Fall 2016

Wavelets on hierarchical trees

Lu Yu
University of Iowa

WAVELETS ON HIERARCHICAL TREES

by

Lu Yu

A thesis submitted in partial fulfillment of the
requirements for the Doctor of Philosophy
degree in Mathematics
in the Graduate College of
The University of Iowa

December 2016

Thesis Supervisor: Professor Palle E. T. Jorgensen

Graduate College
The University of Iowa
Iowa City, Iowa

CERTIFICATE OF APPROVAL

PH.D. THESIS

This is to certify that the Ph.D. thesis of

Lu Yu

has been approved by the Examining Committee for the thesis requirement for the Doctor of Philosophy degree in Mathematics at the December 2016 graduation.

Thesis committee: _____
Palle Jorgensen, Thesis Supervisor

_____
Richard Baker

_____
Weimin Han

_____
Surjit Khurana

_____
Gerhard Strohmer

ACKNOWLEDGEMENTS

I would like to thank my advisor, Professor Palle Jorgensen, for his generous help, guidacne and encouragement during my thesis project. This work would have been impossible without his help.

I would also like to thank my parents, for their understanding and support for me over the years.

ABSTRACT

Signals on hierarchical trees can be viewed as a generalization of discrete signals of length 2^N. In this work, we extend the classic discrete Haar wavelets to a Haar-like wavelet basis that works for signals on hierarchical trees. We first construct a specific wavelet basis and give its inverse and normalized transform matrices.

As analogue to the classic case, operators and wavelet generating functions are constructed for the tree structure. This leads to the definition of multiresolution analysis on a hierarchical tree. We prove the previously selected wavelet basis is an orthogonal multiresolution. Classification of all possible wavelet basis that generate an orthogonal multiresolution is then given. In attempt to find more efficient encoding and decoding algorithms, we construct a second wavelet basis and show that it is also an orthogonal multiresolution. The encoding and decoding algorithms are given and their time complexity are analyzed.

In order to link change of tree structure and encoded signal, we define weighted hierarchical tree, tree cut and extension. It is then shown that a simply relation can be established without the need for global change of the transform matrix. Finally, we apply thresholding to the transform and give an upper bound of error.

PUBLIC ABSTRACT

In pure and applied mathematics, the study of network graphs plays a central role. This thesis offers new tools for analysis and synthesis constructions for such specific finite graphs. We extend hierarchical algorithms which were first developed in the case of L^2 function spaces. The setting of graphs is motivated by the need to represent real world data arising in the study of irregular and complex structures. Real world problems include traffic flow, neural network, cluster analysis, and data analysis in on-line social networks.

Classic signal processing analysis already has a history and relatively mature theories and methods exist there, but extensions to analysis on graphs is so far only in its infancy. While our setting is that of graphs, we stress connections to related algorithms which were first developed in the different context of L^2 function-spaces. Our work makes precise and it identifies, in the setting of graphs, hierarchical structures and associated multiresolution scales on realistic graph families. In our presentation, we also recall some mathematical background from signal processing.

Classic signal processing and multiresolution analysis already exist in continuous models with a history and relatively mature theories there. But we point out that extensions to various families of graphs pose a whole new set of problems. Indeed, hierarchical analysis on graphs is so far only in its infancy. Motivated by applications, our present results are for finite graphs.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

CHAPTER 1
INTRODUCTION

1.1 History and Motivation

The use of hierarchical structures in mathematics has a long history, but not so in the analysis on graphs. This will be the focus of the present thesis.

In some cases, the role of particular hierarchical structures is pretty clear. For example, in various representations of numbers in bases, fractions or Cantor systems. Indeed, even in the more general framework of fractals, the starting point for the hierarchical analysis is the identification of a suitable recursive system, with each level in the system arising as a scaled version of the other. Today, the modern theory of wavelets deals with L^2 functions in one or higher dimensions. Even in this case, the possibilities for useful hierarchical structures is less transparent. When detail is added to one resolution level, the next higher resolution arises.

Since this wavelet setting is Hilbert space, it is natural to look for hierarchical structures defined in terms of (closed) subspaces, with the subspaces suitably nested and with the associated system of relative complement subspaces representing detail subspaces. In this case the scaling operations will then serve to link resolution subspaces. One talks about multiresolution analyses or algorithms, with the latter also playing a big role in signal and

image analysis. They are considered as systems of subspaces and associated operators in Hilbert space, subject to a set of axioms, which of course reflect the hierarchical structures at hand. While this approach has been known in such special cases as the Haar wavelet for a hundred years, the feasibility of rich families of wavelet multiresolution algorithms is of a more recent vintage. At the time of its inception, even though surprisingly, its connection to signal processing is now well documented: multi-band filters down-sampling and up-sampling, and filter banks. The power of the multiresolution method lies in its use of effective hierarchical algorithms, so called filter-banks. For a sample of the rich literature we mention [2], [4], [10], [11], [16] and [45].

We use the word "hierarchical" within mathematics in broad sense of scalable or self-similar structures. Within mathematics such structures arise is several areas, algebra, graph theory, analysis of fractals, number theory (positional number representations), combinatorial structures, feed-back mechanisms, recursive algorithms, complexity. Several features of these areas are reflected in the particular use of hierarchies that goes into modern analysis of wavelets. For example, wavelets realized in $L^2(\mathbb{R}^d)$. But this is by no means the only instance of hierarchical structures, or multiresolution analyses in mathematics. Nonetheless, the latter wavelet context is motivating our present study of analysis of signals which may be represented on finite graphs. Both frameworks, continuous and discrete, are used in image processing. Perhaps the case of wavelet multiresolution analyses on finite graphs is even more

directly linked to engineering applications, than is the case for the parallel analysis of the infinite-dimensional setting of L^2 wavelets. Inside this thesis we will cite related literature, as needed. But for the reader's convenience, we mention here the books [7] and [13] as supplements to the historical aspects alluded to above. Relevant references to the engineering applications involving graph analysis are [5], [8], [9], [12] (traffic analysis), [17] (semi supervised learning), [18].

Motivated by this, and by a list of questions arising in the engineering literature ([12], [17] and [29]), our present aim is to identify hierarchical structures, and associated multiresolution on finite graphs. The choice of graphs again is dictated by the engineering applications at hand. Keeping in mind nested subspaces (reflecting resolutions) and the relative complements representing detail subspaces, it seems more natural to consider infinite graphs. In some way, the infinite case is easier. But, because of the engineering applications, we nonetheless focus our results on the case of classes of finite graphs.

1.2 Signals on Graphs

There has been a surge of research interest in signal processing on graphs in recent years. Graph is a powerful model to represent real world data that arise from irregular and complex structures, thanks to its flexibility and variations. Many real world applications have been found, such as Internet

traffic flow, neural network, cluster analysis and data analysis in on-line social networking service. See [9], [38], [39] and [41] for recent development.

Classic signal processing, the study of signals on Euclidean domain, has a long history and relatively mature theories and methods. It is natural to make analogues for graph signals. However, there are difficulties, some of which are challenging and still under research.

- Due to the irregular nature of graphs, even the most common operations on classic signals are not straight forward in the graph setting. For example, translation and downsampling have no clear analogues in a general graph setting. We usually need a more specific graph structure and make appropriate modification.
- Fourier transform and spectral analysis of signals on graphs usually requires eigen decomposition of the graph Laplacian Matrix. The can lead to calculation difficulties in applications. See [18] and [14].
- Wavelet analysis often involves translation, downsampling and upsampling. These need to be carefully redefined for graph signals. See [12] and [19].
- Problems arising from applications usually yield finite graphs and finite dimensional spaces, rather than infinite dimensional Hilbert spaces that are more common in theoretical use.

We start from discrete time signal of length 2^N, which has been widely studied and used in discrete wavelet transform. The data structure of such

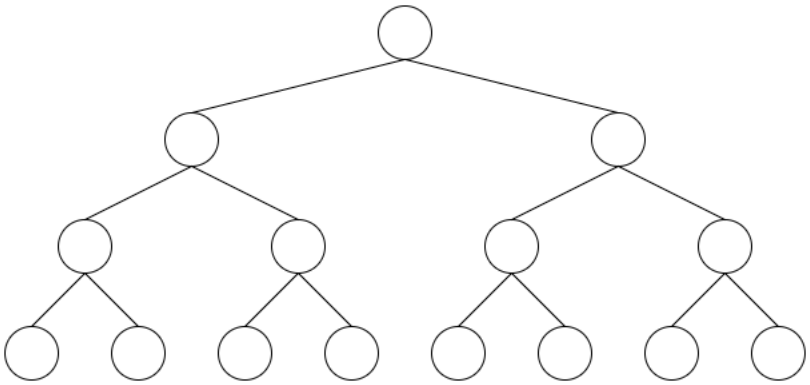

Figure 1.1: Example of perfect binary tree

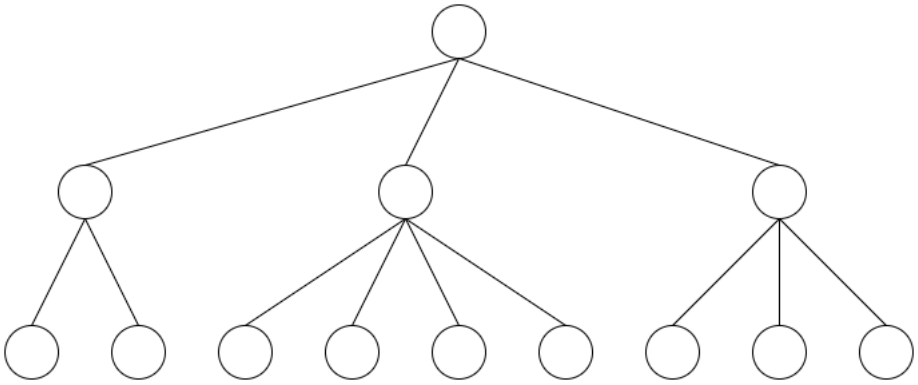

Figure 1.2: Example of hierarchical tree

signal resembles a perfect binary tree. We extend a perfect binary tree to a k-ary tree with signal stored in its bottom level nodes. The difference between these two data structures is not huge, so we are able to carry over many good results from the classic case. Changes need to be made at times and there are still some limitations. See Figures 1.1 and 1.2.

