $a > 1$, that is, $(D_a f)(x) = \frac{1}{\sqrt{a}} f(\frac{x}{a})$ and $(D_a)^{-1}(f)(x) = \sqrt{a} f(ax)$ for any function $f(x)$. We denote multiple compositions of an operator by

$$T^n = \underbrace{T \circ T \circ \ldots \circ T}_{n \text{ times}}.$$

(a) Unscaled ψ.   (b) Scaled $D_2^{-1}\psi$
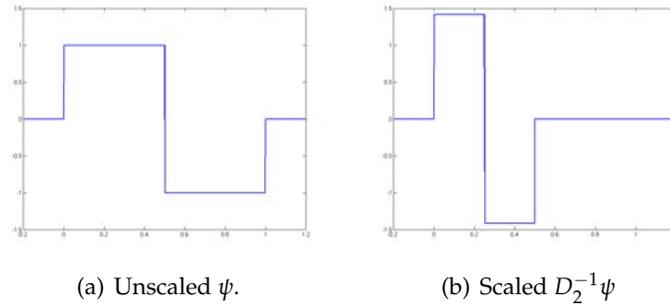
Figure 2.2: The Haar wavelet function ψ, and $D_2^{-1}\psi$.

We can combine these operations, T and D_a, as in [7].

$$D_a^m T^n \varphi(x) = \varphi_{m,n}(x) = \frac{1}{a^{m/2}} \varphi \left(\frac{x - a^m n}{a^m} \right).$$

It can be shown that these functions and transformations lead to an orthonormal basis for $L^2(\mathbb{R})$. For example, for any given m,

$$(\varphi_{m,j}, \varphi_{m,k}) = \delta_{j,k} = \left\{ \begin{array}{ll} 1 & j = k \\ 0 & j \neq k. \end{array} \right.$$

For most single variable wavelet functions, the most commonly used dilation factor is D_2, which scales by 2. For an example of this type of scaling, see Figure 2.2.

Using D_2 and T, then, we can define a dyadic orthonormal wavelet following [5].

Definition 2.2. *A* dyadic orthonormal wavelet *is a function $\psi \in L^2(\mathbb{R})$ such that the set*

$$\left\{ 2^{\frac{n}{2}} \psi \left(2^n t - l \right) : n, l \in \mathbb{Z} \right\}$$

forms an orthonormal basis for $L^2(\mathbb{R})$.

Note that the term "dyadic" refers to the fact that these wavelets use the scaling operator D_2, which scales by 2, not some other number.

Some other single dimensional wavelet functions include the Daubechies' wavelet functions and the Shannon wavelet. An example Daubechies' scaling and wavelet pair is shown in Figure 2.3, and the Shannon wavelet is shown in Figure 2.4. The Daubechies wavelets have many desirable properties, such as compact support (they vanish outside a compact, or finite,

(a) Scaling Function.   (b) Wavelet Function.
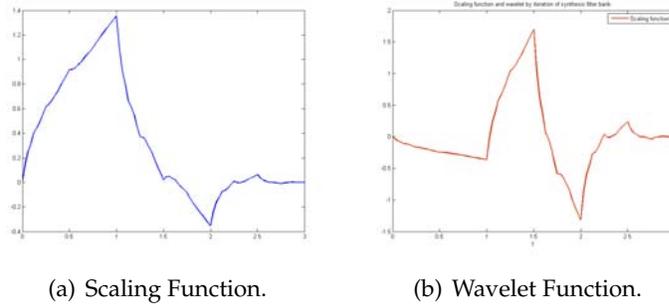
Figure 2.3: The Daubechies 4-tap scaling and wavelet functions [8].
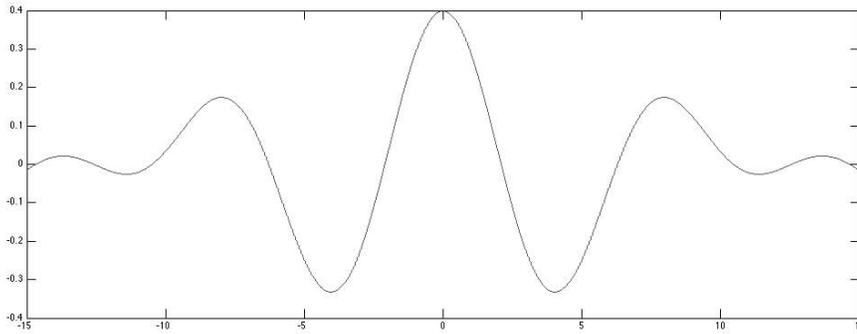
Figure 2.4: The Shannon wavelet function.

interval). Also, their inner products with low-order polynomial functions are zero, making them ideal for data compression applications.

The Haar and Daubechies wavelets are cases in which two functions are scaled and translated to create orthogonal bases, and these are known as the scaling and wavelet functions. The Shannon wavelet, however, requires only one function (using, of course, the D and T operators) to create the orthogonal basis. Thus, it is known as a single wavelet, rather than a multiple wavelet. All the wavelets we will be examining in this paper will be single wavelets, like the Shannon wavelet. Furthermore, the Shannon wavelet has an associated wavelet set, like the wavelets we will discuss in this paper [6]. The wavelet set for the Shannon wavelet is:

$$\left[-1, -\frac{1}{2}\right) \cup \left(\frac{1}{2}, 1\right].$$

Before we discuss the relationship between wavelet sets and wavelet functions, however, we must briefly discuss multidimensional wavelets. Multidimensional wavelets work in much the same way as single dimensional wavelets. One major difference, however, arises in the dilation operator. In multiple dimensions, we can use a scalar dilation factor, or we can use a dilation matrix A. We require A to be expansive, as defined in [5].

Definition 2.3. *We call a matrix A* expansive *if $|\lambda| > 1$ for all eigenvalues λ of A.*

In this case, the definition of a wavelet function is as follows [5, 2]:

Definition 2.4. *Let $1 \leq m < \infty$, and let A be an expansive $n \times n$ matrix. Consider $\{(\psi_j^A)_{m,n} : m \in \mathbb{Z}, n \in \mathbb{Z}^d\}$ where $j = 1, \ldots, J$ for some $J \in \mathbb{Z}$ and $\psi_{m,n}^A$ is defined to be:*

$$\left(\psi_j^A\right)_{m,n}(x) = |det(A)|^{\frac{m}{2}} \psi_j(A^m x - n), \ m \in \mathbb{Z}, \ n \in \mathbb{Z}^d.$$

If $\{(\psi_j^A)_{m,n}\}$ form an orthonormal basis for $L^2(\mathbb{R}^d)$, then the ψ are referred to as multiple dilation-A orthonormal wavelets.

If we take $A = \alpha I$, where α is a scalar such that $\alpha > 1$ and I is the identity matrix, the scaling works just as it did in the single variable case.

2.3 Wavelet Sets

Now that we have introduced the concepts of wavelet functions, we can discuss wavelet sets. For this discussion, first recall the Fourier transform as defined in [5].

Definition 2.5. *If $f, g \in L^1(\mathbb{R}) \cap L^2(\mathbb{R})$, then the* Fourier transform *$\mathcal{F}$ of f is defined to be*

$$(\mathcal{F}f)(s) = \widehat{f}(s) := \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} e^{-ist} f(t) \, dt,$$

and the inverse Fourier transform *of g is*

$$(\mathcal{F}^{-1}g)(t) = \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} e^{ist} g(s) \, ds.$$

Note that multiple normalizations exist for the Fourier transform. We use this one because its complement is its inverse—that is, it is a unitary transformation [5].

Recall the characteristic function of a set $E \subseteq \mathbb{R}^d$ is defined by:

$$\chi_E(x) = \begin{cases} 1 & x \in E, \\ 0 & x \notin E. \end{cases}$$

Note that the Haar scaling function is χ_E for $E = [0,1]$. We can now state the definition of a wavelet set [9]:

Definition 2.6. *A* wavelet set *is a measurable subset E of $\mathbb{R}^n$ such that the inverse Fourier transform of χ_E is an orthonormal wavelet in $L^2(\mathbb{R}^n)$.*

In other words, a set E is a wavelet set if dilations and translations of its inverse Fourier transform yield an orthonormal basis for $L^2(\mathbb{R}^n)$. This is a very strict condition. It is also important to note that not all wavelets are associated with a wavelet set. In order for a wavelet to have a wavelet set, its Fourier transform must be both real and equal to either 1 or 0 at all points. Now, we will examine a characterization of wavelet sets which will assist us in understanding the examples later in this paper. First, however, we need some more terminology. Recall that a partition of a set A is a collection of disjoint subsets A_i such that their union is A. Then we define τ-congruence as [2]:

Definition 2.7. *If A and B are subsets of $\mathbb{R}^n$, then we say A is τ-congruent to B (denoted $A \overset{\tau}{\cong} B$) if there are countable partitions $\{A_p : p \in \mathbb{Z}\}$ and $\{B_p : p \in \mathbb{Z}\}$ of A and B respectively, and a sequence $\{n_p : p \in \mathbb{Z}\} \subseteq \mathbb{Z}^n$ such that for all $p \in \mathbb{Z}$, $A_p = B_p + n_p$.*

As an example of this, consider the set depicted in Figure 2.5. Section A has been moved to section A' via the translation $f(x,y) = (x+1,y)$, and section B has been moved to B' via the translation $f(x,y) = (x,y+1)$. Therefore, by our definition of τ-congruence, the gray shaded set is τ-congruent to the square $[-\frac{1}{2}, \frac{1}{2}]^2$. In general, these partitions need not be rectangular.

Benedetto and Leon [2] proved the following theorem characterizing all wavelet sets in $\mathbb{R}^n$:

Theorem 2.1. *Let $K \subseteq \mathbb{R}^n$ be a measurable set. Then K is a wavelet set if and only if the following two conditions hold:*

   1. $\{K + i : i \in \mathbb{Z}^n\}$ is a partition of $\mathbb{R}^n$.
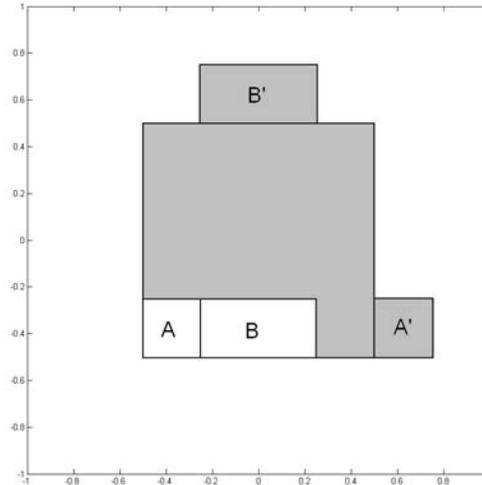
Figure 2.5: An example of τ-congruence

2. $\{2^j K \ : \ j \in \mathbb{Z}\}$ *is a partition of* $\mathbb{R}^n$.

That is, a set K is a wavelet set if and only if K tiles $\mathbb{R}^n$ by both tiling (from condition 1) and scaling (condition 2). Furthermore, it can be shown that the first condition of Theorem 2.1 holds if and only if $K \overset{\tau}{\cong} [-\frac{1}{2}, \frac{1}{2}]^n$. This characterization of wavelet sets will come into play when we examine our two-dimensional example.

Now that we have discussed wavelets and wavelet sets, we can begin examining examples. First, we examine a one-dimensional example, and give several illustrations of wavelet sets and functions derived from that construction. Then, we look at a general example, examine the construction in detail, and give examples of wavelet sets in one and two dimensions. We then examine the convergence properties of the construction procedure in both one and multiple dimensions.

2.4   A Short One-Dimensional Example

This example is taken from [1]. It consists of constructing wavelet sets of the form

$$W_n = \left[-2^{n+1}\pi - \alpha_n, -2^{n+1}\pi \right) \cup \left[-\pi, -\alpha_n \right)$$

$$\cup [\alpha_n, \pi) \cup \left[2^{n+1}\pi, 2^{n+1}\pi + \alpha_n \right),$$

where $\alpha_n = \frac{2^{n+1}\pi}{2^{n+2}-1}$.

To find the wavelet functions associated with these wavelet sets, we need to take the inverse Fourier transform of their characteristic functions. Since the characteristic function of a set is one on the set and zero elsewhere, we know that

$$(\mathcal{F}^{-1}\chi_{W_n}) = \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} e^{ixt} \chi_{W_n}(t) \, dt$$

$$= \frac{1}{\sqrt{2\pi}} \int_{W_n} e^{ixt} \, dt.$$

Therefore, by our definition of W_n,

$$F_n = (\mathcal{F}^{-1}\chi_{W_n})$$

$$= \frac{1}{\sqrt{2\pi}} \left[\int_{-2^{n+1}\pi-\alpha_n}^{-2^{n+1}\pi} e^{ixt} \, dt + \int_{-\pi}^{-\alpha_n} e^{ixt} \, dt \right.$$

$$\left. + \int_{\alpha_n}^{\pi} e^{ixt} \, dt + \int_{2^{n+1}\pi}^{2^{n+1}\pi+\alpha_n} e^{ixt} \, dt \right]$$

$$= \frac{1}{ix\sqrt{2\pi}} \left[e^{-(2^{n+1}\pi)ix} + e^{(-2^{n+1}\pi-\alpha_n)ix} + e^{-i\alpha_n x} - e^{-i\pi x} \right.$$

$$\left. + e^{i\pi x} - e^{i\alpha_n x} + e^{(2^{n+1}\pi+\alpha_n)ix} - e^{(2^{n+1}\pi)ix} \right]$$

$$= \frac{\sqrt{2}}{x\sqrt{\pi}} \left[\sin((2^{n+1}\pi + \alpha_n)x) - \sin(2^{n+1}\pi x) \right.$$

$$\left. + \sin(\pi x) - \sin(\alpha_n x) \right].$$

Substituting $\alpha_n = \frac{2^{n+1}\pi}{2^{n+2}-1}$,

$$F_n = \frac{\sqrt{2}}{x\sqrt{\pi}} \left[\sin\left(\left(2^{n+1}\pi + \frac{2^{n+1}\pi}{2^{n+2}-1}\right)x\right) - \sin\left(2^{n+1}x\right) \right.$$
$$\left. + \sin(\pi x) - \sin\left(\frac{2^{n+1}\pi x}{2^{n+2}-1}\right) \right].$$