Tree is a special category of graphs. Since every finite connected graph has a spanning tree, a subgraph that contains all vertices of the original graph. Therefore, tree can be seen as a building block for a general graph. We hope, by studying signals on hierarchical trees (as will be defined in Chapter 2), we can make a first step effort to link the classic wavelet theory to signals on general graph.

The main contributions of this work are three things.

1. As an analogue to classic wavelet theory, we give definitions to scaling and translation operators for signals on hierarchical trees. We then define multiresolution analysis for such signals and prove that the wavelet basis suggested in computer science literature [17] is an orthogonal multiresolution. We also make a classification for all possible choices that lead to an orthogonal multiresolution.
2. In attempt to find a more efficient algorithm, we construct a new wavelet basis that better respects tree structure. It is then shown that it is also an orthogonal multiresolution. The encoding and decoding algorithms are given. The time complexity of both algorithms is briefly analyzed.

3. As an attempt to link change of tree structure and encoded signal, we define weighted hierarchical tree that can retain information of its structure after some change. Modified wavelet transform is introduce and tree cut is defined. This transform is shown to be a multiresolution, although not orthogonal. We then show that a simply relation can be established between tree cut and encoded signal. Tree extension is also introduced parallel to tree cut. Finally, a classification for all possible choices that lead to a multiresolution on weighted hierarchical trees is given.

Chapter 2 gives definitions and background of hierarchical tree structure and the associated signal space. In Chapter 3, we build wavelet basis suggested in [17], and calculate its transform matrix and inverse transform matrices. Chapter 4 through 6 cover the three main contributions of this thesis, as listed above. Chapter 7 deals with thresholding method and gives a theoretical upper bound of error. Chapter 8 summarizes our work and points to future research possibilities.

As for references, this work is generally based on [1], [7], [13], [22] and [28] for classic wavelet theory. For discrete wavelet algorithms, we have made reference to [21] and [34]. For signal processing, [35] and [34] are used. For general graph theory, we refer to [3], [5], [24] and [46].

CHAPTER 2
BACKGROUND

In our discussion of finite graphs, we have adopted standard definitions. See, for example, [25], [27], [33], [42] and [43]. Since there are variations in the literature, we have fleshed out more precise conditions to better suit our needs. See Definitions 2.2 to 2.6.

2.1 Hierarchical Tree

There are several different ways to define a hierarchical tree. We choose the best one to fit our wavelet building later. See for [17], [36] and [37] for variations. For general background of graph theory, see [43].

Definition 2.1 (Rooted tree)**.** A rooted tree is a tree in which one vertex has been designated the root. In a rooted tree, level of a node is the number of edges along the unique path between it and the root plus one. The root is at top level (level 1) and the largest level is called the bottom level. The level of a rooted tree is equal to its largest level. A child node is a node directly connected to another node (called its parent node) when moving away from the root.

Definition 2.2 (Hierarchical tree)**.** A k-ary tree is a rooted tree in which each node has at most k child nodes. A hierarchical tree is an unweighted and undirected k-ary tree whose leaf nodes are on the same level.

A perfect binary tree is a special case of hierarchical tree where $k = 2$. In order to resemble this structure, we require all the leaf nodes of a hierarchical tree to only occur on the same bottom level. For a more general case, if we have a rooted k-ary tree with some leaf nodes not on the bottom level, we can always extend the tree by adding a path from those nodes to the bottom level.

At each node, we have a subtree, which is also a hierarchical tree, rooted at this node. This leads to the definition of folder and subfolder.

Definition 2.3 (Folder and subfolder)**.** In a hierarchical tree, a folder at a node is the collection of leaf nodes contained in the subtree rooted at this node. A subfolder of a node is a folder generated by one of its child nodes.

We will use the following notation for folders throughout this work.

Definition 2.4 (Folder notation)**.** In a hierarchical tree, the leaf nodes are denoted by $X = \{x_1, x_2, \ldots, x_N\}$. At level l, denote the k-th folder by $X_{l,k}$ and the number of leaf nodes in $X_{l,k}$ by $m_{l,k} = |X_{l,k}|$. Denote the number of nodes at level l by n_l.

It is easy to see that

$$\bigcup_{k=1}^{n_l} X_{l,k} = X \text{ and } \sum_{k=1}^{n_l} m_{l,k} = N \tag{2.1}$$

As a special kind of tree, a hierarchical tree has an efficient representation. Since it is completely determined by its folder structure, we can use the

number of subfolders contained in each folder at level l to represent a hierarchical tree as follows. We can leave out the bottom level because its structure is trivial, always consisting of folders of size 1. We choose this definition so that extension and cut of an existing tree can be carried out easily under this representation (See Chapter 6).

Definition 2.5 (Vector representation)**.** The vector representation of a hierarchical tree is a sequence of vectors $u_l = (u_{l,1}, u_{l,2}, \ldots, u_{l,n_l})$ for $l = 1, 2, \ldots, L-1$, where $u_{l,k}$ is the number of subfolders contained in $X_{l,k}$.

From the above definition, we can see that

$$\sum_{k=1}^{n_l} u_{l,k} = \text{ size of } u_{l+1} = n_{l+1} \tag{2.2}$$

Example 2.1. Figure 2.1 shows a hierarchical tree of level 3 with its folder structure. Following Definitions 2.4 and 2.5, we have

$$L = 3 \text{ and } X = \{x_1, x_2, \ldots, x_9\} \tag{2.3}$$

$$u_1 = (u_{1,1}) = (3) \text{ and } u_2 = (u_{2,1}, u_{2,2}, u_{2,3}) = (2, 4, 3) \tag{2.4}$$

Definition 2.6 (Distance)**.** For two leaf nodes $x, y \in X$, define the distance between x and y as

$$d(x, y) = \begin{cases} \dfrac{|\text{folder}(x, y)|}{|X|} & x \neq y \\ 0 & x = y \end{cases} \tag{2.5}$$

where folder(x, y) is the smallest folder that contains both x and y.

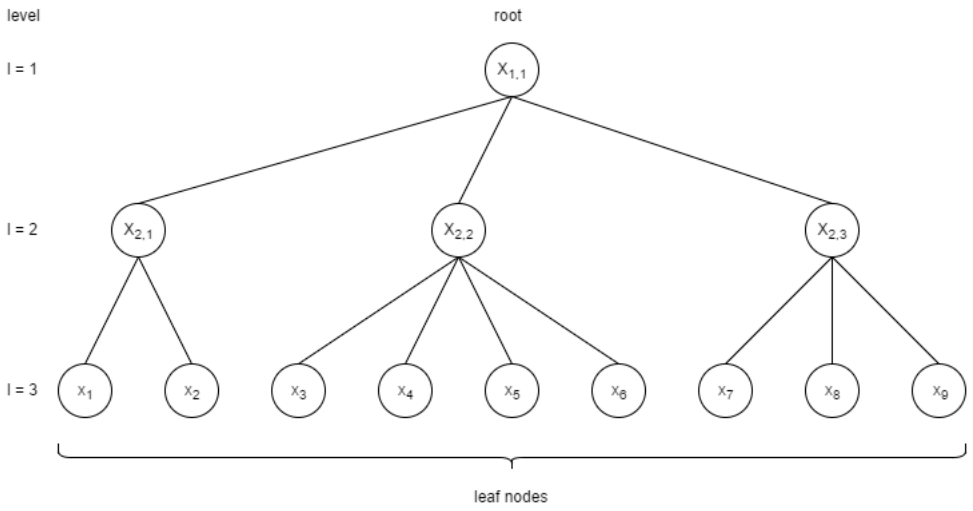

Figure 2.1: Example of hierarchical tree with structure notation

The smaller the distance between two leaf nodes, the "closer" they are. In applications, we would expect data collected in the same small folder to have close relation. For example, two leaf nodes in the same low level folder are supposed to have similar values.

2.2 Signal Space

A perfect binary tree naturally induces the data structure of signal of length 2^N. As an analogue, a hierarchical tree induces the data structure of signal of length N, where N is the number of leaf nodes. For general background of signal processing, see [34] and [35].

Definition 2.7 (Signal space)**.** A signal on a hierarchical tree (Definition 2.2) with leaf nodes $X = \{x_1, x_2, \ldots, x_N\}$ is a function $f : X \to \mathbb{R}$. The signal space is the collection of all signals $V = \{f | f : X \to \mathbb{R}\}$.

Signals on a hierarchical tree can be seen as a vector in $\mathbb{R}^N$, coupled with the tree structure. Thus we can talk about vector space operations. Linear transformations can be viewed as matrices.

In particular, the inner product of two signals f and g is

$$< f, g >= \sum_{i=1}^{N} f(x_i)g(x_i) \tag{2.6}$$

and the L^2 norm of a signal f is

$$\|f\| = (\sum_{i=1}^{N} f(x_i)^2)^{1/2} \tag{2.7}$$

We will also use the following.

Definition 2.8 (Hadamard product)**.** The Hadamard (entrywise) product of two signals f and g is

$$f \circ g = (f(x_1)g(x_1), \ldots, f(x_N)g(x_N)) \tag{2.8}$$

Definition 2.9 (Characteristic function)**.** Let A be a subset of signal space X. The characteristic function on A is

$$\mathbf{1}_A(x_i) = \begin{cases} 1 & x_i \in A \\ 0 & x_i \notin A \end{cases} \tag{2.9}$$

2.3 Classic Wavelet Theory

Following [7] and [22], we give a brief outline of wavelet theory and multiresolution analysis on $L^2(\mathbb{R})$.

Definition 2.10 (Father and mother wavelet)**.** The father wavelet $\phi(x)$ and mother wavelet $\psi(x)$ are functions in $L^2(\mathbb{R})$ that satisfy the self-similarity conditions

$$\phi(x) = \sum_{n\in\mathbb{Z}} a_n \phi(2x - n) \tag{2.10}$$

$$\psi(x) = \sum_{n\in\mathbb{Z}} b_n \phi(2x - n) \tag{2.11}$$

for some sequences $\{a_n\}$ and $\{b_n\}$. These $\{a_n\}$ and $\{b_n\}$ are called masking coefficients.

Usually, we also require that they are piecewise continuous, have compact support and

$$\int_{\mathbb{R}} \phi(x)\,\mathrm{d}x = 1 \text{ and } \int_{\mathbb{R}} \psi(x)\,\mathrm{d}x = 0 \tag{2.12}$$

Definition 2.11 (Classic multiresolution)**.** An orthogonal multiresolution analysis on $L^2(\mathbb{R})$ is a father and a mother wavelet function $\phi(x) \in L^2(\mathbb{R})$ and $\psi(x) \in L^2(\mathbb{R})$ that generates nested subspaces and satisfies the following conditions.

$$V_m = \mathrm{span}(\phi_{m,n} | n \in \mathbb{Z}), \text{where } \phi_{m,n}(x) = 2^{-m/2}\phi(2^{-m}x - n) \tag{2.13}$$

$$W_m = \mathrm{span}(\psi_{m,n} | n \in \mathbb{Z}), \text{where } \psi_{m,n}(x) = 2^{-m/2}\psi(2^{-m}x - n) \tag{2.14}$$

$$\{0\} \subset \cdots \subset V_1 \subset V_0 \subset V_{-1} \subset \cdots \subset L^2(\mathbb{R}) \tag{2.15}$$

$$V_m \oplus W_m = V_{m-1} \text{ and } V_m \perp W_m \tag{2.16}$$

The father and mother wavelets are generated by scaling and translation of a single function. They are localized (compact support) and have unit or zero integral respectively. The father wavelets generate nested spaces and mother wavelets generate their orthogonal complements, the detail spaces. We will attempt to replicate these properties when we build wavelets on hierarchical tree. This is not a direct adaptation as the case of graphs presents a number of challenges that are not present in the classical constructions for wavelet families of $L^2(\mathbb{R})$.

In general, finding solutions to (2.10) and (2.11) is not easy. Haar wavelet is the earliest wavelet discovered and also one of the most commonly used.

Example 2.2 (Haar wavelet)**.** A classic example, proposed by Alfréd Haar, is the father function

$$\phi(x) = \begin{cases} 1 & 0 \leq x < 1 \\ 0 & \text{otherwise} \end{cases} \tag{2.17}$$

With $a_0 = a_1 = 1$ and $b_0 = 1, b_1 = -1$ in Definition 2.10, the associated mother wavelet is

$$\psi(x) = \begin{cases} 1 & 0 \leq x < 1/2 \\ -1 & 1/2 \leq x < 1 \\ 0 & \text{otherwise} \end{cases} \tag{2.18}$$

The father and mother functions are shown in Figure 2.2.

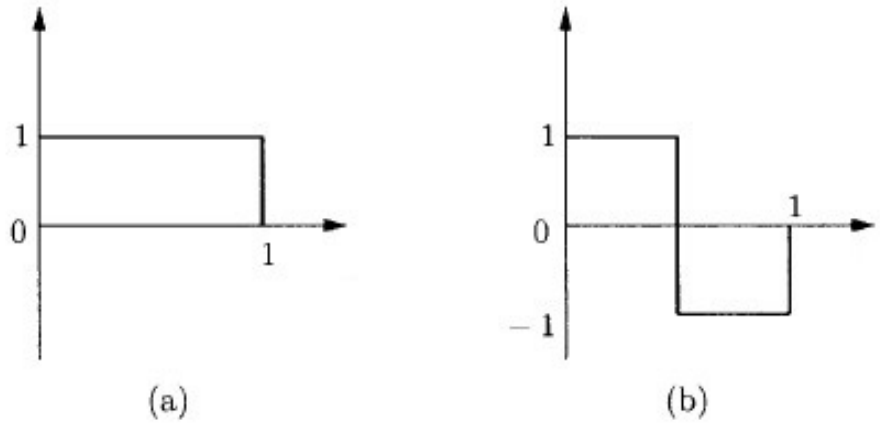

Figure 2.2: Haar father and mother functions

The scaling and translation of Haar mother functions yield an orthonormal basis of $L^2(\mathbb{R})$.

$$\{\psi_{m,n}(x) = 2^{-m/2}\psi(2^{-m}x - n)|m, n \in \mathbb{Z}\} \tag{2.19}$$

Example 2.3 (Haar matrix)**.** In the setting of hierarchical tree, it is sometimes more convenient to consider the discrete form of Haar wavelet. It is usually given in the matrix form called Haar matrix H_{2^N} that acts on signals of length 2^N. Start with (we use an unnormalized version here)

$$H_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{2.20}$$

The other Haar matrices are defined recursively as

$$H_{2N} = \begin{pmatrix} H_N \otimes (1, 1) \\ I_N \otimes (1, -1) \end{pmatrix} \tag{2.21}$$

where $\otimes$ is the matrix Kronecker product.

For example, $H_8 =$

$$\begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & -1 & -1 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{pmatrix} \tag{2.22}$$

The first row is the father wavelet that gives the average (or sum). The other rows are mother wavelets of various levels that calculates localized differences within the signal. Similar to the continuous case, the rows of H_{2^N} form an orthogonal basis of the vector space $\mathbb{R}^{2^N}$. Therefore, we can use the coordinates under this basis, inner products with the rows, to represent any vector.

Since a hierarchical tree can be seen as a generalized perfect binary tree, we will make analogue to the Haar matrix when we construct wavelet basis later in this work.

CHAPTER 3
BUILDING WAVELETS

In this chapter, we build a specific wavelet basis for signals on a hierarchical tree proposed by [17]. Following the notations in Chapter 2, we look at a hierarchical tree of level L with folders and folder sizes

$$\{X_{l,k}\} \text{ and } m_{l,k} = |X_{l,k}| \text{ for } l = 1, 2, \ldots, L; k = 1, 2, \ldots, n_l \tag{3.1}$$

and vector representation

$$u_l = (u_{l,1}, u_{l,2}, \ldots, u_{l,n_l}) \text{ for } l = 1, 2, \ldots, L-1 \tag{3.2}$$

3.1 Father and Mother Wavelets

A hierarchical tree is a generalized structure of a perfect binary tree, which naturally represents the structure of a signal of length 2^N. Therefore, we want to start from the Haar matrix defined in Example 2.3 and a special case in (2.22).

The first row is the father wavelet that takes the average value of the whole signal. The second row is the level 2 mother wavelet that calculates the difference of average value between two folders. The third and fourth row are level 3 mother wavelets that calculates the difference within level 2 folders.

As an analogue, we let the level one father wavelet take the average of the whole signal $\phi_{1,1} = (1/N, \ldots, 1/N)$.

At level l, we use a series of translated and localized father wavelets

$$\phi_{l,k} = \frac{1}{m_{l,k}} \mathbf{1}_{X_{l,k}}, k = 1, 2, \ldots, n_l \tag{3.3}$$

where $\mathbf{1}_{X_{l,k}}$ is the characteristic function on $X_{l,k}$.

For the mother wavelets, we want to use them to calculate the difference within folders at different levels. We would also like the mother wavelets to share some nice properties of the classic case, such as local support and zero mean. One possible choice is the following from [17]. The choice is certainly not unique. In Chapter 5, another form of mother wavelet is given.

Given l and k, $\psi_{l,k,j}$ calculates the difference within folder $X_{l-1,k}$. Suppose on the next level downwards, $X_{l-1,k}$ splits into $u_{l-1,k}$ subfolders $\{Y_{l,k,j} | j = 1, 2, \ldots, u_{l-1,k}\}$. We build our mother wavelets as

$$\psi_{l,k,j}(x_i) = \begin{cases} \dfrac{1}{|\bigcup_{n=1}^{j} Y_{l,k,n}|} & x_i \in \bigcup_{n=1}^{j} Y_{l,k,n} \\ \dfrac{-1}{|Y_{l,k,j+1}|} & x_i \in Y_{l,k,j+1} \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

$$j = 1, 2, \ldots, u_{l-1,k} - 1; k = 1, 2, \ldots, n_{l-1}; l = 2, 3, \ldots, L$$

This mother wavelet treats the leaf nodes in the first j subfolders evenly and takes the average. It then calculates the difference with the average of the next subfolder. The following are some useful properties of the above wavelets.