Figures 2.6 through 2.11 are the wavelet sets and wavelet functions for the $n = 0, 1, 2$ cases of the above example. These are all single wavelets, which means no other functions are required to create bases for $L^2(\mathbb{R}^n)$. Using translation and dilation on these functions alone will create the basis functions.
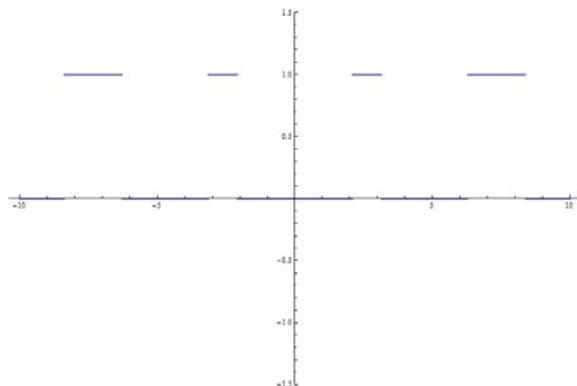
Figure 2.6: Characteristic function of the wavelet set for $n = 0$:
$$\left\{2\pi < |x| \leq \tfrac{8\pi}{3}\right\} \bigcup \left\{\tfrac{2\pi}{3} < |x| \leq \pi\right\}$$
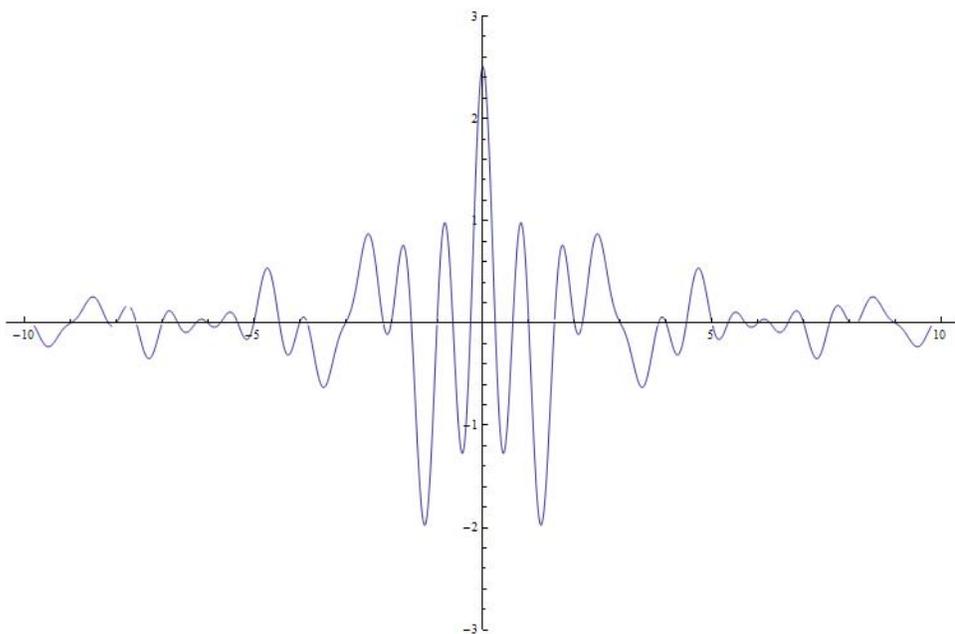
Figure 2.7: Wavelet function for $n = 0$:
$$F_0(x) = \frac{\sqrt{2}}{x\sqrt{\pi}}\left[\sin\left(\tfrac{8\pi}{3}x\right) - \sin(2\pi x) + \sin(\pi x) - \sin\left(\tfrac{2\pi}{3}x\right)\right]$$
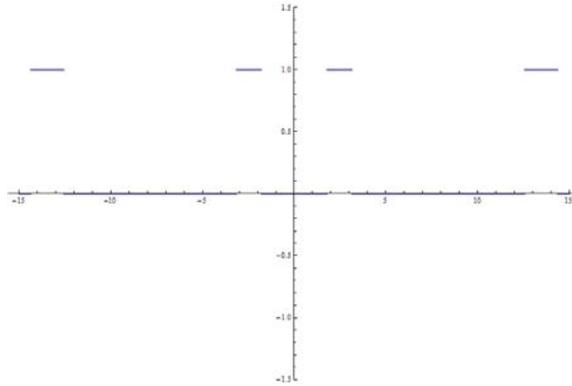
Figure 2.8: Characteristic function of the wavelet set for $n = 1$:
$$\left\{ 4\pi < |x| \leq \tfrac{32\pi}{7} \right\} \cup \left\{ \tfrac{4\pi}{7} < |x| \leq \pi \right\}$$
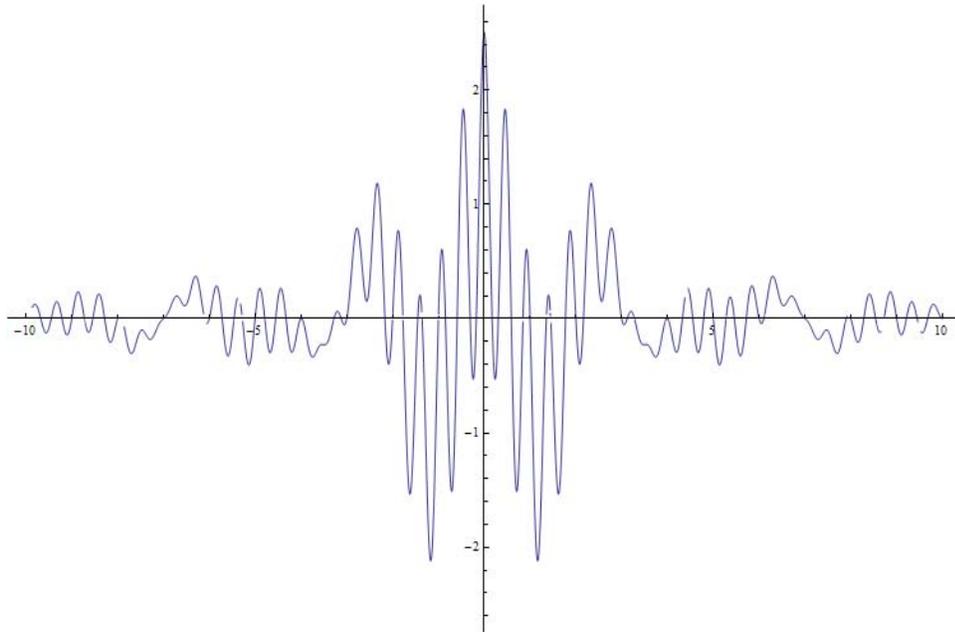
Figure 2.9: Wavelet function for $n = 1$:
$$F_0(x) = \frac{\sqrt{2}}{x\sqrt{\pi}} \left[\sin\left(\tfrac{32\pi}{7}x\right) - \sin(4\pi x) + \sin(\pi x) - \sin\left(\tfrac{4\pi}{7}x\right) \right]$$
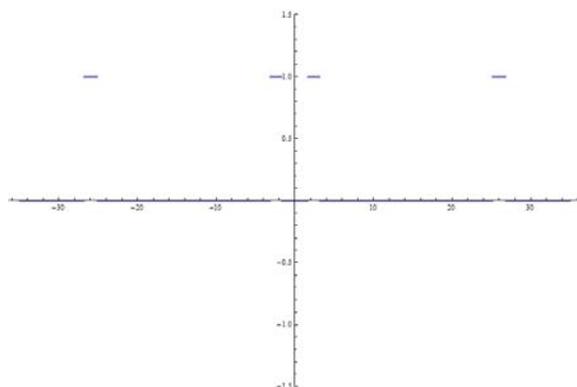
Figure 2.10: Characteristic function of the wavelet set for $n = 2$:
$$\left\{8\pi < |x| \leq \tfrac{128\pi}{15}\right\} \cup \left\{\tfrac{8\pi}{15} < |x| \leq \pi\right\}$$
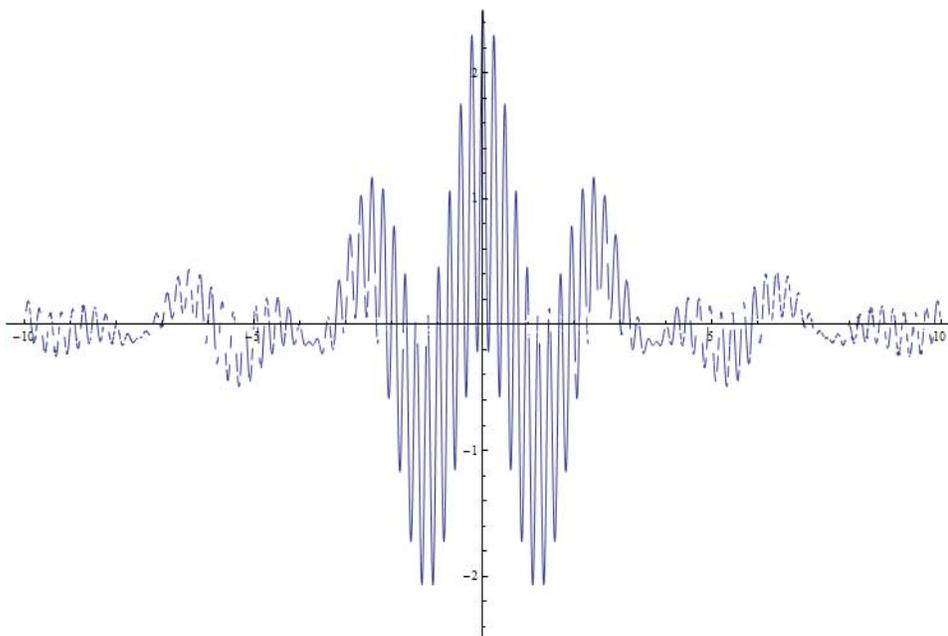
Figure 2.11: Wavelet function for $n = 2$:
$$F_0(x) = \frac{\sqrt{2}}{x\sqrt{\pi}}\left[\sin\left(\tfrac{128\pi}{15}x\right) - \sin(8\pi x) + \sin(\pi x) - \sin\left(\tfrac{8\pi}{15}x\right)\right]$$

Chapter 3

A Wavelet Set Algorithm

3.1 Introduction to the Algorithm

In this section, we will explain the process used in [2] to construct wavelet sets. We will begin by examining the general procedure, and then consider a specific example of one of this class of wavelet sets and how it is constructed.

We begin with an arbitrary set $K_0 \subseteq [-N, N]^d \subseteq \mathbb{R}^d$. We require this set to be a neighborhood of the origin (that is, it must contain some open ball around the origin), and to have measure 1 (i.e., length one in $\mathbb{R}$, area one in $\mathbb{R}^2$, etc.). It must also be τ-congruent to $[-\frac{1}{2}, \frac{1}{2}]^d$. Recall from Definition 2.7 that the τ-congruence of two sets A and B requires that there exist countable partitions A_i and B_i and a piecewise-defined translation $f_i(B_i) = B_i + n_i = A_i$, where $n_i \in \mathbb{Z}^d$ for each i.

In all the specific examples we consider in this paper, we will take K_0 to be $[-\frac{1}{2}, \frac{1}{2}]^d$. However, this is not necessary for the construction to yield wavelet sets. As shown in [2], other sets can be used for K_0 as long as they meet the conditions set forth in the previous paragraph.

Next, we define a transformation T which is defined on K_0 such that $T : K_0 \longrightarrow [-2N, 2N]^d \setminus [-N, N]^d$. We further stipulate that, for any given $\xi \in K_0$, $T(\xi) = \xi + k_\xi$ for some $k_\xi \in \mathbb{Z}^d$. Thus, we can think of T as a piecewise-defined linear transformation. To prove that this process results in a wavelet set, it is important to note that T must be a measure preserving mapping.

Now, to construct our wavelet set through an iterative process, first

$$A_0 = K_0 \cap \left(\bigcup_{j \geq 1} 2^{-j} K_0 \right) .$$

That is, A_0 is the intersection of K_0 with the union of all half-scalings of it-self. Now, we know that T can act on A_0, since $T : [-N, N]^d \longrightarrow [-2N, 2N]^d \setminus [-N, N]^d$ and $A_0 \subseteq K_0 \subseteq [-N, N]^d$. Thus, we proceed by defining

$$K_1 = (K_0 \setminus A_0) \bigcup T A_0 .$$

This set is the first in our sequence K_n, the limit of which is our wavelet set K. We continue the process by defining

$$A_1 = K_1 \cap \left(\bigcup_{j \geq 1} 2^{-j} K_1 \right) .$$

Again, we know that $A_1 \subseteq [-N, N]^d$, so T can act on A_1. Thus we define

$$K_2 = ((K_0 \setminus A_0) \setminus A_1) \bigcup T A_0 \bigcup T A_1 .$$

Continuing this process iteratively leads to the general definition of A_n as

$$A_n = K_n \cap \left(\bigcup_{j \geq 1} 2^{-j} K_n \right) ,$$

and of K_{n+1} as

$$K_{n+1} = ((((K_0 \setminus A_0) \setminus A_1) \dots) \setminus A_n) \bigcup \left(T A_0 \bigcup T A_1 \dots \bigcup T A_n \right) .$$

From our construction, it can be shown that the limiting set K must be τ-congruent to our original set K_0. Thus, we know that K is τ-congruent to $[-\frac{1}{2}, \frac{1}{2}]^d$.

As we saw in Theorem 2.1, in order for a set to be a wavelet set, it must form partitions of $\mathbb{R}^d$ both by translation and by scaling. When a set is τ-congruent to $[-\frac{1}{2}, \frac{1}{2}]^d$ it must also form a partition of $\mathbb{R}^d$ by translation. A full proof that K is a wavelet set based on Theorem 2.1 is provided in [2].

3.2   An Example of the Process At Work in $\mathbb{R}^2$

To illustrate the construction in $\mathbb{R}^2$, consider $K_0 = [-\frac{1}{2}, \frac{1}{2}]^2$. By setting $A_1 = B_1 = [-\frac{1}{2}, \frac{1}{2}]^2$ and allowing $n_1 = 0$, we can see that K_0 is indeed τ-congruent to $[-\frac{1}{2}, \frac{1}{2}]^2$ (see Figure 3.1). Note that for all these diagrams, the sets are depicted by the black areas in the diagrams.
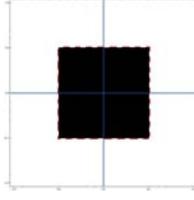
Figure 3.1: An example K_0.

In this example, we take T to be

$$
T(x_1, x_2) = \begin{cases}
(x_1 - 2N, x_2 - 2N) & x_1 \in [0, N], \ x_2 \in [0, N] \\
(x_1 + 2N, x_2 - 2N) & x_1 \in [-N, 0], \ x_2 \in [0, N] \\
(x_1 + 2N, x_2 + 2N) & x_1 \in [-N, 0], \ x_2 \in [-N, 0] \\
(x_1 - 2N, x_2 + 2N) & x_1 \in [0, N], \ x_2 \in [-N, 0].
\end{cases}
$$

Now, we begin our construction for this T by calculating A_0. In this case, note that

$$
\bigcup_{i=1}^{\infty} 2^{-i} K_0 = \bigcup_{i=1}^{\infty} \left[-\frac{1}{2^{i+1}}, \frac{1}{2^{i+1}} \right]^2 = \left[-\frac{1}{4}, \frac{1}{4} \right]^2.
$$

Therefore, since $A_0 = K_0 \cup (\bigcap_{i=1}^{\infty} 2^{-i} K_0)$, we can see that in this case $A_0 = [-\frac{1}{4}, \frac{1}{4}]^2$, and this set is shown in Figure 3.2(a). Recall

$$
K_1 = (K_0 \setminus A_0) \bigcup T A_0,
$$

and based on our calculations of A_0 and K_0, it follows that K_1 is as shown in Figure 3.2(b).

Now we examine $A_1 = K_1 \cap \left(\bigcup_{j \geq 1} 2^{-j} K_1 \right)$. This is more complicated than A_0 was, but in this case, A_1 is shown in Figure 3.2(c). Using the T we examined previously, we can then see that K_2 is as shown in Figure 3.2(d). Continuing this process iteratively will generate a wavelet set K which looks approximately like the sixth iteration of the process, as shown in Figure 3.3.

(a) A_0 (b) K_1 (c) A_1 (d) K_2
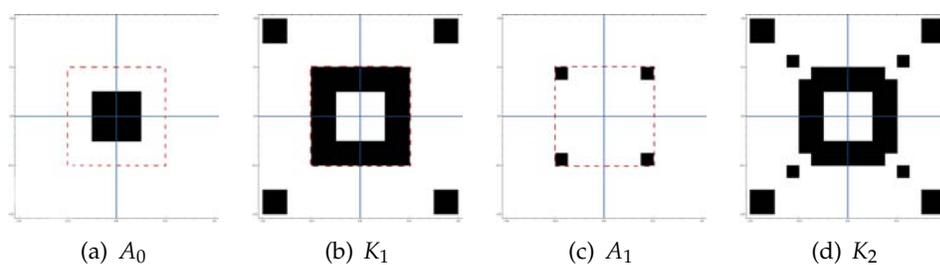
Figure 3.2: Illustrations of the Algorithm.
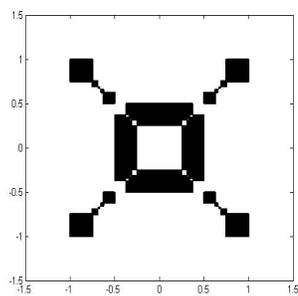
Figure 3.3: K_6.
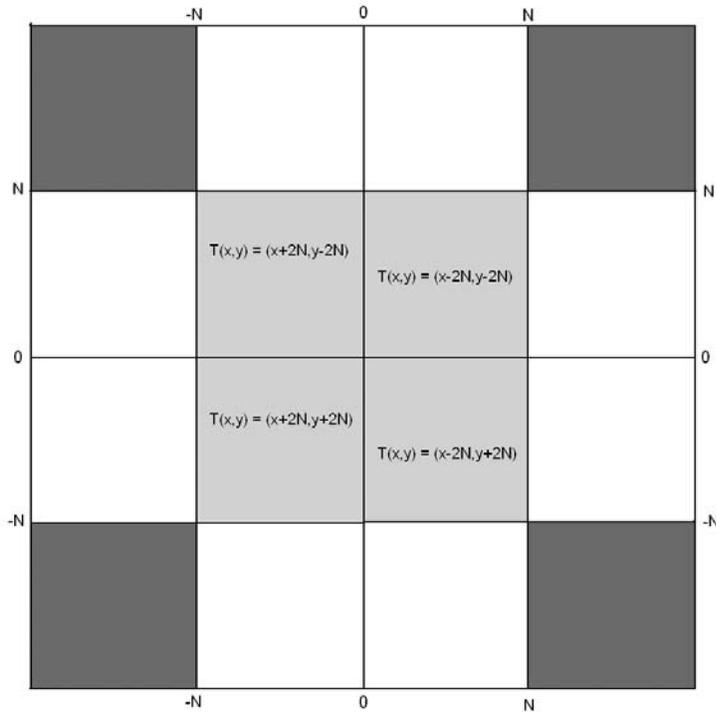
Figure 3.4: Illustration of T_1.

3.3   More Illustrations in Two Dimensions

In this section, we will illustrate three different T mappings and the sequence of K sets (K_1, K_2, K_3 and K_6) pertaining to each one. The first T map is T_1, illustrated in Figure 3.4, which we used in our example in the previous section.

The mapping T_2, shown in Figure 3.6, is similar to T_1 in that it partitions $[-N, N]^2$ into squares as well. It translates those squares differently, however, and the resulting wavelet set inherits this dynamic.

Our last example construction is with T_3, which is illustrated in Figure 3.8. This T partitions the unit square into triangles which are then translated. An interesting property of this particular map is that the limiting wavelet set will be connected, unlike the previous two cases.

In the next section we consider the convergence properties of this algorithm. We will begin by examining the algorithm and its convergence in one dimension. Then we will offer an examination of the algorithm in multiple dimensions.

(a) K_1.
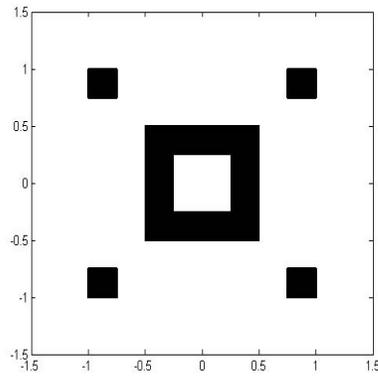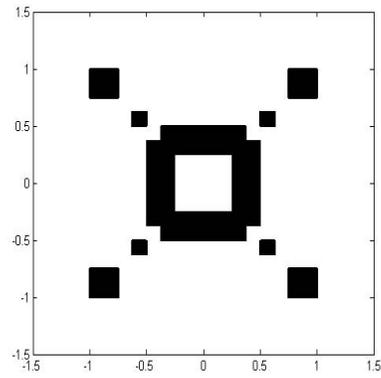
(b) K_2.
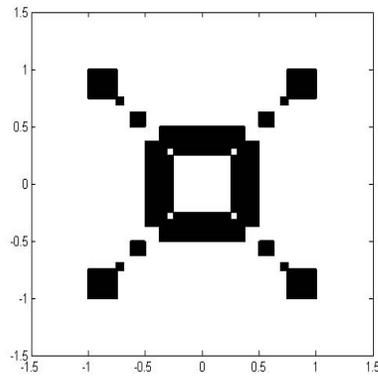
(c) K_3.
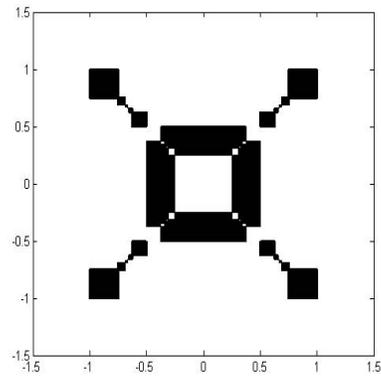
(d) K_6.

Figure 3.5: Some K_n sets for T_1.
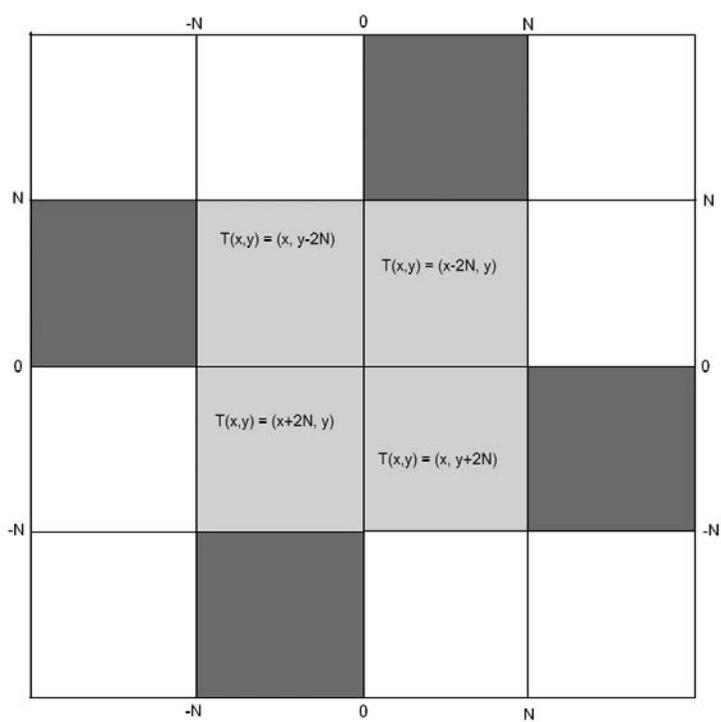
Figure 3.6: Illustration of T_2.

(a) K_1.
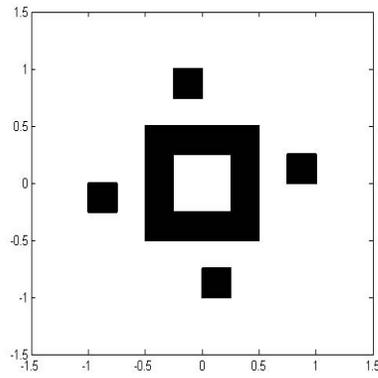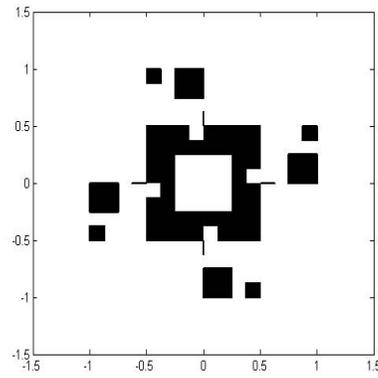
(b) K_2.
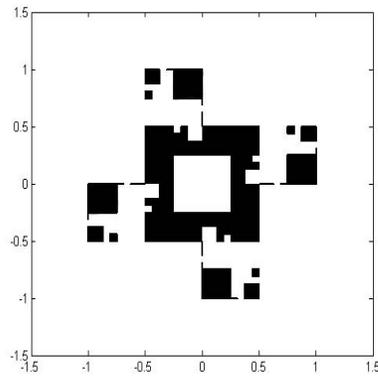
(c) K_3.
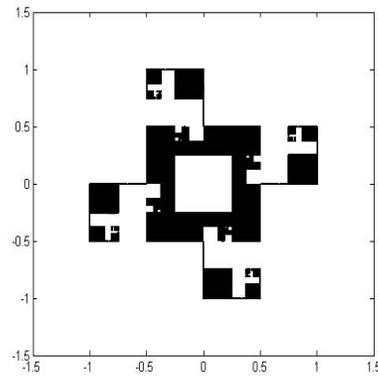
(d) K_6.

Figure 3.7: Some K_n sets for T_2.
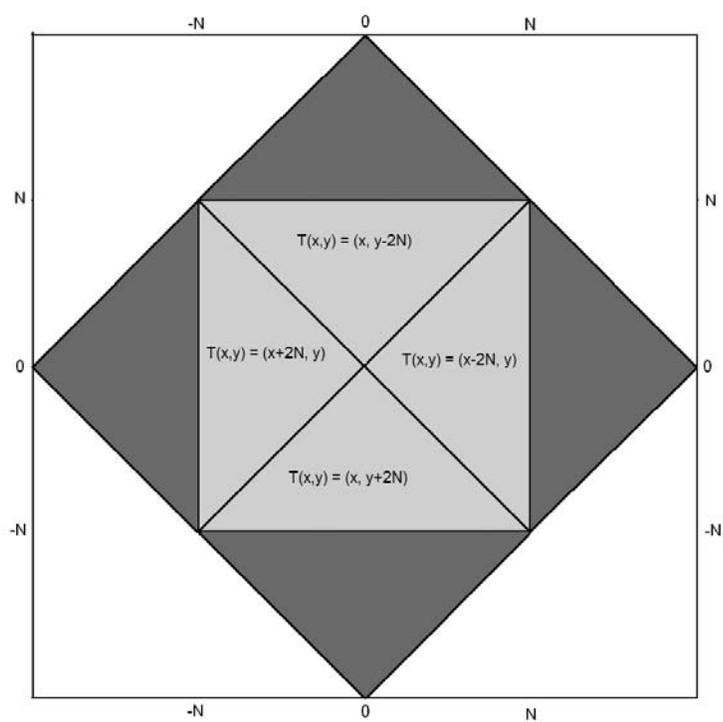
Figure 3.8: Illustration of T_3.

(a) K_1.
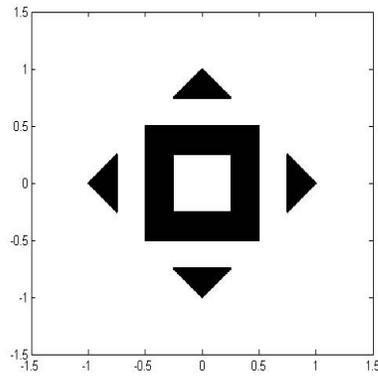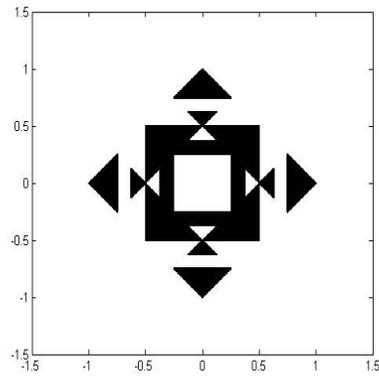
(b) K_2.
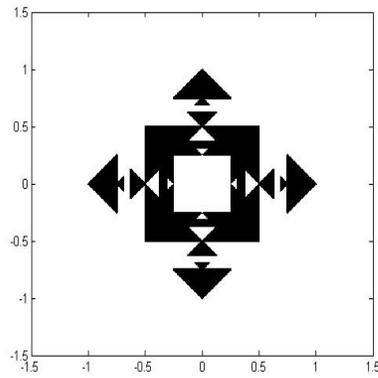
(c) K_3.
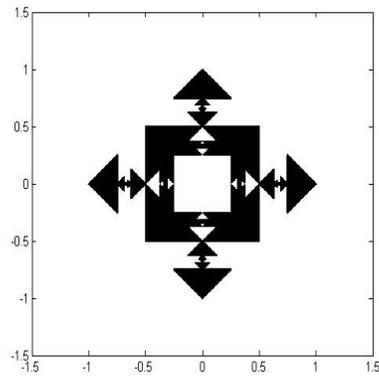
(d) K_6.

Figure 3.9: K_n sets for T_3.