Lemma 3.1. The total number of wavelets, defined in (3.4), is $N - 1$.

Proof. Fix level l, number of wavelets is

$$\sum_{k=1}^{n_{l-1}}(u_{l-1,k}-1)=\sum_{k=1}^{n_{l-1}}(u_{l-1,k})-n_{l-1} \tag{3.5}$$

By definition, $\sum_{k=1}^{n_{l-1}}(u_{l-1,k})=n_l$. Thus, total number of wavelets is

$$\sum_{l=2}^{L}(n_l-n_{l-1})=n_L-n_1=N-1 \tag{3.6}$$

□

Lemma 3.2. The wavelet functions defined in (3.4) are mutually orthogonal.

Proof. First, note that every mother wavelet has zero sum, which means its inner product with any other signal who has constant value on the support of the wavelet signal is zero. Now suppose we have ψ_{l_1,k_1,j_1} and ψ_{l_2,k_2,j_2}.

Case 1 $l_1 > l_2$

ψ_{l_2,k_2,j_2} has constant value on all folders at level l_1, so they are orthogonal.

Case 2 $l_1 = l_2$ and $k_1 \neq k_2$

By definition, the supports of two wavelets have an empty intersection.

Case 3 $l_1 = l_2$ and $k_1 = k_2$ and $j_1 > j_2$

The support of ψ_{l_2,k_2,j_2} is $\bigcup_{n=1}^{j_2+1} Y_{l_2,k_2,n}$. With $j_1 \geq j_2+1$, by definition ψ_{l_1,k_1,j_1} has constant value on the support of ψ_{l_2,k_2,j_2}. Thus they are orthogonal. □

Lemma 3.3. A mother wavelet ψ_{l_1,k_1,j_1} defined in (3.4) is orthogonal to a father wavelet ϕ_{l_2,k_2} defined in (3.3), when $l_1 > l_2$. In particular, all mother wavelets are orthogonal to $\phi_{1,1}$.

Proof. When $l_1 > l_2$, ϕ_{l_2,k_2} has constant value on the support of ψ_{l_1,k_1,j_1}. □

3.2 Transform Matrix

Like the Haar matrix, we put $\phi_{1,1}$ in the first row and the rest rows are mother wavelets at various scales. The transform matrix is

$$M = \begin{pmatrix} \phi_{1,1} \\ \psi_{l,k,j} \end{pmatrix} = \begin{pmatrix} \phi_{1,1} \\ (\psi_{2,k,j}) \\ \vdots \\ (\psi_{L,k,j}) \end{pmatrix} \tag{3.7}$$

$$j = 1, 2, \ldots, u_{l-1,k} - 1; k = 1, 2, \ldots, n_{l-1}; l = 2, 3, \ldots, L$$

Here the mother wavelets $\{\psi_{l,k,j}\}$ are arranged in increasing order of l, k and j, with priority decreasing from l to k to j.

Given an input signal $f = (f_1, \ldots, f_N)^T$, we have an output $g = Mf = (g_1, \ldots, g_N)^T$. $\phi_{1,1}$ calculates the average of input signal and stores it in g_1. Fix l and with various k and j, a block of wavelets $(\psi_{l,k,j})$ gives the difference of average value at level l (level l details) and store it in the corresponding components in g. These are similar to discrete Haar transform. Actually, if we have a perfect binary tree, its transform matrix as defined in (3.7) is the Haar matrix with different scaling in some rows. After normalization, they will be exactly the same.

$$\begin{pmatrix} \phi_{1,1} \\ (\psi_{2,k,j}) \\ \vdots \\ (\psi_{L,k,j}) \end{pmatrix} \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_N \end{pmatrix} = \begin{pmatrix} g_1 \\ g_2 \\ \vdots \\ g_N \end{pmatrix} \tag{3.8}$$

Example 3.1. We give a concrete transform matrix based on tree structure shown in Figure 2.1. For a general program to calculate the transform matrix used in this chapter, see Appendix A.1.

$$M = \begin{pmatrix} 1/9 & 1/9 & 1/9 & 1/9 & 1/9 & 1/9 & 1/9 & 1/9 & 1/9 \\ 1/2 & 1/2 & -1/4 & -1/4 & -1/4 & -1/4 & 0 & 0 & 0 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & -1/3 & -1/3 & -1/3 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 1/2 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/3 & 1/3 & 1/3 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1/2 & 1/2 & -1 \end{pmatrix} \tag{3.9}$$

This matrix is a stretched version of Haar matrix with modification made according to tree structure. Thus it inherits many nice properties from Haar matrix. The first row is the father wavelet. The second and third rows are mother wavelets at level 2. The last six rows are level 3 mother wavelets. Note that the last six rows are blocked. Every transform matrix generated by (3.7) will have the bottom rows in this block form, since each bottom level folder contains exactly one leaf node. These blocks can be seen as the basic building elements of the wavelets, similar to the vector $(1, -1)$ for discrete Haar transform.

3.3 Inverse Transform Matrix

Theorem 3.4 (Invertible transform)**.** The transform matrix M defined in (3.7) is an N by N invertible matrix and its rows are mutually orthogonal.

Proof. By Lemma 3.1, M is an N by N matrix. By Lemma 3.2 and 3.3, the rows of M are mutually orthogonal. Obviously, each row has a non-zero norm,

so M is invertible. □

We only give the inverse transform matrix for theoretical use here. In practice, there might be (and probably are) more efficient algorithms.

Since the rows of the transform matrix M are mutually orthogonal, we have a diagonal matrix

$$MM^T = D = \text{diag}(d_1, \ldots, d_N) \tag{3.10}$$

Theorem 3.5 (Inverse matrix)**.** Let M be the transform matrix given in (3.7) and D be the diagonal matrix given in (3.10). Then the inverse transform matrix M^{-1} is

$$M^{-1} = M^T D^{-1} \tag{3.11}$$

where $D^{-1} = \text{diag}(d_1^{-1}, \ldots, d_N^{-1})$.

Proof.

$$M(M^T D^{-1}) = MM^T D^{-1} = DD^{-1} = I \tag{3.12}$$

For square matrices, it suffices to show one sided inverse. □

In some situations, it is better to use normalized transform and inverse transform matrices. With the above results, they are given as follows.

Theorem 3.6 (Normalization)**.** Let M be the transform matrix given in (3.7) and D be the diagonal matrix given in (3.10). Then the normalized transform

matrix $\tilde{M}$ is

$$\tilde{M} = D^{-1/2}M \tag{3.13}$$

And the normalized inverse transform matrix $\tilde{M}^{-1}$ is

$$\tilde{M}^{-1} = M^T D^{-1/2} \tag{3.14}$$

where $D^{-1/2} = \text{diag}(d_1^{-1/2}, \ldots, d_N^{-1/2})$.

Proof. Since $MM^T = D$, where $d_i^{1/2}$ is the norm of the i-th row of M, left multiply M by $D^{-1/2}$ normalizes it. Now that $\tilde{M}$ is an orthogonal matrix, we have

$$\tilde{M}^{-1} = \tilde{M}^T = (D^{-1/2}M)^T = M^T(D^{-1/2})^T = M^T D^{-1/2} \tag{3.15}$$

□

CHAPTER 4
MULTIRESOLUTION ANALYSIS

In studying analysis on graphs, we aim to create multiresolution analyses to achieve the power coming from intrinsic scaling similarity. This has both significance in mathematics, as well as in encoding algorithm. The main task we accomplish in this chapter is thus to identify the features of multiresolution analysis, the spaces forming a resolution, the associated operators, and the scaling similarity relation for the operators. Once we achieve the multiresolution machinery for our graph models, it will follow that many of the proofs from the classic wavelet theory, as well as signal processing tools will carry over. We have managed to do this in such a way that the proofs in the applications to graph multiresolution analysis become relatively simple.

We start from the classic multiresolution analysis on $L^2(\mathbb{R})$ in Chapter 2 (See Definition 2.11) and make proper changes for the graph setting. We need to take of several things to make a new definition of orthogonal multiresolution analysis on hierarchical trees.

- Our signal space is finite, thus we only have a finite number of wavelets and nested spaces. They start from the whole signal spaced and scaled down level by level.
- The scaling and translation need to be redefined to better suit hierarchical tree structure. They are obvious in the classic case ($L^2(\mathbb{R})$), but not

trivial for graphs.

- We want a father wavelet ϕ to be scaled and translated along folders to generate other father and mother wavelets at various levels.
- We want the father wavelets to span the nested spaces (V_n) and mother wavelets to span their orthogonal complements (W_n).
- We attempt to carry over as many good properties as possible from the classic case. If this cannot be done in some case, we will point out the reason.

indexHierarchical tree

4.1 Constructing Operators

We first give a special case of the nested spaces and their orthogonal complements, defined as follows. They are the natural choice that arises from hierarchical tree structure. We follow definitions and notations in Chapter 2.

Definition 4.1 (Nested spaces)**.** For fixed level l of a hierarchical tree, $V_l = \{f | f \text{ is constant on } X_{l,k} \text{ for all } k\}$.

V_l is the space of all signals constant on level l folders, an analogue to piecewise constant functions on $\mathbb{R}$. In particular, V_1 is the space of constant signals on the whole tree. V_L is the space of functions constant on single leaf nodes, thus $V_L = V$. It is obvious from definition that V_l form a sequence $V_1 \subset V_2 \subset \cdots \subset V_L$.