Chapter 4

Convergence in One Dimension

4.1 Why Convergence Matters

The convergence of the wavelet set algorithm is important for multiple reasons. From a practical standpoint, convergence helps us determine how useful this method would be in constructing wavelets for use in signal processing applications. To construct a wavelet function by this method, we must first use the method to approximate the wavelet set, and then approximate the inverse Fourier transform of that set. As such, the resulting wavelets will not be a perfect orthonormal basis for $L^2(\mathbb{R}^n)$. By examining the convergence of this algorithm, we can get an idea of how close the resulting function will be to the wavelet function we desire.

From a more theoretical standpoint, examining the convergence of this algorithm helps us understand the algorithm itself more fully. Especially when we examine the algorithm in multiple dimensions, we will see some interesting simplifications in the algorithm which become apparent through our examination of its convergence. Even in one dimension, by examining convergence of the algorithm, we acquire a better understanding of how it works and its significance. We also examine convergence of this algorithm to become more comfortable with the definitions and terminology used, as well as out of sheer enjoyment of the mathematics involved.

This chapter extends the work on the algorithm presented in [2] by examining the convergence of the algorithm in the one-dimensional case.

4.2   The Algorithm in One Dimension

In one dimension, we consider the case where $K_0 = [-\frac{1}{2}, \frac{1}{2}]$. In this case, our T transformation is

$$T(x) = \begin{cases} x - 1, & x > 0 \\ x + 1, & x < 0. \end{cases}$$

We can easily see that T maps K_0 to $[-1, 1] \setminus [-\frac{1}{2}, \frac{1}{2}]$, as well as that it conforms to the other properties of such transformations mentioned in the previous chapter.

We can now begin to follow the algorithm. Clearly, $A_0 = [-\frac{1}{4}, \frac{1}{4}]$ and $TA_0 = [-1, -\frac{3}{4}] \cup [\frac{3}{4}, 1]$. Thus, we know that

$$K_1 = \left[-1, -\frac{3}{4}\right] \cup \left[-\frac{1}{2}, -\frac{1}{4}\right] \cup \left[\frac{1}{4}, \frac{1}{2}\right] \cup \left[\frac{3}{4}, 1\right].$$

Continuing this process we find

$$K_{2n} = \left\{ 1 \geq |x| \geq \frac{3}{4} - \sum_{j=1}^{n-1} 4^{-(j+1)} \right\} \cup$$

$$\left\{ \frac{1}{2} + \sum_{j=1}^{n} 2^{-(2j+1)} \geq |x| \geq \frac{1}{2} \right\} \cup$$

$$\left\{ \sum_{j=1}^{n} 2^{-(2j-1)} - \frac{1}{2} \geq |x| \geq \sum_{j=1}^{n} 4^{-n} \right\}$$

and

$$K_{2n-1} = \left\{ 1 \geq |x| \geq \frac{3}{4} - \sum_{j=1}^{n-1} 4^{-(j+1)} \right\} \cup$$

$$\left\{ \frac{1}{2} + \sum_{j=1}^{n-1} 2^{-(2j+1)} \geq |x| \geq \frac{1}{2} \right\} \cup$$

$$\left\{ \sum_{j=1}^{n-1} 2^{-(2j+1)} - \frac{1}{2} \geq |x| \geq \sum_{j=1}^{n} 4^{-n} \right\}.$$

Using the formula for geometric series, this simplifies to

$$K_{2n} = \left\{ 1 \geq |x| \geq \frac{2 + 4^{-(n-1)}}{3} \right\} \bigcup$$

$$\left\{ \frac{2 - 2^{-(2n+3)}}{3} \geq |x| \geq \frac{1}{2} \right\} \bigcup$$

$$\left\{ \frac{1 + 2^{-(2n+3)}}{3} \geq |x| \geq \frac{1 - 4^{-(n-1)}}{3} \right\}$$

and

$$K_{2n-1} = \left\{ 1 \geq |x| \geq \frac{2 + 4^{-(n-1)}}{3} \right\} \bigcup$$

$$\left\{ \frac{2 - 2^{-(2n+5)}}{3} \geq |x| \geq \frac{1}{2} \right\} \bigcup$$

$$\left\{ \frac{1 + 2^{-(2n+5)}}{3} \geq |x| \geq \frac{1 - 4^{-(n-1)}}{3} \right\}.$$

Taking the limits as n approaches infinity of these two expressions, we have

$$\lim_{n \to \infty} K_{2n} = \left\{ 1 \geq |x| \geq \frac{1}{2} \right\}$$

and

$$\lim_{n \to \infty} K_{2n-1} = \left\{ 1 \geq |x| \geq \frac{1}{2} \right\}.$$

This is good, since the process is designed to converge to one set. Thus, we know that our limiting wavelet set has the form

$$K = \left[-1, -\frac{1}{2} \right] \bigcup \left[\frac{1}{2}, 1 \right].$$

Note that this is the wavelet set for the Shannon wavelet we used as an example in Chapter 2.

However, while a convergence analysis on the sets themselves would be interesting, the most useful information is a convergence analysis on the intermediary functions, the limit of which will be our wavelet function.

To do this, we must take the inverse Fourier transforms of the sets above. Recall that the inverse Fourier transform of a function f is defined to be

$$(\mathcal{F}^{-1}f)(t) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(s) e^{ist} \, ds.$$

In this case, we are examining the inverse Fourier transforms of the characteristic functions of the K_n sets. Thus, our inverse Fourier transforms are merely

$$(\mathcal{F}^{-1}\chi_{K_n})(t) = \frac{1}{\sqrt{2\pi}} \int_{K_n} e^{ist} \, ds.$$

Furthermore, all our sets are symmetric with respect to the y-axis. This means that our functions will all be real, because

$$\int_{K_n} e^{ist} \, ds = \int_{K_n} (\cos(st) + i\sin(st)) \, ds.$$

Since $\sin(ts)$ is odd in s and K_n is symmetric about the y-axis, we know that the integral of $\sin ts$ over K_n must be zero. Therefore,

$$(\mathcal{F}^{-1}\chi_{K_n})(t) = \frac{1}{\sqrt{2\pi}} \int_{K_n} \cos(st) \, ds.$$

As an example, we compute in detail, without using the shortcut above, the inverse Fourier transform of the even sets. A similar examination of the odd sets would be tedious and redundant given their similarity to the even sets. Consider the even sets

$$K_{2n} = \left\{ 1 \geq |x| \geq \frac{2 + 4^{-(n-1)}}{3} \right\} \bigcup$$

$$\left\{ \frac{2 - 2^{-(2n+3)}}{3} \geq |x| \geq \frac{1}{2} \right\} \bigcup$$

$$\left\{ \frac{1 + 2^{-(2n+3)}}{3} \geq |x| \geq \frac{1 - 4^{-(n-1)}}{3} \right\}.$$

Now, we define the inverse Fourier transforms of the sets to be

$$
\begin{aligned}
F_{2n} &= \mathcal{F}^{-1}\left(\chi_{K_{2n}}(t)\right) \\
&= \int_{K_{2n}} \frac{1}{\sqrt{2\pi}} e^{ixt}\, dt \\
&= \frac{1}{\sqrt{2\pi}} \left[\int_{\frac{2+4^{-(n-1)}}{3}}^{1} e^{ixt}\, dx + \int_{-1}^{-\frac{2+4^{-(n-1)}}{3}} e^{ixt}\, dx \right. \\
&\qquad + \int_{\frac{1}{2}}^{\frac{2-2^{-(2n+3)}}{3}} e^{ixt}\, dx + \int_{-\frac{2-2^{-(2n+3)}}{3}}^{-\frac{1}{2}} e^{ixt}\, dx \\
&\qquad \left. + \int_{\frac{1-4^{-(n-1)}}{3}}^{\frac{1+2^{-(2n+3)}}{3}} e^{ixt}\, dx + \int_{-\frac{1+2^{-(2n+3)}}{3}}^{-\frac{1-4^{-(n-1)}}{3}} e^{ixt}\, dx \right] \\
&= \frac{1}{ix\sqrt{2\pi}} \left[e^{ix} - e^{ix\frac{2+4^{-(n-1)}}{3}} + e^{-ix\frac{2+4^{-(n-1)}}{3}} - e^{-ix} \right. \\
&\qquad + e^{ix\frac{2-2^{-(2n+3)}}{3}} - e^{ix\frac{1}{2}} + e^{-ix\frac{1}{2}} - e^{-ix\frac{2-2^{-(2n+3)}}{3}} \\
&\qquad \left. + e^{ix\frac{1+2^{-(2n+3)}}{3}} - e^{ix\frac{1-4^{-(n-1)}}{3}} + e^{-ix\frac{1-4^{-(n-1)}}{3}} - e^{-ix\frac{1+2^{-(2n+3)}}{3}} \right].
\end{aligned}
$$

However, we know that $e^{it} - e^{-it} = 2it\sin t$. Therefore, we know that

$$
\begin{aligned}
F_{2n} = \frac{\sqrt{2}}{x\sqrt{\pi}} \left[\sin(x) - \sin\left(\frac{2+4^{-(n-1)}}{3}x\right) + \sin\left(\frac{2-2^{-(2n+3)}}{3}x\right) \right. \\
\left. - \sin\left(\frac{1}{2}x\right) + \sin\left(\frac{1+2^{-(2n+3)}}{3}x\right) - \sin\left(\frac{1-4^{-(n-1)}}{3}x\right) \right].
\end{aligned}
$$

A similar analysis of the odd sets shows us that

$$
\begin{aligned}
F_{2n-1} = \frac{\sqrt{2}}{x\sqrt{\pi}} \left[\sin(x) - \sin\left(\frac{2+4^{-(n-1)}}{3}x\right) + \sin\left(\frac{2-2^{-(2n+5)}}{3}x\right) \right. \\
\left. - \sin\left(\frac{1}{2}x\right) + \sin\left(\frac{1+2^{-(2n+5)}}{3}x\right) - \sin\left(\frac{1-4^{-(n-1)}}{3}x\right) \right].
\end{aligned}
$$

Now we have formulas for all of the functions produced by taking the inverse Fourier transforms of the intermediary sets in the process for constructing wavelet sets. Furthermore, we know that the limit set is

$$
K = \left[-1, -\frac{1}{2}\right] \cup \left[\frac{1}{2}, 1\right]
$$

and therefore that the limiting wavelet function is

$$
\begin{aligned}
F(x) &= \frac{1}{\sqrt{2\pi}} \int_{K_n} \cos(xs)\, ds \\
&= \frac{1}{\sqrt{2\pi}} \left[\int_{-1}^{-\frac{1}{2}} \cos(xs)\, ds + \int_{\frac{1}{2}}^{1} \cos(xs)\, ds \right] \\
&= \frac{1}{x\sqrt{2\pi}} \left[\sin\left(\frac{-x}{2}\right) - \sin(-x) + \sin(x) - \sin(\frac{x}{2}) \right].
\end{aligned}
$$

However, $\sin x$ is an odd function, so $\sin(-x) = -\sin(x)$, so

$$
\begin{aligned}
F(x) &= \frac{1}{x\sqrt{2\pi}} \left[2\sin(x) - 2\sin\left(\frac{x}{2}\right) \right] \\
&= \frac{\sqrt{2}}{x\sqrt{\pi}} \left[\sin(x) - \sin\left(\frac{x}{2}\right) \right].
\end{aligned}
$$

In the next section, we examine the differences between the intermediary functions and the limit function.

4.3   Convergence Analysis

We used Matlab to compare the intermediary functions with the limiting wavelet function. To do so, we sampled each of the functions on the interval $[-15.00001, 14.99999]$ with a step size of 0.0001. The reason for the shifting of the interval was that, as we saw above, the functions all have a factor of x in their denominators. As such, Matlab cannot evaluate the functions at 0, so by shifting the interval a small amount, we can use Matlab to examine the functions around zero without having to deal differently with the discontinuity.

Once we had sampled each of the functions (including the limit function), we compared the intermediary functions F_n to the limit function F using a number of different methods. Let X_{F_n} denote the vector of sampled points of F_n, let X_F denote the vector of sampled values of F, and let M denote the length of each vector. Furthermore, if Y is a vector, we use $Y(i)$ to denote the ith component of the vector, indexed from 0.

The first method of analysis was a variation on error in quadrature. The formula is

$$
EC(n) = \sqrt{\sum_{i=0}^{M} (X_{F_n}(i) - X_F(i))^2}.
$$

The second method was a simple average of absolute errors:

$$AE(n) = \frac{\sum_{i=0}^{M} |X_{F_n}(i) - X_F(i)|}{M}.$$

The third method was a maximum absolute error:

$$ME(n) = \max_{0 \leq i \leq M} |X_{F_n}(i) - X_F(i)|.$$

The fourth and fifth methods were p-norms for $p = 3$ and $p = 10$:

$$P(n, p) = \left(\sum_{i=0}^{M} |X_{F_n}(i) - X_F(i)|^p \right)^{\frac{1}{p}}.$$

Table 4.1 states the results of these calculations for all values of n between 3 and 15, as well as results for F_{30}, F_{45}, F_{60} and F_{75}. The plot in Figure 4.1 shows these results plotted for n between 3 and 15, and Figure 4.2 shows a logarithmic plot of the same results. Figures 4.3, 4.4 and 4.5 show plots of the fifth, seventh, and ninth functions, respectively, against the limit function F.

Notice, in Table 4.1, the errors for F_{60} and F_{75} are equal. We therefore hypothesize that by F_{60} or so, we have reached the point at which computation errors have a greater effect on the convergence of the algorithm than the algorithm itself. Just by examining this table, we can see that the algorithm is converging quite rapidly.

As a further illustration of this, consider Figures 4.1 and 4.2. The first of these plots is a normal line plot of the errors from Table 4.1. The second, however, is the same plot on a logarithmic scale. This plot clearly demonstrates that the error in the even functions and the error in the odd functions each decays at a logarithmic rate. Now consider Figures 4.3, 4.4 and 4.5. These figures serve as a further illustration of the dramatic decreases in error, of these functions. We can see that F_5 differs from F substantially in only a few places, and by F_9, the functions essentially overlap with F, the limiting function.