Definition 4.2 (Orthogonal complements)**.** For $l = 1, 2, \ldots, L-1$, W_l is the

orthogonal complement of V_l in V_{l+1}. That is, $W_l \oplus V_l = V_{l+1}$ and $W_l \perp V_l$.

From the above definitions, we have $(\oplus_{l=1}^{L-1} W_l) \oplus V_1 = V_L = V$. As analogue to the classic case, V_l are called nested subspaces and W_l are called detail subspaces.

Lemma 4.1. V_l and W_l in Definitions 4.1 and 4.2 are subspaces of V (closed under linear combination).

Proof. Obviously the property of being constant on level l folders still holds under linear combination. Thus V_l is a closed subspace of V. Any orthogonal complement W_l is also a closed subspace. □

As an analogue to classic signal processing, we then build some useful operators on a hierarchical tree.

Definition 4.3. The average matrix $A(n, m) = \frac{1}{n}\mathbf{1}_{n\times m}$, where $\mathbf{1}_{n\times m}$ is an n by m matrix with all entries equal to 1.

Definition 4.4 (Translation operator)**.** At level l of a hierarchical tree, we have folders $X_{l,1}, \ldots, X_{l,n_l}$ with sizes $m_{l,1}, \ldots, m_{l,n_l}$. The translation operator T_l is

$$\begin{pmatrix} 0 & 0 & \cdots & 0 & A(m_{l,1}, m_{l,n_l}) \\ A(m_{l,2}, m_{l,1}) & 0 & \cdots & 0 & 0 \\ 0 & A(m_{l,3}, m_{l,2}) & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & A(m_{l,n_l}, m_{l,n_l-1}) & 0 \end{pmatrix} \tag{4.1}$$

Some remarks follow.

- The translation is defined in a level specific way, due to the irregular tree structure.
- At each level, the translation operator calculates the sum of a folder, take average and send it to the next folder. The last folder goes back to the first. That is, a block cyclic permutation matrix.
- At the bottom level, T_L is the cyclic permutation matrix and we are back to the classic translation.
- The translation preserves the sum of whole signal.

Following this definition, we can carry over some important properties from the classic case.

Lemma 4.2. Each nested subspace V_l is invariant under T_m if $l \geq m$.

Proof. When $l \geq m$, $V_m \subseteq V_l$. By the definition of T_m, its image always has constant value on level m folders. So we have $T_m(V_l) \subseteq V_m \subseteq V_l$. □

Lemma 4.3. The detail subspace W_1 is invariant under T_2.

Proof. Signals in W_1 are exactly those that are constant on level 2 folders and have zero sum on level 1 folder. Having zero sum on level 1 folder is equivalent to having zero sum over the whole signal. T_2 preserves the sum over whole signal and it also preserves the property of being constant on level 2 folders. □

Example 4.1. A concrete example of T_2 is given here, based on Figure 2.1 tree structure.

$$T_2 = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1/2 & 1/2 & 1/2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1/2 & 1/2 & 1/2 \\ 1/4 & 1/4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/4 & 1/4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/4 & 1/4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/4 & 1/4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/3 & 1/3 & 1/3 & 1/3 & 0 & 0 & 0 \\ 0 & 0 & 1/3 & 1/3 & 1/3 & 1/3 & 0 & 0 & 0 \\ 0 & 0 & 1/3 & 1/3 & 1/3 & 1/3 & 0 & 0 & 0 \end{pmatrix} \tag{4.2}$$

Next is an analogue to the scaling operator. This again depends on tree structure and is level specific.

Definition 4.5 (Scaling operator)**.** At level l of a hierarchical tree defined in Chapter 2, we have folders $X_{l,1}, \ldots, X_{l,n_l}$ with sizes $m_{l,1}, \ldots, m_{l,n_l}$. The scaling operator S_l is

$$\begin{pmatrix} A(m_{l,1}, m_{l,1}) & 0 & \cdots & 0 \\ 0 & A(m_{l,2}, m_{l,2}) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & A(m_{l,n_l}, m_{l,n_l}) \end{pmatrix} \tag{4.3}$$

The scaling operator calculates the average of a folder at level l and puts it into each node in this folder. Some properties follow.

Theorem 4.4. Let $f \in V_l$ be a signal. Let S_m be the scaling operator (Definition 4.5).

(i) If $l \leq m$, then $S_m f = f$.

(ii) if $l > m$, then $S_m f \in V_m$.

(iii) S_m is a projection in $V = V_L$.

(iv) $S_m f \in V_m$ for any signal f and any m.

Proof. When $l \leq m$, f is constant on level l and thus level m folders, so the average matrices $A(\cdot,\cdot)$ change nothing. By definition, the image of S_m is always constant on level m folders, so $S_m f \in V_m$ holds for any signal f. Now $S_m f \in V_m$, so $S_m S_m f = f$ for any signal f. This means S_m is idempotent and a projection. □

Theorem 4.5. The translation operator T_l (Definition 4.4) is cyclic with a cycle length n_l.

(i) $T_l^{n_l} = S_l$;

(ii) $T_l^{n_l+1} = T_l$;

(iii) $f \in V_l$, $T_l^{n_l} f = f$.

Proof. $T_l^2 =$

$$\begin{pmatrix} 0 & 0 & \cdots & 0 & A(m_{l,1}, m_{l,n_l}) \\ A(m_{l,2}, m_{l,1}) & 0 & \cdots & 0 & 0 \\ 0 & A(m_{l,3}, m_{l,2}) & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & A(m_{l,n_l}, m_{l,n_l-1}) & 0 \end{pmatrix}^2 =$$

$$\begin{pmatrix} 0 & \cdots & 0 & A(m_{l,1}, m_{l,n_l-1}) & 0 \\ 0 & \cdots & 0 & 0 & A(m_{l,2}, m_{l,n_l}) \\ A(m_{l,3}, m_{l,1}) & \cdots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & A(m_{l,n_l}, m_{l,n_l-2}) & 0 & 0 \end{pmatrix}$$

Each multiplication moves the column blocks to the left by one block, while the column sizes are fixed for the blocks. By induction, we get $T_l^{n_l} = S_l$ and $T_l^{n_l+1} = T_l$. When $f \in V_l$, $T_l^{n_l} f = S_l f = f$ by the Theorem 4.4. □

Example 4.2. A concrete example of S_2 is given here, based on Figure 2.1 tree structure.

$$S_2 = \begin{pmatrix} 1/2 & 1/2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1/2 & 1/2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/4 & 1/4 & 1/4 & 1/4 & 0 & 0 & 0 \\ 0 & 0 & 1/4 & 1/4 & 1/4 & 1/4 & 0 & 0 & 0 \\ 0 & 0 & 1/4 & 1/4 & 1/4 & 1/4 & 0 & 0 & 0 \\ 0 & 0 & 1/4 & 1/4 & 1/4 & 1/4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1/3 & 1/3 & 1/3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1/3 & 1/3 & 1/3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1/3 & 1/3 & 1/3 \end{pmatrix} \tag{4.4}$$

4.2 Generating Wavelets and Spaces

With the translation and scaling operators defined in Definitions 4.4 and 4.5, father and mother wavelets can be generated at various levels. These wavelets will form the bases of nested spaces V_l. We are able to achieve this in a way very similar to the classic case.

1. Let $\phi = (1, 0, \ldots, 0) \in V$. This is our starting point. Set $\phi_{L,1} = \phi$.

2. $\phi_{L,k}$ are generated by taking translations of $\phi_{L,1}$ on level L.

$$\phi_{L,k} = T_L^{k-1}\phi_{L,1}, k = 1, 2, \ldots, N \tag{4.5}$$

3. They form a basis for V_L. (Proof in Theorem 4.6 below.)

$$V_L = \text{span}(\phi_{L,k} | k = 1, 2, \ldots, N) \tag{4.6}$$

4. $\phi_{L-1,1} = S_{L-1}\phi_{L,1}$ gives a father wavelet on level $L-1$ and we continue with the above steps until reaching level 1, the root node.

5. In general, we have

$$\phi_{l,1} = S_l \phi_{l+1,1} \tag{4.7}$$

$$\phi_{l,k} = T_l^{k-1} \phi_{l,1}, k = 1, 2, \ldots, n_l \tag{4.8}$$

$$V_l = \text{span}(\phi_{l,k} | k = 1, 2, \ldots, n_l) \tag{4.9}$$

Theorem 4.6 (Father wavelets as basis)**.** Let $\{\phi_{l,k} | k = 1, 2, \ldots, n_l\}$ be generated from the above process.

(i) They are the same functions as defined in (3.3).

(ii) They form a basis for V_l.

Proof. In the above construction, we have

$$\phi_{l,1} = S_l \phi_{l+1,1} = S_l \phi_{L,1} = S_l \phi = \frac{1}{m_{l,1}} \mathbf{1}_{X_{l,1}} \tag{4.10}$$

The translation operator on level l moves this block to the next one while preserving the sum, thus

$$\phi_{l,2} = T_l \phi_{l,1} = T_l \frac{1}{m_{l,1}} \mathbf{1}_{X_{l,1}} = \frac{1}{m_{l,2}} \mathbf{1}_{X_{l,2}} \tag{4.11}$$

By induction, we obtain

$$\phi_{l,k} = T_l^{k-1} \phi_{l,1} = \frac{1}{m_{l,k}} \mathbf{1}_{X_{l,k}} \tag{4.12}$$

The characteristic functions on folders form the natural basis of V_l, which is the collection of all functions that are constant on level l folders. Now each father wavelet is a scalar multiple of a characteristic function on each folder,

so they also form a basis for V_l. □

The mother wavelets introduced in Chapter 3 is a stretched and averaged version of Haar wavelets. In order to generate them, we first define a weight function.

Definition 4.6 (Weight function)**.** Given l and k, suppose folder $X_{l-1,k}$ splits into $u_{l-1,k}$ subfolders $\{Y_{l,k,j}\}, j = 1, 2, \ldots, u_{l-1,k}$ on level l. When $n < u_{l-1,k}$ and $j \leq n$, the weight function is defined as

$$w(l, k, j, n) = \frac{|Y_{l,k,n}|}{\sum_{i=1}^{j} |Y_{l,k,i}|} \tag{4.13}$$

This gives the relative weight in folder $X_{l-1,k}$ of n-th subfolder among first j subfolders on level l.

In the above definition, $Y_{l,k,1}$ (the first subfolder split from $X_{l-1,k}$) is a folder on level l. Let ϕ_{l,k_0} be the father wavelet generated in (4.8) for this folder. We can build our mother wavelets as follows.

$$\begin{aligned} \psi_{l,k,j} &= \sum_{n=1}^{j} w(l, k, j, n) T_l^{n-1} \phi_{l,k_0} - T_l^j \phi_{l,k_0} \\ &\text{for } j = 1, 2, \ldots, u_{l-1,k} - 1; k = 1, 2, \ldots, n_{l-1}; l = 2, 3, \ldots, L \end{aligned} \tag{4.14}$$

For simplicity, we write the above process in one operator $\psi_{l,k,j} = G_{l,k,j}\phi_{l,k_0}$. Note that $G_{l,k,j}$ is a composite of scaling, translation and linear combination.

Example 4.3. A concrete example of how the mother wavelets are generated is given here, based on Figure 2.1 tree structure. There is only one folder on level 1 (the root node) and it splits into 3 folders on level 2 with sizes 2, 4 and

3. The weight function is

$$w(2,1,2,1) = 2/(2+4) = 1/3 \text{ and } w(2,1,2,2) = 4/(2+4) = 2/3 \tag{4.15}$$

The first father wavelet for this folder is

$$\phi_{2,1} = (1/2, 1/2, 0, 0, 0, 0, 0, 0, 0) \tag{4.16}$$

The mother wavelets are

$$\begin{aligned} \psi_{2,1,1} &= \phi_{2,1} - \phi_{2,2} \\ &= (1/2, 1/2, -1/4, -1/4, -1/4, -1/4, 0, 0, 0) \\ \psi_{2,1,1} &= 1/3\phi_{2,1} + 2/3\phi_{2,2} - \phi_{2,3} \\ &= (1/6, 1/6, 1/6, 1/6, 1/6, 1/6, -1/3, -1/3, -1/3) \end{aligned} \tag{4.17}$$

Theorem 4.7. The system in (4.14) generates the mother wavelets defined in (3.4).

Proof. By definition, weight function w recalculates the average and evenly distributes them among the subfolders, while preserving the sum of the function. By Definition 4.4, the translation operator also preserves the sum of the function. From the construction of $\phi_{l,k}$ in (4.7) and (4.8), each of them has sum equal to 1. Therefore, $\sum_{n=1}^{j} w(l,k,j,n) T_l^{n-1} \phi_{l,k_0}$ evenly distributes weights among the first j subfolders of $X_{l-1,k}$ and keeps the sum equal to 1. This gives the same result as the positive part defined in (3.4). Similarly, $-T_l^j \phi_{l,k_0}$ gives the negative part in (3.4) and it also has sum equal to 1. □

Theorem 4.8 (Mother wavelets as basis)**.** Let $\{\psi_{l,k,j} | j = 1, 2, \ldots, u_{l-1,k} -$

$1; k = 1, 2, \ldots, n_{l-1}\}$ be generated in (4.14). Then they form a basis for W_{l-1} for $l = 2, 3, \ldots, L$.

Proof. By (4.14), each $\psi_{l,k,j}$ is a linear combination of $\phi_{l,k}$. So $\psi_{l,k,j} \in V_l$. By Theorem 3.3, each $\psi_{l,k,j}$ is orthogonal to V_{l-1}. Thus, $\text{span}(\psi_{l,k,j}) \subset W_{l-1}$, which is defined as the orthogonal complement of V_{l-1} in V_l.

On the other hand, $\dim(V_l) = n_l$ and $\dim(V_{l-1}) = n_{l-1}$. So $\dim(W_{l-1})$ $= \dim(V_l) - \dim(V_{l-1}) = n_l - n_{l-1}$. From the proof of Lemma 3.1, we have $|\{\psi_{l,k,j} | j = 1, 2, \ldots, u_{l-1,k} - 1; k = 1, 2, \ldots, n_{l-1}\}| = n_l - n_{l-1}$. By Lemma 3.2, $\psi_{l,k,j}$ are mutually orthogonal and thus linearly independent. This means $\{\psi_{l,k,j} | j = 1, 2, \ldots, u_{l-1,k} - 1; k = 1, 2, \ldots, n_{l-1}\}$ are $n_l - n_{l-1}$ linearly independent functions in W_{l-1}, whose dimension is $n_l - n_{l-1}$. Therefore, they form a basis of W_{l-1}. □

A summary of the above process is given below. And it is shown in Figure 4.1.

1. Begin with a model function $\phi = (1, 0, \ldots, 0) \in V$.
2. Use scaling operator to create the first father wavelet for each level $\phi_{l,1} = S_l\phi$.
3. Repeatedly translate the first father wavelet function at each level to generate all father wavelets $\phi_{l,k} = T_l^{k-1}\phi_{l,1}$.
4. $\{\phi_{l,k}\}$ are the basis that spans V_l.
5. Apply $G_{l,k,j}$ (as defined in (4.14)), which is a combination of scaling and

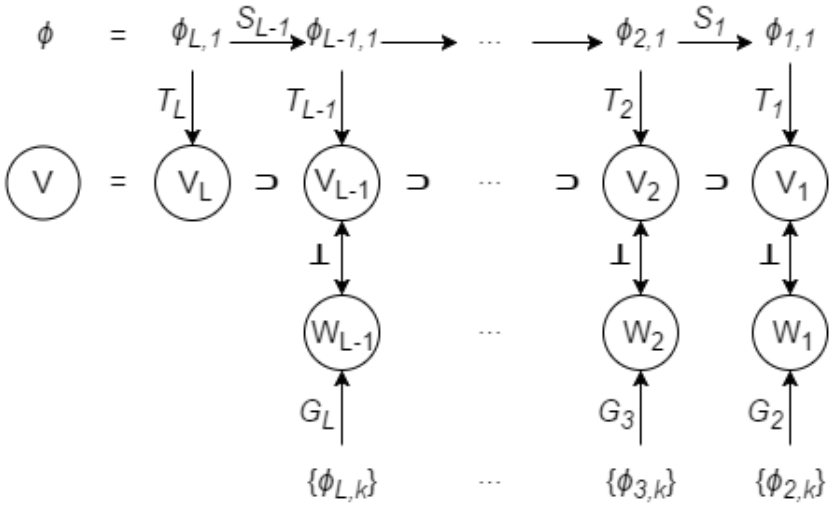

Figure 4.1: Tree based nested spaces generated by translation, scaling, and their orthogonal complements

translation, on selected father wavelets (those that appears on the first subfolder) to obtain mother wavelets $\psi_{l,k,j} = G_{l,k,j}\phi_{l,k_0}$.

6. $\{\psi_{l,k}\}$ are the basis that spans W_l.

4.3 Multiresolution

With translation and scaling on hierarchical tree properly defined, now we can define a multiresolution analysis on hierarchical tree.

Definition 4.7 (Multiresolution)**.** Let a hierarchical tree structure and its signal space V be defined as in Chapter 2. An orthogonal multiresolution analysis on the hierarchical tree is a signal (father wavelet function) $p \in V$, together with the translation and scaling operators defined in Definitions 4.4 and 4.5, that generates father wavelets $p_{l,k}$, mother wavelets $q_{l,k}$, nested spaces

P_l, orthogonal complements Q_l and satisfies the following conditions.

$$\begin{aligned} p_{l,k} &= \text{finite linear combination of } \{T_l^{k-1} S_l p | k = 1, 2, \ldots, n_l\} \\ &\text{for } k = 1, 2, \ldots, n_l \text{ and } l = 1, 2, \ldots, L \end{aligned} \tag{4.18}$$

$$\begin{aligned} q_{l,k} &= \text{finite linear combination of } \{T_l^{k-1} S_l p | k = 1, 2, \ldots, n_l\} \\ &\text{for } k = 1, 2, \ldots, n_l \text{ and } l = 2, 3, \ldots, L \end{aligned} \tag{4.19}$$

$$P_l = \text{span}(p_{l,k}) \text{ for } l = 1, 2, \ldots, L \tag{4.20}$$

$$Q_l = \text{span}(q_{l+1,k}) \text{ for } l = 1, 2, \ldots, L-1 \tag{4.21}$$

$$\{\text{constant signals on } X\} = P_1 \subset P_2 \subset \cdots \subset P_L = V \tag{4.22}$$

$$Q_l \text{ is the orthogonal complement of } P_l \text{ in } P_{l+1} \tag{4.23}$$

Some remarks on the above definition.