There are a number of interesting features of the errors in these functions to be noted. The increase of the error in even numbered functions over odd numbered ones is particularly intriguing. This is not a feature we will see repeated in our examination of multidimensional convergence in the next section. As such, we hypothesize that this is unique to the one-dimensional case. In any case, this would be an interesting area for further investigation.
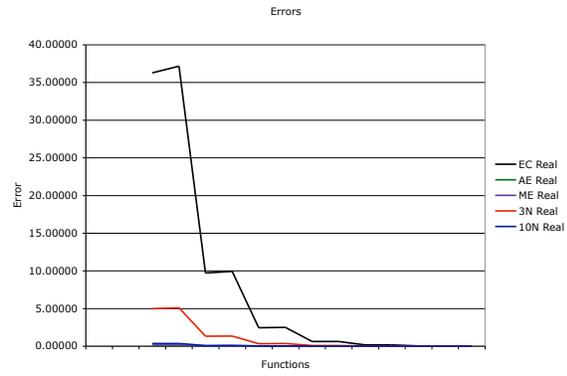
Figure 4.1: Plot of convergence data
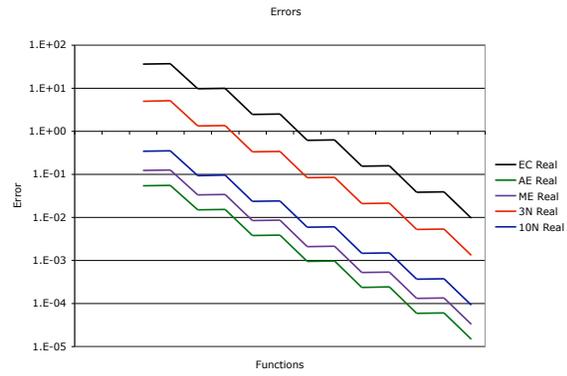
Figure 4.2: Logarithmic plot of convergence data

Table 4.1: Convergence Analysis Error Values

	$EC(n)$	$AE(n)$	$ME(n)$	$P(n,3)$	$P(n,10)$
F_3	36.38093	0.05387	0.12244	4.97353	0.34251
F_4	37.15638	0.05527	0.12551	5.09303	0.35109
F_5	9.71042	0.01492	0.03331	1.32328	0.09320
F_6	9.93828	0.01527	0.03409	1.35426	0.09538
F_7	2.44902	3.76941×10^{-3}	8.37315×10^{-3}	0.33326	0.02344
F_8	2.50607	3.85725×10^{3}	8.56794×10^{-3}	0.34102	0.02398
F_9	0.61315	9.43850×10^{-4}	2.09401×10^{-3}	0.08341	5.86241×10^{-3}
F_{10}	0.62741	9.65910×10^{-4}	2.14271×10^{-3}	0.08535	5.99875×10^{-3}
F_{11}	0.15333	2.36041×10^{-4}	5.23514×10^{-4}	0.02086	1.46568×10^{-3}
F_{12}	0.15690	2.41531×10^{-4}	5.35688×10^{-4}	0.02134	1.49977×10^{-3}
F_{13}	0.03834	5.90148×10^{-5}	1.30879×10^{-4}	5.21489×10^{-3}	3.66424×10^{-4}
F_{14}	0.03923	6.03873×10^{-5}	1.33922×10^{-4}	5.33617×10^{-3}	3.74945×10^{-4}
F_{15}	9.58431×10^{-3}	1.47540×10^{-5}	3.27196×10^{-5}	1.30374×10^{-3}	9.16061×10^{-5}
F_{30}	5.98588×10^{-7}	9.21460×10^{-10}	2.04349×10^{-9}	8.14250×10^{-8}	5.72122×10^{-9}
F_{45}	8.92220×10^{-12}	1.37368×10^{-14}	3.05250×10^{-14}	1.21374×10^{-12}	8.52976×10^{-14}
F_{60}	1.50551×10^{-14}	1.49248×10^{-17}	2.22045×10^{-16}	2.45189×10^{-15}	3.17462×10^{-16}
F_{75}	1.50551×10^{-14}	1.49248×10^{-17}	2.22045×10^{-16}	2.45189×10^{-15}	3.17462×10^{-16}
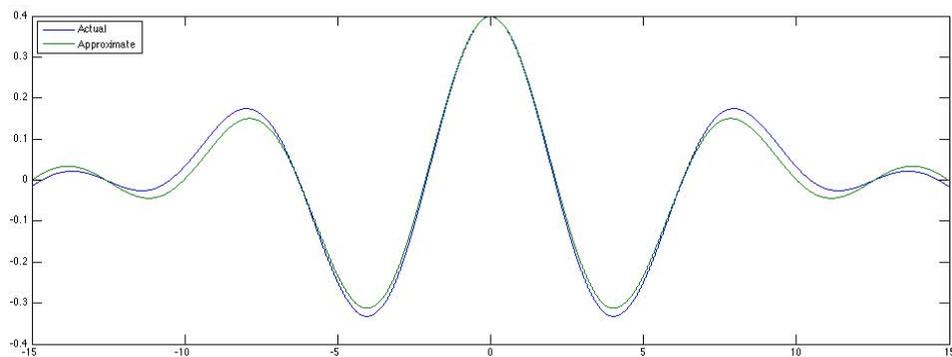
Figure 4.3: Comparing F_5 with F
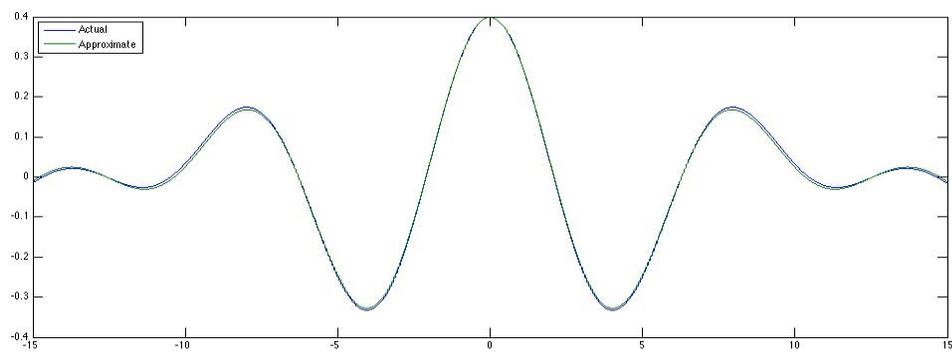
Figure 4.4: Comparing F_7 with F
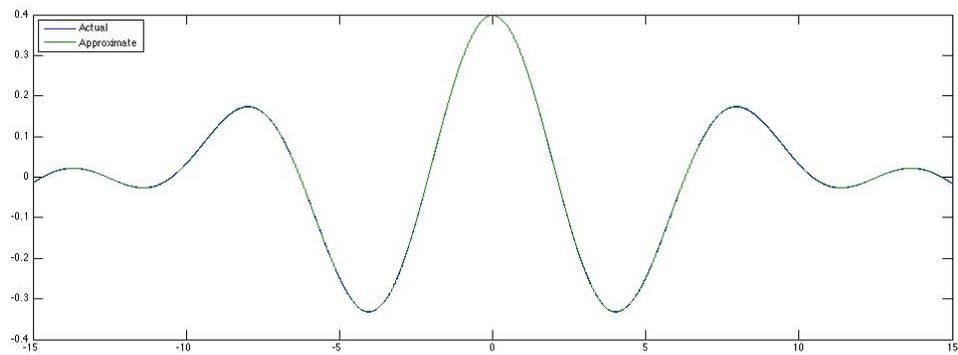
Figure 4.5: Comparing F_9 with F

Chapter 5

The Algorithm: Convergence in Higher Dimensions

We now move to a discussion of convergence in higher dimensions, further extending the work on the algorithm from [2]. This discussion illuminates some interesting features of the algorithm itself. The aim of our discussion is to examine how quickly this algorithm converges to the limiting wavelet set. Now, at each step, there are two things that change about the set. Consider the two subsets of K_n defined by:

$$K_n^- = ((K_0 \setminus A_0) \setminus \ldots \setminus A_{n-1}) \tag{5.1}$$

$$\text{and} \tag{5.2}$$

$$K_n^+ = TA_0 \bigcup \ldots \bigcup TA_{n-1}. \tag{5.3}$$

The key insight is that at each step, the two differences lie in the removal of A_{n-1} from K_n^- and the addition of TA_{n-1}. Our idea is to show that the size of A_n decreases as n increases, and this will give us a general idea of the speed of convergence of the algorithm.

To find a bound on the size of A_n we will be using the usual Lebesgue measure μ. The key property of this measure for our purposes is that, if A and B are sets with Lebesgue measure, then

$$\mu(A \cap B) \leq \min\{\mu(A), \mu(B)\}. \tag{5.4}$$

This property follows intuitively from the fact that $A \cap B$ must be a subset of both A and B, so it cannot be larger than either one. Therefore, it also must be (at most) as large as the smaller of the two.

Now that we have established the importance of examining the sizes of A_n, we begin our examination of simplifications of the algorithm.

5.1   General Simplifications

In this section, we discuss the assertion that

$$A_n = K_n^- \cap \left(\bigcup_{i=1}^{\infty} 2^{-i} T A_{n-1} \right). \tag{5.5}$$

For a complete and rigorous proof of this assertion, please refer to the appendix. In this section, we will discuss this proof from a more intuitive standpoint, and leave the appendix to the interested reader.

To begin with, we examine the definitions of K_n^- and K_n^+ given in equations (5.1) and (5.3) above. We can use the definitions in the algorithm to find bounds on these two sets. That is, we know that K_n^- is a subset of K_0, which in turn is a subset of $[-N, N]^d$. We also know that T maps from K_0 into what we will refer to as a square annulus, $[-2N, 2N]^d \setminus [-N, N]^d$. From this we can conclude that since K_n^+ is a union of sets which have all been acted upon by T, K_n^+ must be a subset of the square annulus. Merely by using these two containment properties, then, we can show that

$$A_n = K_n^- \cap \left(\bigcup_{i=1}^{\infty} 2^{-i} K_n \right), \tag{5.6}$$

which simplifies A_n substantially.

Now, by the properties of intersections and unions of sets and using the fact that $K_n = K_n^- \cup K_n^+$ by definition, we know that equation (5.6) is equivalent to

$$A_n = \left(K_n^- \cap \left(\bigcup_{i=1}^{\infty} 2^{-i} K_n^- \right) \right) \cup \left(K_n^- \cap \left(\bigcup_{i=1}^{\infty} 2^{-i} K_n^+ \right) \right).$$

By carefully manipulating our definitions of A_n and K_n, and by using properties of complements, intersections and unions of sets, we can show that

$$K_1^- \cap \left(\bigcup_{i=1}^{\infty} 2^{-i} K_1^- \right) = \emptyset.$$

However, since each K_n^- is contained in each of the K_m^- with m less than n, this in fact shows that

$$K_n^- \cap \left(\bigcup_{i=1}^{\infty} 2^{-i} K_n^- \right) = \emptyset, \tag{5.7}$$

which further simplifies A_n to

$$A_n = K_n^- \cap \left(\bigcup_{i=1}^{\infty} 2^{-i} K_n^+ \right). \tag{5.8}$$

At this point, we are only one step away from the result we desire, albeit a very long and complicated step. We are trying to show that A_n simplifies from the expression in equation (5.8) to

$$A_n = K_n^- \cap \left(\bigcup_{i=1}^{\infty} 2^{-i} TA_{n-1} \right). \tag{5.9}$$

Since $K_n^+ = TA_0 \cup \ldots \cup TA_{n-1}$, we are actually very close to our final result. To prove this last step, we use an inductive argument. Clearly the base case holds, since $K_1^+ = TA_0$ and thus

$$A_1 = K_1^- \cap \left(\bigcup_{i=1}^{\infty} 2^{-i} TA_0 \right),$$

which conforms with the expression in equation (5.9). We then proceed by strong induction, and assume that equation (5.9) holds for all $j \leq n - 1$. After yet again using properties of complements, intersections, unions, and the definitions of the sets themselves, we find that all terms in K_n^+ cancel except the last one, leaving us with

$$A_n = K_n^- \cap \left(\bigcup_{i=1}^{\infty} 2^{-i} TA_{n-1} \right). \tag{5.10}$$

For reasons that will become apparent in the next section, we have reason to believe that A_n can be simplified yet further. As we will discuss in the case of $K_0 = [-\frac{1}{2}, \frac{1}{2}]^d$, in that case we have shown that, in fact,

$$A_n = K_n^- \cap \left(\frac{1}{2} TA_{n-1} \right).$$

In the general case, however, we leave this as a conjecture. This conjecture is supported not only by its proof in the special case. The algorithm we use for this procedure depends on its truth. Although the algorithm does not implement the extensive simplifications we have shown here, it does depend on the fact that the infinite union in the expression for A_n simplifies to a single half-scaling, as we have hypothesized. Since the algorithm works as expected, this serves as a fairly strong indication that this conjecture is correct. Nevertheless, it remains to be proven rigorously.

5.2 Starting With a Box

The most important special case of this theorem is that in which we start with $K_0 = [-\frac{1}{2}, \frac{1}{2}]^d$, or a d-dimensional box. In this case, we can prove the conjecture in the last section. That is, we have found a sufficient bound on A_n. In this case, the previous analysis still holds, so we already know that

$$A_n = K_n^- \cap \left(\bigcup_{i=1}^{\infty} 2^{-i} T A_{n-1} \right). \tag{5.11}$$

However, in the case where $K_0 = [-\frac{1}{2}, \frac{1}{2}]^d$, we can see that $A_0 = [-\frac{1}{4}, \frac{1}{4}]^d$. Therefore,

$$K_n^- \subseteq K_0 \setminus A_0 = \left[-\frac{1}{2}, \frac{1}{2} \right]^d \setminus \left[-\frac{1}{4}, \frac{1}{4} \right]^d.$$

We also know that

$$T A_n \subseteq [-1, 1]^d \setminus \left[-\frac{1}{2}, \frac{1}{2} \right]^d.$$

This means that, for $i \geq 2$,

$$2^{-i} T A_n \subseteq \left[-\frac{1}{4}, \frac{1}{4} \right]^d$$

and therefore that

$$2^{-i} T A_{n-1} \bigcap K_n^- = \emptyset.$$

Therefore, we know that, in this case,

$$A_n = K_n^- \cap \frac{1}{2} T A_{n-1}. \tag{5.12}$$

Now, we know that $K_n^- \subseteq K_0$, so $\mu(K_n^-) \leq 1$. We also know that $\mu(A_0) = (\frac{1}{2})^d$. As we discussed earlier, we can conclude from equation (5.12) that

$$\mu(A_n) \leq \min(\mu(T A_{n-1}), \mu(K_n^-)). \tag{5.13}$$

Most importantly, we know that T is a measure-preserving mapping, so $\mu(T A_{n-1}) = \mu(A_{n-1})$ and therefore,

$$\mu(A_n) \leq \mu(A_{n-1}).$$

Therefore, since $\mu(A_0) = (\frac{1}{2})^d$, we know that

$$\mu(A_n) \leq 2^{-(d+n)}. \tag{5.14}$$

Now that we have a bound on $\mu(A_n)$, we examine the general iteration process. Recall that $K_n = K_n^- \cup K_n^+$. At each step we perform two operations. We subtract A_{n-1} from K_{n-1}^- and we add TA_{n-1} to K_{n-1}^+. Since T must be a measure-preserving mapping, these two operations are the same size, so to speak. That is,

$$\mu(A_{n-1}) = \mu(TA_{n-1}).$$

This gives us a sense of how quickly this process is converging to the limiting wavelet set, since $\mu(A_n)$ is bounded as shown in equation (5.14).