- In this general definition, we used p, q, P and Q in place of ϕ, ψ, V and W to avoid reusing of symbols.
- For a finite graph like a hierarchical tree, the signal space is finite dimensional. So we only need a finite number of nested spaces.
- Any finite linear combination can be used. This allows more choices in wavelet construction.

Theorem 4.9. In Definition 4.7, nested subspace $P_l \subset V_l$ and detail subspace $Q_l \subset V_{l+1}$, where V_l is defined in Definition 4.1.

Proof. For any function $p \in V$, by Theorem 4.4, $S_l p \in V_l$. So their translations

are also in V_l. Thus any linear combination of scaling and translation is also in V_l. Likewise, $Q_l \in V_{l+1}$, since it is spanned by level $l+1$ father wavelets. This means that the V_l in Definition 4.1 is the maximum possible choice under Definition 4.7. □

Theorem 4.10 (Multiresolution)**.** We have an orthogonal multiresolution analysis on hierarchical tree (Definition 4.7) with the following choices.

(i) Father wavelet function $\phi = (1, 0, \ldots, 0) \in V$;

(ii) System generated by (4.7), (4.8) and (4.14);

(iii) The nested spaces $\{V_l\}$ (Definition 4.1) and their orthogonal complements $\{W_l\}$ (Definition 4.2).

Proof. The generating process in (4.7), (4.8) and (4.14) only uses linear combination of scaling and translation, so (4.18) and (4.19) are satisfied. Theorems 4.6 and 4.8 lead to (4.20), (4.21) and (4.23). With $P_l = V_l$ as in Definition 4.1, (4.22) is also satisfied. □

By Theorems 4.6 and 4.7, this also means the wavelets we constructed in Chapter 3 generates a multiresolution analysis.

Since the mother wavelets generate orthogonal complement spaces at each level, the rows of transform matrix can be divided into blocks that calculate details at each level. For example, the bottom block $\{\psi_{L,k,j}\}$ acts on an input signal and yields bottom level details for the output (transformed signal). As we go up in the transform matrix, we get coarser details at higher

level. This continues until we reach the top row, which gives the average of the entire input signal. The transformed signal has the structure that as we go down we get more and more detailed information. As an analogue to the classic case, if we can accept some loss of signal quality, we can throw away bottom level details to obtain higher data transfer rate. However, we can choose proper wavelets to control the loss at an acceptable level. Figure 4.2 summarizes this.

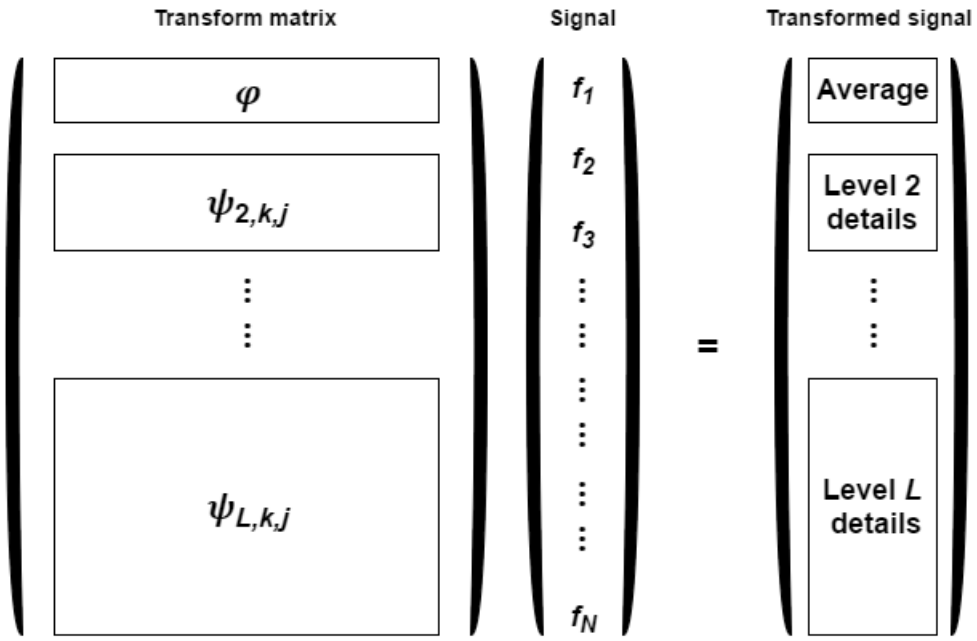

Figure 4.2: Average and details in the transformed signal

4.4 Comparison with Classic Multiresolution

During the construction of a multiresolution analysis, many good properties can be carried over from the classic case. They are listed below.

- Nested subspaces and their orthogonal complements are closed subspaces. (Lemma 4.1)
- Nested spaces are invariant under translation. (Lemma 4.2)
- Scaling maps functions at one level to the next (coarser) level. (Theorem 4.4)
- Translation is cyclic. (Theorem 4.5) On the bottom level, it becomes the normal cyclic permutation on a periodical signal. All the matrices $A(\cdot, \cdot)$ become a single 1.
- Father wavelets are similar to step functions. Mother wavelets are similar to difference of step functions and have zero sum.
- Although the process is more complicated than the classic case, all father and mother wavelets are generated by scaling, translation and linear combination of a single generator. (Theorems 4.6 and 4.8; Also Definition 4.7)
- Father wavelets are mutually orthogonal at each level, since they are scalar multiples of characteristic functions.
- Mother wavelets are mutually orthogonal at each level. (Lemma 3.2)
- The transform matrix has mutually orthogonal rows and we can calculate

its inverse easily. (Theorems 3.4 and 3.5)

- With a perfect binary tree, our construction essentially gives the Haar matrix. (Theorem 4.11 below)

Theorem 4.11. With a perfect binary tree, the wavelets generated by $\phi = (1, 0, \ldots, 0) \in V$, (4.7), (4.8) and (4.14) form a Haar matrix (Example 2.3) with different scalar multiples on rows.

Proof. With a perfect binary tree, each non bottom folder has exactly two subfolders. Thus the weight function is always 1 and j can only be 1 in (4.14). $\psi_{L,k,1} = \phi_{L,k_0} - T_L\phi_{L,k_0}$ is vector $(1, -1)$ put in the corresponding position of the folder. These are the bottom level wavelets and they are the same as the bottom half rows of the Haar Matrix.

On the next level up, we have folders of size four. The wavelets are vectors $(1/2, 1/2, -1/2, -1/2)$ put in the corresponding position of the folder, which differs from the Haar Matrix by a scalar of $1/2$. By induction, on level l the difference is a scalar of 2^{l-l}. □

There are also some limitations in the setting of hierarchical tree.

- Due to the complexity and variations of tree structure, our operators and wavelets are mostly level and structure specific.
- For father wavelets, we have constructed them in a bottom up way. That is, we start from the most detailed space and generate coarser and coarser spaces. In the classic case, the scaling can be done in two way. But our

approach only achieves one way scaling.

- For mother wavelets, we are unable to find a proper way to define scaling for them, due to the uncertainty in tree structure.
- The generating process for the mother wavelets is complex and requires calculation of weight function at every folder. We attempt to fix this in the next chapter.

4.5 Classification

In this section, we work on a brief classification of all possible wavelets that generate orthogonal multiresolution (Definition 4.7). In Theorem 4.10, we have offered a constructive approach to multiresolution wavelets. It is part of our separate classification result below. By comparison to the case of classic wavelets in $L^2(\mathbb{R}^d)$, it is interesting to note that there is not an analogous known classification for the classic wavelet families, not even for the case $d = 1$. See [7], [13] and [28]. Both Theorem 4.10 and our parallel classification result are new.

For simplicity, we require that the nested subspaces V_l be defined as in Definition 4.1. Therefore the detail spaces W_l are as in Definition 4.2.

We also require that father wavelets are generated by (5.1) and (5.2), from $\phi = (1, 0, \ldots, 0) \in V$. Therefore, the father wavelets are piecewise constant functions on level l folders, as defined in (4.7) and (4.8). With these conditions, (4.18), (4.20) and (4.22) in Definition 4.7 are satisfied.

Now fix level l and consider potential mother wavelets $\{q_{l,n}\}$ that satisfy (4.19) and (4.21). In proof of Theorem 4.8, we have shown that $\dim(W_{l-1}) = \dim(V_l) - \dim(V_{l-1}) = n_l - n_{l-1}$. Therefore, we need exactly $n_l - n_{l-1}$ mother wavelets to form a basis (linearly independent functions) for W_{l-1}.

On the other hand, by (4.19) $q_{l,n}$ must be a linear combination of level l father wavelets. Therefore, potential mother wavelets must have the form

$$q_{l,n} = \sum_{k=1}^{n_l} a_{l,n,k}\phi_{l,k} \tag{4.24}$$

for $n = 1, 2, \ldots, n_l - n_{l-1}$. Here the coefficients $a_{l,n,k}$, depending on l, n and k, are some constants to be determined.

Written in matrix form, this becomes

$$\begin{pmatrix} q_{l,1} \\ \vdots \\ q_{l,n_l-n_{l-1}} \end{pmatrix} = \begin{pmatrix} a_{l,1,1} & \cdots & a_{l,1,n_l} \\ \vdots & \vdots & \vdots \\ a_{l,n_l-n_{l-1},1} & \cdots & a_{l,n_l-n_{l-1},n_l} \end{pmatrix} \begin{pmatrix} \phi_{l,1} \\ \vdots \\ \phi_{l,n_l} \end{pmatrix} \tag{4.25}$$

We denote the coefficient matrix by A_l, which is an $n_l - n_{l-1}$ by n_l matrix with $A_l(n,k) = a_{l,n,k}$.