Furthermore, we can calculate the maximum measure of all the A_n together. Recall that

$$K_n^- = ((K_0 \setminus A_0) \setminus \ldots \setminus A_{n-1}),$$

so if we refer to the limit set as K_∞, then in the limit,

$$K_\infty^- = K_0 \setminus A_0 \setminus A_1 \setminus \ldots,$$

where we have left out parentheses but tacitly maintain the order of operations from the original definition. As long as the size of all the A_n put together is less than the size of K_0, this set will have finite, nonzero Lebesgue measure, as desired.

We know from equation (5.14) that

$$\sum_{i=0}^{\infty} \mu(A_i) \leq \sum_{i=0}^{\infty} 2^{-(d+i)}$$
$$= \frac{2^{-d}}{1 - \frac{1}{2}}$$
$$= 2^{-(d-1)}.$$

Thus the total measure of the A_n is at most $2^{-(d-1)}$, and so for $d > 1$ we know that the total measure of the A_n is at most $\frac{1}{2}$. Since $K_0 = [-\frac{1}{2}, \frac{1}{2}]^d$, as we saw above,

$$\mu(K_0) = 1 > 2^{-(d-1)} = \sum_{i=0}^{\infty} \mu(A_i). \tag{5.15}$$

Thus, we also know that K_∞^- will have finite, nonzero measure. This proves that, in this special case, the algorithm will converge, as well as giving us a sense of how quickly it does so.

5.3   Conclusions and Remarks

This chapter, in conjunction with the rigorous proofs in the appendix, provides a fairly good understanding of the convergence of the K_n in the multidimensional case. In the case where K_0 is a box, we have an even more precise understanding of the process. However, this does not end our inquiry into the convergence of the process as a whole. After all, our end result is to arrive at a wavelet function, which we create by taking the inverse Fourier transform of the characteristic function of the limiting wavelet set. So we need to discuss how our examination of convergence in the wavelet set relates to the convergence properties of the wavelet function.

In the case of the box, this is fairly straightforward. As $\mu(A_n)$ decreases, so will the amount by which the function changes at any given step. Again, the characteristic function of K_n is one on the set and zero elsewhere, so

$$(\mathcal{F}^{-1}\chi_{K_n})(x) = \int_{K_n} e^{isx}\, ds.$$

Furthermore, we can easily see that if A and B are subsets of $\mathbb{R}^n$ and $A \cap B = \varnothing$, then

$$\chi_{A \cup B}(t) = \chi_A(t) + \chi_B(t).$$

By the containment properties we have been using throughout this chapter, we know that K_n^- and K_n^+ are disjoint, so

$$\chi_{K_n}(t) = \chi_{K_n^-}(t) + \chi_{K_n^+}(t). \tag{5.16}$$

Now, the difference between K_{n-1}^- and K_n^- is the set A_n, which has measure at most $2^{-(d+n)}$. Therefore, the difference between their inverse Fourier transforms will be the integral over that same small set of e^{isx}. The same logic applies to the difference between K_{n-1}^+ and K_n^+. Thus, the difference between two consecutive F_n, where F_n is the inverse Fourier transform of χ_{K_n}, is a pair of integrals over increasingly tiny sets of e^{isx}, a function that is bounded over any bounded set (such as the K_n). Note that $|e^{isx}| \leq 1$, so

$$\left\| \int_A e^{isx}\, dx \right\| \leq \mu(A).$$

Although this does not constitute a rigorous proof or concept of convergence, it gives us an idea that the functions are converging similarly rapidly to the sets.

Chapter 6

Coding the Algorithm

In this chapter, we discuss the process for implementing the algorithm in Matlab. To do this, we adapt the pseudocode given in [2] for their algorithm. Matlab is not the only programming tool we could have used, but we hope this discussion will illuminate the implementation process in general, and facilitate the implementation process by providing an extensive discussion of the code, rather than the pseudocode given in [2]. We begin by examining the auxiliary functions which are required to run the algorithm, then discuss the algorithm itself, and finally briefly discuss how to use Matlab to explore the algorithm as we have done.

6.1 Auxiliary Functions

The first auxiliary function we include is the function inK.m, which determines if a vector is an element of $K_0 = [-\frac{1}{2}, \frac{1}{2}]^d$.

```
function [bb] = inK(x)

% Given a vector x, this function outputs a 1 if
% x is in K_0 = [−0.5,0.5]^d where d is the length
% of x. Otherwise, the function outputs a 0.

bb = 1;

for i = 1:length(x)
    if abs(x(i)) > 0.5
        bb = 0;
    end;
end;
```

The next two functions are `inN1.m` and `inN2.m`, and they determine if a vector is in $[-\frac{1}{2}, \frac{1}{2}]^d$ and $[-1, 1]^d$, respectively. These are the two sets $[-N, N]^d$ and $[-2N, 2N]^d$ from the algorithm. Our first function is `inN1.m`:

```
function [cc] = inN1(x)

% Given a vector "x", determines if x is in the smaller
% box [-N,N]^d which bounds K_0 from inK.m.

cc = 1;

for i = 1:length(x)
    if abs(x(i)) > 0.5
        cc = 0;
    end;
end;
```

Our second function is `inN2.m`:

```
function [ff] = inN2(x)

% Given a vector "x", determines if x is in the larger
% box [-2N,2N]^d where [-N,N]^d bounds K_0 from inK.m.

ff = 1;

for i = 1:length(x)
    if abs(x(i)) > 1.0
        ff = 0;
    end;
end;
```

The next function we include is called `inRangeT.m`, and this function determines whether a vector is in the range of T_1 as explained earlier in this paper. It is important to note that, when different T transformations are used, this function must be rewritten to reflect the different ranges involved. For other sample versions of this function, please see the appendix.

```
function [dd] = inRangeT(x)

% Given a vector "x", determines if x is in the range
% of the transformation T_1.

dd = 1;
```

```
for i = 1:length(x)
    if abs(x(i)) > 1
        dd = 0;
    elseif abs(x(i)) < 0.5
        dd = 0;
    end;
end;
```

6.2   The Algorithm Itself

In this section, we begin by including the code `createSet.m`. We then proceed by examining the algorithm more closely and explaining how it conforms with the algorithm as we explained it earlier in this paper. The code is as follows:

```
function [aa] = createSet(x, n)

% Given a vector "x" and an integer "n", outputs whether or not
% x is in the nth iteration of the wavelet set algorithm performed
% on the set K from the program inK.m

aa = 0;

% Checks if the iteration requested is the zeroth. If so,
% outputs the same value as that of inK, or the set K_0.
if n == 0
    aa = inK(x);
    return
end;

% Checks if the vector x is (not) in [—N,N]^d ...
if inN1(x) == 0

    % If x is neither in [—N,N]^d nor in the range of T, then
    % x cannot be in K_n.
    if inRangeT(x) == 0
        aa = 0;
    return
    end;

    % If x is not in [—N,N]^d, but is in the previous set K_{n—1},
    % then x is in K_n.
    if createSet(x, n—1) == 1
        aa = 1;
    return
```

```
        end;

        % Since x (not in [-N,N]^d) is in the range of T, we can now
        % redefine it to be its inverse image under T.
        x = Tinverse(x);

        % If that inverse image is in the previous set K_{n-1} ...
        if createSet(x, n-1) == 1

            % If 2T^{-1}(x) is in [-2N,2N]^d and 2T^{-1}(x) is in the
            % previous set K_{n-1}, then x is in K_n.
            if inN2(2.*x) == 1 && createSet(2.*x, n-1) == 1
                aa = 1;
            return

            % If the above condition is not met, x is not in K_n.
            else
                aa = 0;
            return
            end;
        end;
    end;  % Ends examination of x not in [-N,N]^d (and resets to
          % original x from 2T^{-1}(x)).

    % If x is in the previous set K_{n-1} ...
    if createSet(x, n-1) == 1

        % If 2x is in [-2N,2N]^d and 2x is in the previous set K_{n-1}
        % then x is not in K_n.
        if inN2(2.*x) == 1 && createSet(2.*x, n-1) == 1
            aa = 0;
        return

        % Otherwise, x is in the set if and only if x was in K_{n-1}.
        else
            aa = createSet(x, n-1);
        return
        end;
    end;
```

Since it is important for our work, we will go over this code more carefully and explain how each part corresponds to the algorithm.

```
aa = 0;

% Checks if the iteration requested is the zeroth. If so,
% outputs the same value as that of inK, or the set K_0.
if n == 0
    aa = inK(x);
    return
end;
```

This section of the code begins by presuming that our vector x is not in the set we are constructing and then examines the base case when $n = 0$. In this case, we wish to output K_0, which is defined by the auxiliary function inK. The function returns the value of inK applied to x, as desired.

This next section of code is long and complicated, so we start by noting that it begins with the following code.

```
% Checks if the vector x is (not) in [−N,N]^d ...
if inN1(x) == 0
```

That is, none of the following cases are run for those vectors in the smaller box $[-N, N]^d$. The first case is as follows:

```
% If x is neither in [−N,N]^d nor in the range of T, then
% x cannot be in K_n.
if inRangeT(x) == 0
    aa = 0;
return
end;
```

This case is rather self-explanatory using the comment in the code. Recall that $K_n^- = ((K_0 \setminus A_0) \setminus \ldots \setminus A_{n-1}$ and $K_n^+ = TA_0 \cup \ldots \cup TA_{n-1}$. If x is in neither the smaller box nor the range of our transformation T, then it can be in neither $K_n^- \subseteq K_0 \subseteq [-N, N]^d$ nor $K_n^+ \subseteq \text{range}(T)$. The next case is:

```
% If x is not in [−N,N]^d, but is in the previous set K_{n−1},
% then x is in K_n.
if createSet(x, n−1) == 1
    aa = 1;
return
end;
```

This case reflects the fact that if x is not in the smaller box $[-N, N]^d$ but is in the previous set K_{n-1}, then x must be an element of K_{n-1}^+. Since $K_i^+ \subseteq K_j^+$ for all $i < j$, we know that if $x \in K_{n-1}^+$ then $x \in K_n^+ \subseteq K_n$. Now,

after these cases, we know that all other $x \notin [-N, N]^d$ must be in the range of our transformation T, so we redefine x to be its inverse under T:

```
% Since x (not in [−N,N]^d) is in the range of T, we can now
% redefine it to be its inverse image under T.
x = Tinverse(x);
```

Once we have done this, we examine some possibilities about this new value for x. For clarity, although it is denoted as a simple x in the code, we will continue to refer to it as $T^{-1}(x)$ in our explanation.

```
% If that inverse image is in the previous set K_{n−1} ...
if createSet(x, n−1) == 1
```

This code states that the next conditional statement only applies to those x such that $T^{-1}(x) \in K_{n-1}$.

```
% If 2T^{−1}(x) is in [−2N,2N]^d and 2T^{−1}(x) is in the
% previous set K_{n−1}, then x is in K_n.
if inN2(2.*x) == 1 && createSet(2.*x, n−1) == 1
    aa = 1;
return

% If the above condition is not met, x is not in K_n
else
    aa = 0;
return
end;
```

This conditional statement considers the vector $y = 2T^{-1}(x)$. If $y \in [-2N, 2N]^d$ and $y \in K_{n-1}$, then $x \in K_n$. This deals with the part of the algorithm in which we calculate A_n and transform it by T. We can see that, in this case, $y \in K_{n-1}^+$ and thus $x \in T(\frac{1}{2}K_{n-1}^+)$. At this point, we also end all the previous conditional statements, effectively resetting x to its original value (since in this case it would never have been changed).

The final piece of code is conditional upon the following statement:

```
% If x is in the previous set K_{n−1} ...
if createSet(x, n−1) == 1
```

Now, recall that at this point, $x \in [-N, N]^d$, so this statement that $x \in K_{n-1}$ is really a stipulation that $x \in K_{n-1}^-$. If so, then the following code is executed:

```
% If 2x is in [−2N,2N]^d and 2x is in the previous set K_{n−1}
% then x is not in K_n.
if inN2(2.*x) == 1 && createSet(2.*x, n−1) == 1
    aa = 0;
return

% Otherwise, x is in the set if and only if x was in K_{n−1}.
else
    aa = createSet(x, n−1);
return
end;
```

This final conditional statement, then, explores the case where $x \in K_{n-1}^-$ and $2x \in K_{n-1}^+$. If this is the case, then clearly $x \in K_{n-1} \cap \frac{1}{2}K_{n-1}$, so $x \in A_{n-1}$ and therefore $x \notin K_n$. Finally, if $x \in K_{n-1}^-$ and $2x \notin K_{n-1}^+$, then $x \in K_n$. This reflects the fact that those elements of K_{n-1}^- which are not in A_{n-1} will remain and be elements of K_n.

Now that we have explained the functions we use to implement this algorithm, we will explain the command line methods we can use to explore the algorithm with Matlab.

6.3   Using Matlab to Explore the Algorithm

The functions explained above provide us the tools we need to explore this algorithm with Matlab. However, we must know the correct commands to use on the command line to properly use these functions. To do this, we begin by providing an example series of commands which will create a plot of K_{10} using the transformation T_1 as in Figure 3.5. To use these commands, we first must write the program createZ.m.

```
function [gg] = createZ(x, y, n)

% Given x, y as used to create meshgrid, creates a matrix of Z values
% from createSet

gg = zeros(length(x), length(y));

for i = 1:length(x)
    for j = 1:length(y)
        gg(i, j) = createSet( [x(i), y(j)], n);
    end;
end;
```

This function takes as input a pair of vectors x and y. These vectors will later be used on the command line with the function `meshgrid`, which creates a matrix of all possible pairings of components of these matrices. This matrix is then used to create the contour plots we need. The program `createZ` takes these same x and y as input and outputs a vector which can be used as the Z, or height values for our contour plots, from the function `createSet`.

```
>> x = −1.5:0.0005:1.5;
>> y = −1.5:0.0005:1.5;
>> [X, Y] = meshgrid(x, y);
>> Z = createZ(x, y, 10);
```

The commands up until this point have created X, Y and Z, which we will now use to create a contour plot as in Figure 3.5.

```
>> contourf(X, Y, Z)
>> colormap bone
>> colormapeditor
```

The last two commands here are completely optional. The first changes the plot to one which is black and white, where the white areas denote points in the set K_{10}. The last command, `colormapeditor`, allows us to change the colors so that black areas denote points in the set.

There are a couple of Matlab tips to note before we move on. First of all, using semicolons after the lines in the first section of code may seem blatantly obvious, but is worth mentioning. Forgetting to do this will fill your screen with vectors and matrices, not to mention taking up extra time. There are many colormaps which can be used, but especially for characteristic functions of sets, it is easiest to see the set when the contour map is in black and white. We chose the `bone` colormap to do this, but this is only one of a few options.

Now we briefly discuss the process by which we can visualize the inverse Fourier transforms of these characteristic sets. The key point here is to remember to use the `ifftshift` function to ensure that the Fourier transformed function is oriented properly. The correct code is below, and will produce two plots, one of the absolute values of the function, and one of the real values.

```
>> S = ifftshift(ifft2(Z));
>> contourf(X, Y, abs(S))
```

(a) $|\mathcal{F}(F_{10})|$.

(b) $\Re(\mathcal{F}(F_{10}))$.
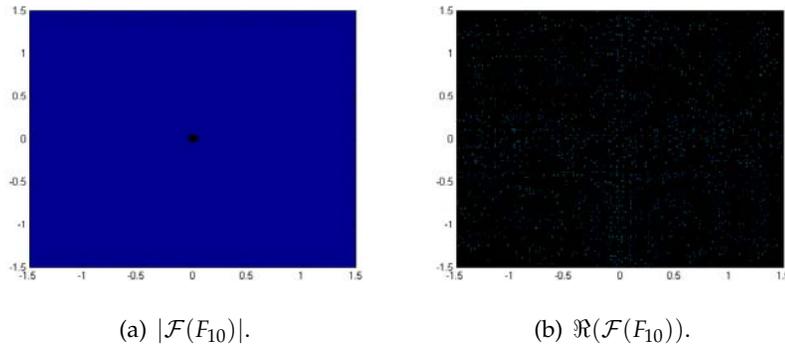
Figure 6.1: Unzoomed plots.

(a) $|\mathcal{F}(F_{10})|$.

(b) $\Re(\mathcal{F}(F_{10}))$.
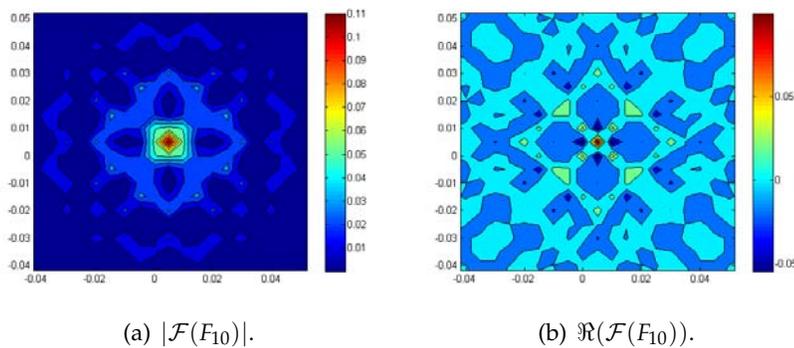
Figure 6.2: Zoomed plots.

```
>> figure
>> contourf(X, Y, real(S))
```

The resulting plots will not look like wavelet functions, as seen in Figure 6.1. However, once we zoom in on the origin a few times, we get the plots in Figure 6.2.

The other important error to note is that of not shifting the inverse Fourier transform using the `ifftshift` function. In this case, the situation in Figure 6.3 will result. If we zoom in on a corner, we get the results in Figure 6.4.

Note that in these zoomed figures, we have zoomed in on the upper left hand corner, or the corner in the second quadrant. The other corners are similar, with appropriate symmetries taken into account.

(a) $|\mathcal{F}(F_{10})|$.

(b) $\Re(\mathcal{F}(F_{10}))$.
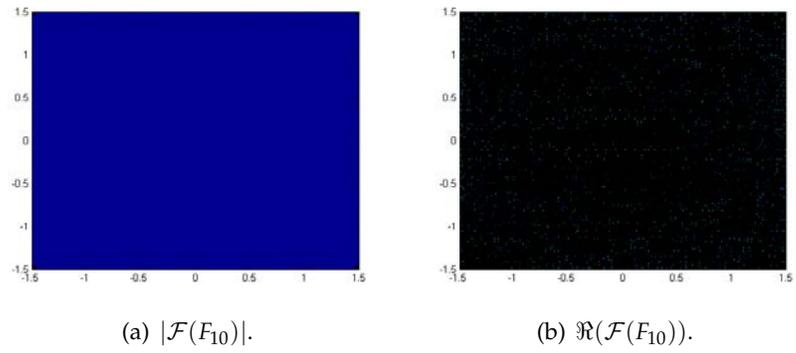
Figure 6.3: Unzoomed, unshifted plots.

(a) $|\mathcal{F}(F_{10})|$.

(b) $\Re(\mathcal{F}(F_{10}))$.
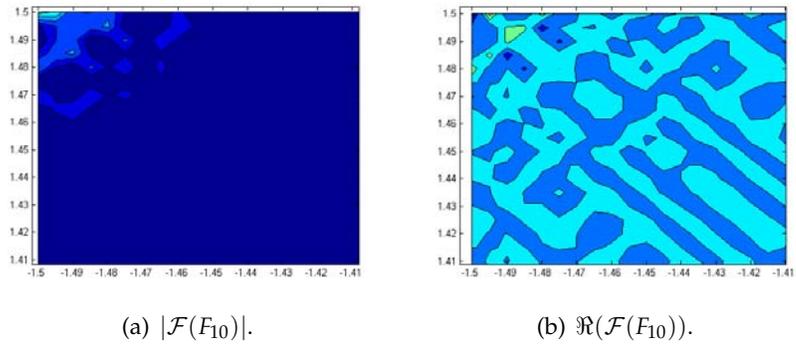
Figure 6.4: Zoomed, unshifted plots.

The last bit of code we include in this section is the code used to compare sets of functions as in the chapter on convergence in one dimension. This code is well commented and fairly self-explanatory. For definitions of the error methods used, see Section 4.3.

```matlab
function [] = compYs(x,y1,y2)

% Computes different comparisons between two different "functions"
% sampled at the same points x and plots y1 against y2 on the
% same set of axes.
% Inputs:
%   x  = a vector of x—values
%   y1 = a vector of samples of the first function (at x)
%   y2 = a vector of samples of the second function (at x)
%
% Computes the following comparisons:
%
% Error in Quadrature of Real Values:
%   SQRT( SUM( ABS( y1(i)—y2(i) )^2 ) )
%
% Average Absolute Error (N is the number of points in x):
%   SUM( ABS( y1(i) — y2(i) ) ) / N
%
% Maximum Error in Real Values:
%   MAX( ABS( y1(i) — y2(i) ) )
%
% p—Norm for p = 3:
%   NORM( y1 — y2 , 3)
%
% p—Norm for p = 10:
%   NORM( y1 — y2 , 10)

% ——————————————————————————————————

% Check to make sure sizes of vectors all match
if length(x) ≠ length(y1)
    disp('Error: These vectors have unequal lengths')
    return;
elseif length(x) ≠ length(y2)
    disp('Error: These vectors have unequal lengths')
    return;
end;

% ——————————————————————————————————

% Compute Re, which is a vector of absolute values of errors in real
% values of y1 and y2, respectively (if y1 and y2 are real, simply
% computes the absolute values of the differences between components
% of y1 and y2):

Re = zeros(length(x),1);

for i = 1:length(x)
    Re(i) = real(y1(i)) — real(y2(i));
```

```matlab
end;

% ───────────────────────────────────────

% Compute error in quadrature for real values:

ECA = 0;

for i = 1:length(x)
    ECA = ECA + Re(i)^2;
end;

ECA = sqrt(ECA);

% Display error in quadrature for real values:
disp('The error in quadrature of the real values is:')
disp(ECA)

% ───────────────────────────────────────

% Compute average error in real values:

AEA = 0;

for i = 1:length(x)
    AEA = AEA + abs(Re(i));
end;

AEA = AEA/length(x);

% Display average error in real values:
disp('The average error in real values is:')
disp(AEA)

% ───────────────────────────────────────

% Compute maximum error in absolute values:

MEA = max( abs(Re) );

% Display maximum error in real values:
disp('The maximum absolute real error is:')
disp(MEA)

% ───────────────────────────────────────

% Compute the 3—Norm of absolute values:
N3 = norm(abs(Re), 3);
```

```matlab
% Display the 3-Norm:
disp('The 3-Norm of the real error vector is:')
disp(N3)

% Compute the 10-Norm:
N10 = norm(abs(Re), 10);

% Display the 10-Norm:
disp('The 10-Norm of the real error vector is:')
disp(N10)

% ─────────────────────────────────────────────

% Creates a set of three subplots displaying the following:
%   1. The real parts of y1 and y2
%   2. The imaginary parts of y1 and y2
%   3. The absolute values of y1 and y2


subplot(2,1,1);
plot(x,real(y1),x,real(y2))
axis([-15 15 -0.4 0.4])
title('Real Parts')
h = legend('Actual','Approximate',2);
set(h,'Interpreter','none')
subplot(2,1,2);
plot(x,abs(y1),x,abs(y2))
axis([-15 15 0 0.4])
title('Absolute Values')
h = legend('Actual','Approximate',2);
set(h,'Interpreter','none')
```

Chapter 7

Future Work

There are many aspects of this topic in which opportunities for future work abound. The most obvious is that of proving the final hypothesis regarding convergence of the process in general multi-dimensional cases. We can see from the coded algorithm that this hypothesis must be true for the algorithm to work properly. All that remains is to find a rigorous mathematical proof of this step. There is also the question of the convergence in the one-dimensional case, where the even functions F_{2n} were consistently farther away from the limit than the odd functions F_{2n-1} of smaller index. It would be interesting to see if this sort of behavior was unique to the one-dimensional case, and if so, if it occurs in all one-dimensional implementations of the algorithm.

Another interesting theoretical direction to take this topic would involve some recent work by Kathy Merrill on wavelet sets that are finite unions of convex sets. Professor Merrill presented a preliminary report on this work at the special session on wavelet sets and tilings of $\mathbb{R}^n$ at the Joint Mathematics Meetings of 2008 in San Diego. These sets are constructed using a similar procedure, and it would be interesting to examine such sets in the context of the algorithm examined in this paper. It would also be interesting to investigate the convergence properties of those cases, and to evaluate any differences between those properties and the convergence patterns observed in this paper.

On a less theoretical note, there are a number of computational problems which could be interesting and accessible on this topic. The first would be to examine implementation of this algorithm in other programs, such as Python or Mathematica. The algorithm can clearly be adapted to other programming languages, but it would be interesting to examine the

benefits and challenges of those languages compared with Matlab. This would also be helpful in determining the applicability of the results of this analysis. Perhaps in another language, there is a more effective, efficient or accurate method of acquiring the wavelet function associated with the limiting set. This would be not only interesting but also useful in terms of finding applications for this work.

Similarly, it is possible that this algorithm can be streamlined to make it more efficient. As it stands, the auxiliary functions appear to be the main source of slow run-times, especially as the T transformations become more complicated (see Appendix B for examples). These functions are called multiple times, which increases the effect their inefficiency can have on the algorithm as a whole. Using the simplest T transformation and running the sixth iteration of the algorithm takes over seven minutes. The most complicated T transformation, T_2, takes around fourteen minutes to run the sixth iteration. Even if the algorithm itself cannot be improved, it would be useful to find a better method of implementing the necessary auxiliary functions to run the code.

Finally, we have not yet investigated implementation of this algorithm on starting sets K_0 which are not the box $[-\frac{1}{2}, \frac{1}{2}]^d$. Clearly, such an implementation is possible and would be a valuable aspect of the algorithm to implement in code. Again, a significant source of increased run-time can be caused by increased complication of K_0. Nonetheless, this type of set, with non-box versions of K_0, seem to be less commonly examined in the literature, so it would be useful to develop a number of examples of their usage and implementation.

Beyond these specific examples, the topic of wavelet sets is a relatively recent development in mathematics, and there are many avenues available for study. We hope that this thesis has provided an introduction to the interesting properties of wavelet sets, as well as a glimpse into the difficulties and rewards associated with their study. We would encourage the interested reader to consult the papers cited in the bibliography.

Appendix A

General Convergence Proofs

In this appendix, we show the results from Chapter 3 in a more rigorous format. We begin with a lemma.

Lemma A.1. *In the construction set forth by Benedetto and Leon,*

$$K_n^- \cap \left(\bigcup_{i=1}^{\infty} 2^{-i} K_n^- \right) = \emptyset.$$

Proof: To prove this lemma, we first examine the case where $n = 1$. In this case, we are examining the quantity

$$Q = K_1^- \cap \left(\bigcup_{i=1}^{\infty} 2^{-i} K_1^- \right).$$

Now we use the properties of set intersections, unions and complements, to find:

$$
\begin{aligned}
Q &= K_1^- \cap \left(\bigcup_{i=1}^{\infty} 2^{-i} K_1^- \right) \\
&= (K_0 \setminus A_0) \cap \left(\bigcup_{i=1}^{\infty} 2^{-i} (K_0 \setminus A_0) \right) \\
&= (K_0 \cap \overline{A_0}) \cap \left(\bigcup_{i=1}^{\infty} 2^{-i} (K_0 \cap \overline{A_0}) \right).
\end{aligned}
\tag{A.1}
$$

However, we know that, by definition,

$$A_0 = K_0 \cap \left(\bigcup_{i=1}^{\infty} 2^{-i} K_0 \right).$$

Therefore,

$$\overline{A_0} = \overline{K_0 \cap \left(\bigcup_{i=1}^{\infty} 2^{-i} K_0 \right)}$$

$$= \overline{K_0} \cup \overline{\left(\bigcup_{i=1}^{\infty} 2^{-i} K_0 \right)}.$$

Thus, note that

$$K_0 \setminus A_0 = K_0 \cap \overline{A_0}$$

$$= K_0 \cap \left(\overline{K_0} \cup \overline{\left(\bigcup_{i=1}^{\infty} 2^{-i} K_0 \right)} \right)$$

$$= (K_0 \cap \overline{K_0}) \cup \left(K_0 \cap \overline{\left(\bigcup_{i=1}^{\infty} 2^{-i} K_0 \right)} \right)$$

$$= K_0 \cap \overline{\left(\bigcup_{i=1}^{\infty} 2^{-i} K_0 \right)}$$

$$= K_0 \cap \left(\bigcap_{i=1}^{\infty} 2^{-i} \overline{K_0} \right). \tag{A.2}$$

Returning to our original quantity, we see that

$$Q = \left(K_0 \cap \left(\bigcap_{j=1}^{\infty} 2^{-j} \overline{K_0} \right) \right) \cap \left(\bigcup_{i=1}^{\infty} 2^{-i} \left(K_0 \cap \left(\bigcap_{k=1}^{\infty} 2^{-k} \overline{K_0} \right) \right) \right)$$

$$= \bigcup_{i=1}^{\infty} \left[\left(K_0 \cap \left(\bigcap_{j=1}^{\infty} 2^{-j} \overline{K_0} \right) \right) \cap \left(2^{-i} \left(K_0 \cap \left(\bigcap_{k=1}^{\infty} 2^{-k} \overline{K_0} \right) \right) \right) \right]$$

$$= \bigcup_{i=1}^{\infty} \left[\left(K_0 \cap 2^{-i} K_0 \right) \cap \left(\bigcap_{j=1}^{\infty} \bigcap_{k=1}^{\infty} \left(2^{-j} \overline{K_0} \cap 2^{-(k+i)} \overline{K_0} \right) \right) \right]. \tag{A.3}$$

However, every set operation other than the complement in the expression above is an intersection, and intersections commute and associate freely with each other. Therefore, this entire expression is equivalent to

$$Q = \bigcup_{i=1}^{\infty} \left[\left(2^{-i} K_0 \cap \left(\bigcap_{j=1}^{\infty} 2^{-j} \overline{K_0} \right) \right) \cap \left(K_0 \cap \left(\bigcap_{k=1}^{\infty} 2^{-(k+i)} \overline{K_0} \right) \right) \right]. \tag{A.4}$$

Now, we note that for any set A, $A \cap A = A$. Applying this property to $2^{-i}\overline{K_0}$, we can see that

$$Q = \bigcup_{i=1}^{\infty} \left[\left(\left(2^{-i}K_0 \cap 2^{-i}\overline{K_0}\right) \cap \left(\bigcap_{j=1}^{\infty} 2^{-j}\overline{K_0}\right)\right) \right.$$
$$\left. \cap \left(K_0 \cap \left(\bigcap_{k=1}^{\infty} 2^{-(k+i)}\overline{K_0}\right)\right)\right]$$
$$= \bigcup_{i=1}^{\infty} \left[\left(\varnothing \cap \left(\bigcap_{j=1}^{\infty} 2^{-j}\overline{K_0}\right)\right) \cap \left(K_0 \cap \left(\bigcap_{k=1}^{\infty} 2^{-(k+i)}\overline{K_0}\right)\right)\right]$$
$$= \varnothing.$$

Therefore, we have shown that

$$K_1^- \cap \left(\bigcup_{i=1}^{\infty} 2^{-i}K_1^-\right) = \varnothing. \tag{A.5}$$

However, recall that $K_n^- = ((K_0 \setminus A_0) \setminus \ldots \setminus A_{n-1}$ and therefore $K_n \subseteq K_m$ for all $m < n$. Thus, for all $n > 1$, we know that $K_n^- \subseteq K_1^-$. This implies that

$$K_n^- \cap \left(\bigcup_{i=1}^{\infty} K_n^-\right) \subseteq K_1^- \cap \left(\bigcup_{i=1}^{\infty} K_1^-\right)$$
$$= \varnothing.$$

Thus, for all $n \geq 1$,

$$K_n^- \cap \left(\bigcup_{i=1}^{\infty} 2^{-i}K_n^-\right) = \varnothing. \tag{A.6}$$

$\square$

Now that we have proved our lemma, we can state and prove our broader theorem.

Theorem A.1. *In the construction set forth by Benedetto and Leon,*

$$A_n = \bigcup_{i=1}^{\infty} \left(K_n^- \cap \frac{1}{2}TA_{n-1} \right).$$

Proof: First, we recall that A_n is defined to be

$$A_n = K_n \cap \left(\bigcup_{i=1}^{\infty} 2^{-i} K_n \right). \tag{A.7}$$

Now define $K_n = K_n^+ \cup K_n^-$ where

$$K_n^- = ((K_0 \setminus A_0) \setminus \ldots \setminus A_{n-1})$$
$$K_n^+ = TA_0 \cup \ldots \cup TA_{n-1}.$$

From the definition of A_n, we know that $A_n \subseteq K_n$ for all n. Also, from the definitions above, we know that $K_n^- \subseteq K_0$ for all n. As a matter of fact, $K_m^- \subseteq K_n^-$ for all $m \leq n$. Thus, for $n \geq 1$, we know that

$$K_n^- \subseteq [-N, N]^d. \tag{A.8}$$

We know that T maps K_0 to $[-2N, 2N]^d \setminus [-N, N]^d$, so by the definition of K_n^+, clearly

$$K_n^+ \subseteq [-2N, 2N]^d \setminus [-N, N]^d. \tag{A.9}$$

Now we begin to consider A_n from its definition. Using the fact that $K_n = K_n^- \cup K_n^+$, we know that

$$A_n = K_n \cap \left(\bigcup_{i=1}^{\infty} 2^{-i} K_n \right)$$
$$= (K_n^- \cup K_n^+) \cap \left(\bigcup_{i=1}^{\infty} 2^{-i} K_n \right)$$
$$= \left(K_n^- \cap \left(\bigcup_{i=1}^{\infty} 2^{-i} K_n \right) \right) \cup \left(K_n^+ \cap \left(\bigcup_{i=1}^{\infty} 2^{-i} K_n \right) \right).$$

However, by containment properties A.8 and A.9, we know that, for all $i \geq 1$, $2^{-i} K_n \subseteq [-N, N]^d$ and thus by containment property A.9,

$$K_n^+ \cap \left(\bigcup_{i=1}^{\infty} 2^{-i} K_n \right) = \emptyset. \tag{A.10}$$

Thus, we can now simplify A_n:

$$A_n = \left(K_n^- \cap \left(\bigcup_{i=1}^{\infty} 2^{-i} K_n \right) \right)$$
$$= \left(K_n^- \cap \left(\bigcup_{i=1}^{\infty} 2^{-i} K_n^- \right) \right) \cup \left(K_n^- \cap \left(\bigcup_{i=1}^{\infty} 2^{-i} K_n^+ \right) \right).$$

By the lemma proved above, we know that

$$K_n^- \cap \left(\bigcup_{i=1}^{\infty} 2^{-i} K_n^- \right) = \varnothing.$$

Therefore, we have shown that

$$A_n = K_n^- \cap \left(\bigcup_{i=1}^{\infty} 2^{-i} K_n^+ \right). \tag{A.11}$$

Now, we claim that

$$A_n = K_n^- \cap \left(\bigcup_{i=1}^{\infty} 2^{-i} K_n^+ \right)$$
$$= K_n^- \cap \left(\bigcup_{i=1}^{\infty} 2^{-i} T A_{n-1} \right).$$

To prove this, we proceed inductively. For the base case, we use $n = 1$. When $n = 1$, we have

$$A_1 = K_1^- \cap \left(\bigcup_{i=1}^{\infty} 2^{-i} K_1^+ \right)$$
$$= K_1^- \cap \left(\bigcup_{i=1}^{\infty} 2^{-i} T A_0 \right). \tag{A.12}$$

Therefore, clearly the base case conforms to the proposition. Now, we assume the proposition holds for all $1 \leq j \leq n - 1$, and proceed by strong

induction. We examine the nth case:

$$
\begin{aligned}
A_n &= K_n^- \cap \left(\bigcup_{i=1}^{\infty} 2^{-i} K_n^+ \right) \\
&= K_n^- \cap \left(\bigcup_{i=1}^{\infty} \left(2^{-i} \bigcup_{s=0}^{n-1} TA_s \right) \right) \\
&= \bigcup_{s=0}^{n-1} \left[K_n^- \cap \left(\bigcup_{i=1}^{\infty} 2^{-i} TA_s \right) \right].
\end{aligned} \tag{A.13}
$$

Before we continue, we note that $K_m^- \subseteq K_n^-$ for all $m < n$, since

$$
\begin{aligned}
K_m^- &= K_0 \cap \overline{A_0} \cap \ldots \cap \overline{A_{m-1}} \\
&\subseteq K_0 \cap \overline{A_0} \cap \ldots \cap \overline{A_{m-1}} \cap \ldots \overline{A_{n-1}} \\
&= K_n^-.
\end{aligned}
$$

Next, we consider the sth term in A.13, for $0 \leq s \leq n - 2$. We call this term P_s.

$$
\begin{aligned}
P_s &= K_n^- \cap \left(\bigcup_{i=1}^{\infty} 2^{-i} TA_s \right) \\
&\subseteq \left(K_{s+1}^- \cap \left(\bigcup_{i=1}^{\infty} 2^{-i} TA_s \right) \cap \overline{A_{s+1}} \right).
\end{aligned} \tag{A.14}
$$

However, by our inductive hypothesis,

$$
A_{s+1} = K_{s+1}^- \cap \left(\bigcup_{i=1}^{\infty} 2^{-i} TA_s \right). \tag{A.15}
$$

Therefore, combining this with the expression in A.14, we see that

$$
\begin{aligned}
P_s &\subseteq \left(K_{s+1}^- \cap \left(\bigcup_{i=1}^{\infty} 2^{-i} TA_s \right) \right) \cap \overline{(A_{s+1})} \\
&= \left(K_{s+1}^- \cap \left(\bigcup_{i=1}^{\infty} 2^{-i} TA_s \right) \right) \cap \overline{\left(K_{s+1}^- \cap \left(\bigcup_{i=}^{\infty} 2^{-i} TA_s \right) \right)} \\
&= \varnothing.
\end{aligned}
$$

All terms $0 \leq s \leq n - 2$ are equal to $\emptyset$, so the only remaining term is

$$P_{n-1} = K_n^- \cap \left(\bigcup_{i=1}^{\infty} 2^{-i} T A_{n-1} \right).$$

Thus

$$A_n = K_n^- \cap \left(\bigcup_{i=1}^{\infty} 2^{-i} T A_{n-1} \right). \tag{A.16}$$

$\square$

Appendix B

Extra Code

In this appendix, we include four auxiliary functions to the algorithm which were not included in the coding chapter. These are those written specifically for T_2 and T_3, for both the `inRange` and the `Tinverse` functions.

The following two functions are `inRangeTb` and `inRangec`, which test if a given point is contained in the range of T_2 and T_3, respectively, as defined in Chapter 3.

```
function [dd] = inRangeTb(x)

dd = 0;

if 0 ≤ abs(x(1)) && 0.5 ≥ abs(x(1)) && 0.5 ≤ abs(x(2)) && 1 ≥ abs(x(2))
    dd = 1;
elseif 0 ≤ abs(x(2)) && 0.5 ≥ abs(x(2)) && 0.5 ≤ abs(x(1)) && 1 ≥ abs(x(1))
    dd = 1;
end;
```

```
function [dd] = inRangeTc(x)

dd = 0;

if x(1) ≥0 && x(2) ≥ 0.5 && (x(1)+x(2)) ≤ 1
    dd = 1;
elseif x(1) ≥ 0.5 && x(2) ≥ 0 && (x(1)+x(2)) ≤ 1
    dd = 1;

elseif x(1) ≥ 0.5 && x(2) ≤ 0 && (x(1)−x(2)) ≤ 1
    dd = 1;
elseif x(1) ≥ 0 && x(2) ≤ −0.5 && (x(1)−x(2)) ≤ 1
```

```
    dd = 1;

elseif x(1) ≤ 0 && x(2) ≥ 0.5 && (x(2)−x(1)) ≤ 1
    dd = 1;
elseif x(1) ≤ −0.5 && x(2) ≥ 0 && (x(2)−x(1)) ≤ 1
    dd = 1;

elseif x(1) ≤ −0.5 && x(2) ≤ 0 && (x(1)+x(2)) ≥ −1
    dd = 1;
elseif x(1) ≤ 0 && x(2) ≤ −0.5 && (x(1)+x(2)) ≥ −1
    dd = 1;
end;
```

Finally, the next two functions are `Tinverseb.m` and `Tinversec.m`, which are the inverse maps of T_2 and T_3, respectively.

```
function [ee] = Tinverseb(x)

ee = x;

if 0 ≤ abs(x(1)) && 0.5 ≥ abs(x(1)) && 0.5 ≤ abs(x(2)) && 1 ≥ abs(x(2))
    ee(2) = ee(2) − sign(ee(2));
elseif 0 ≤ abs(x(2)) && 0.5 ≥ abs(x(2)) && 0.5 ≤ abs(x(1)) && 1 ≥ abs(x(1))
    ee(1) = ee(1) − sign(ee(1));
end;



function [ee] = Tinversec(x)

ee = zeros(length(x));

if 0.5 ≤ x(1) && 1 ≥ x(1)
    ee(1) = x(1)−1;
    ee(2) = x(2);
elseif −0.5 ≥ x(1) && −1 ≤ x(1)
    ee(1) = x(1)+1;
    ee(2) = x(2);
elseif 0.5 ≤ x(2) && 1 ≥ x(2)
    ee(2) = x(2)−1;
    ee(1) = x(1);
elseif −0.5 ≥ x(2) && −1 ≤ x(2)
    ee(2) = x(2)+1;
    ee(1) = x(1);
end;
```

We include these functions as an appendix for two reasons. First, we hope that the interested reader will be able to reproduce the sets created by this algorithm with very little difficulty. This code will assist the reader in

this implementation, although the code itself is far from efficient.

The second reason is to note the increasing complexity of the code as the T transformation increases in complexity. As this complexity increases, so does the run time of the algorithm as a whole. We were unable to write more efficient versions of these functions, and as a result did not pursue more complicated T transformations or more complicated starting sets K_0. With some streamlining, however, it should be possible to implement such things by adapting the code we have provided here.

Bibliography

[1] Lawrence W. Baggett, Herbert A. Medina, and Kathy D. Merrill. Generalized multi-resolution analyses and a construction procedure for all wavelet sets in $\mathbb{R}^n$. *The Journal of Fourier Analysis and Applications*, 5(6): 563–573, 1999.

[2] John J. Benedetto and Manuel Leon. The construction of single wavelets in d-dimensions. *The Journal of Geometric Analysis*, 11(1):1–15, 2001.

[3] John J. Benedetto and Manuel Leon. The construction of multiple dyadic minimally supported frequency wavelets on $\mathbb{R}^d$. *AMS Contemporary Mathematics*, 247, 1999.

[4] Xingde Dai, David R. Larson, and Darrin M. Speegle. Wavelet sets in $\mathbb{R}^n$. *The Journal of Fourier Analysis and Applications*, 3(4):451–456, 1997.

[5] David R. Larson. Unitary systems and wavelet sets. In *Wavelet Analysis and Applications*, Applied Numerical Harmonic Analysis, pages 143–171. Birkhäuser, Basel, 2007.

[6] Kathy Merrill. Constructing wavelets from generalized filters. In *European Women in Mathematics—Marseille 2003*, volume 135 of *CWI Tract*, pages 1–9. Centrum Wisk. Inform., Amsterdam, 2005.

[7] Wayne Polyzou, Brian Kessler, and Gerald Payne. Wavelet notes. University of Iowa, 14 November 2003.

[8] Gilbert Strang and Kevin Amaratunga. Wavelets, filter banks and applications. Online Course Notes, 2004. http://web.mit.edu/18.327/.

[9] Xiaofei Zhang and David R. Larson. Interpolation maps and congruence domains for wavelet sets. *Preprint*, 2006.