In order for $\{q_{l,n}\}$ to form a basis, A_l must have full rank, or equivalently, its rows must be linearly independent.

By (4.23) in Definition 4.7, we also require each $q_{l,n}$ orthogonal to level $l-1$ father wavelets. This is equivalent to the condition that the rows of A_l have zero sum on level $l-1$ folders. That is

$$\sum_{X_{l,k} \subset X_{l-1,j}} a_{l,n,k} = 0 \tag{4.26}$$

for $n = 1, 2, \ldots, n_l - n_{l-1}$ and for $j = 1, 2, \ldots, n_{l-1}$.

In summary, all possible mother wavelets that yield orthogonal multiresolution (Definition 4.7) are given in the form of (4.24) or (4.25), for $l = 2, 3, \ldots, L$, and the following conditions are satisfied.

(i) A_l in (4.25) has full rank. Or equivalently, its rows are linearly independent.

(ii) (4.26) is satisfied.

Finally, we point out that the above analysis is theoretical. In applications, it is usually more efficient to construct mother wavelets in a similar fashion across all levels. Further, it is often better to construct mother wavelets such that each $q_{l,n}$ is only supported on a certain level $l-1$ folder (or some other local support). We have done so with the choice in Chapter 3 and will do so again with the choice in Chapter 5.

CHAPTER 5
A SECOND WAVELET BASIS

In building of wavelets in Chapter 3 and construction of multiresolution analysis in Chapter 4, we find it complicated to define the mother wavelets proposed in [17]. The main problem is that (4.14) requires repeated calculation of average on an uncertain number of folders to generate mother wavelets. In applications, this means many potentially unnecessary evaluation steps, which leads to inefficient algorithm.

Motivated by this, we modify the wavelets proposed in [17] and obtain a new kind of wavelet basis on hierarchical tree. We will show that this also yields an orthogonal multiresolution analysis with more clear meaning.

5.1 Construction

Let the tree structure be defined as in Chapter 2. Let the nested spaces V_l and their orthogonal complements be as in Definitions 4.1 and 4.2. Let the translation and scaling operations be as in Definitions 4.4 and 4.5. The father wavelets $\phi_{l,k}$ are unchanged.

1. Let $\phi = (1, 0, \ldots, 0) \in V$. Set $\phi_{L,1} = \phi$.

2. Father wavelets are generated recursively as

$$\phi_{l,1} = S_l \phi_{l+1,1} \text{ for } l = 1, 2, \ldots, L-1 \tag{5.1}$$

$$\phi_{l,k} = T_l^{k-1} \phi_{l,1} \text{ for } k = 1, 2, \ldots, n_l \tag{5.2}$$

Mother wavelets are defined in a similar way to Chapter 4. Suppose folder $X_{l-1,k}$ splits into subfolders $\{Y_{l,k,j}\}, j = 1, 2, \ldots, u_{l-1,k}$ on level l. $Y_{l,k,1}$ is the first subfolder, which is also a level l folder. Let the ϕ_{l,k_0} be the father wavelet generated in (5.1) and (5.2) for this folder. We build new mother wavelets as

$$\psi_{l,k,j} = \sum_{n=1}^{j} T_l^{n-1}\phi_{l,k_0} - jT_l^j\phi_{l,k_0} \tag{5.3}$$

Compared with the previous mother wavelets given in (4.14), we leave out the weight function. As a result, each subfolder (as opposed to each leaf node) is treated equally when it is used to generate a mother wavelet. As we will discuss later in this chapter, this change not only simplifies the expression, but also gives more efficient algorithm and leads to possibility of graph structure modification.

Example 5.1. A concrete example based on Figure 2.1 of the transform matrix M of the new wavelet basis is given below in (5.4). For comparison, (3.9) is also listed below. For a general program to calculate the transform matrix used in this chapter, see Appendix A.2.

$$M = \begin{pmatrix} 1/9 & 1/9 & 1/9 & 1/9 & 1/9 & 1/9 & 1/9 & 1/9 & 1/9 \\ 1/2 & 1/2 & -1/4 & -1/4 & -1/4 & -1/4 & 0 & 0 & 0 \\ 1/2 & 1/2 & 1/4 & 1/4 & 1/4 & 1/4 & -2/3 & -2/3 & -2/3 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & -3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & -2 \end{pmatrix} \tag{5.4}$$

$$M = \begin{pmatrix} 1/9 & 1/9 & 1/9 & 1/9 & 1/9 & 1/9 & 1/9 & 1/9 & 1/9 \\ 1/2 & 1/2 & -1/4 & -1/4 & -1/4 & -1/4 & 0 & 0 & 0 \\ 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & 1/6 & -1/3 & -1/3 & -1/3 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 1/2 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/3 & 1/3 & 1/3 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1/2 & 1/2 & -1 \end{pmatrix} \tag{3.9}$$

Similar to (3.9), (5.4) is also a stretched version of Haar matrix. The last six rows are blocked, representing bottom level mother wavelets. Every transform matrix generated by (5.1) - (5.3) will have the bottom rows in this block form. These blocks can be seen as the basic building elements of the wavelets, similar to the vector $(1, -1)$ for discrete Haar transform.

Compared with (3.9), the first two rows of (5.4) are exactly the same and the last six rows only differ by a scalar. The third row in (5.4), however, is not a scalar of the third row in (3.9), due to different construction process. In (5.4), the second and the third rows are not orthogonal. This gives an example that mother wavelets within the same level, as we define in this chapter, are not necessarily orthogonal to each other.

5.2 Multiresolution

Most properties of the wavelet defined in Chapter 3 can be easily carried over to the new one. We list them here and give proofs when necessary.

Lemma 5.1 (Father wavelets as basis)**.** The father wavelets $\{\phi_{l,k} | k = 1, 2, \ldots, n_l\}$ given in (5.1) and (5.2) form a basis of V_l as in Definition 4.1.

Proof. We make no change on father wavelets, so this is obvious. □

Theorem 5.2. Let mother wavelet $\psi_{l,k,j}$ be given in (5.3) and father wavelet $\phi_{m,k}$ be given in (5.1) and (5.2). If $m < l$, then $\psi_{l,k,j}$ is orthogonal to $\phi_{m,k}$.

Proof. Both $\phi_{m,k}$ and $\psi_{m,k,j}$ are linear combinations of $\phi_{m,1}$, thus they are constant on level m folders. From (5.3), $\psi_{l,k,j}$ is supported on a level l subfolder. Therefore, when $m < l$, both $\phi_{m,k}$ and $\psi_{m,k,j}$ are constant on the support of $\psi_{l,k,j}$.

Also from (5.3), ϕ_{l,k_0} has component sum equal to 1 and translation preserves component sum. Thus the component sum of $\psi_{l,k,j}$ is $j - j = 0$. Since $\psi_{l,k,j}$ is supported on a level l subfolder with zero component sum, it is orthogonal to any function constant on level m folders when $m < l$. □

Theorem 5.3 (Mother wavelets as basis)**.** Let $\{\psi_{l,k,j} | j = 1, 2, \ldots, u_{l-1,k} - 1; k = 1, 2, \ldots, n_{l-1}\}$ be given in (5.3). They form a basis for W_{l-1} (Definition 4.2) for $l = 2, 3, \ldots, L$.

Proof. By Theorem 5.2, $\psi_{l,k,j}$ is orthogonal to V_{l-1}. By (5.3), $\psi_{l,k,j} \in V_l$. Therefore, $\text{span}(\psi_{l,k,j} | j = 1, 2, \ldots, u_{l-1,k} - 1; k = 1, 2, \ldots, n_{l-1}) \subset W_{l-1}$.

Similar to the proof of Theorem 4.8, we have $\dim(W_{l-1}) = \dim(V_l) - \dim(V_{l-1}) = n_l - n_{l-1}$. And the same number of mother wavelets $|\{\psi_{l,k,j} | j = 1, 2, \ldots, u_{l-1,k} - 1; k = 1, 2, \ldots, n_{l-1}\}| = n_l - n_{l-1}$.

Now it suffices to show that $\{\psi_{l,k,j}\}$ are linearly independent for fixed l. When $k_1 \neq k_2$, the support of $\psi_{l,k_1,j}$ and $\psi_{l,k_2,j}$ are disjoint. So we only need to show linear independence for fixed l and fixed k.

Put the vectors $\{\psi_{l,k,j}|j=1,2,\ldots,u_{l-1,k}-1\}$ as rows in a matrix. By (5.3) and let $s=u_{l-1,k}-1$, we have

$$\begin{pmatrix} \phi_{l,k_0}-T_l\phi_{l,k_0} \\ \phi_{l,k_0}+T_l\phi_{l,k_0}-2T_l^2\phi_{l,k_0} \\ \vdots \\ \phi_{l,k_0}+\cdots+T_l^{s-1}\phi_{l,k_0}-sT_l^s\phi_{l,k_0} \end{pmatrix} = \begin{pmatrix} 1 & -1 & 0 & 0 & \ldots & 0 \\ 1 & 1 & -2 & 0 & \ldots & 0 \\ & & \vdots & & & \\ 1 & 1 & \ldots & 1 & 1 & -s \end{pmatrix} \begin{pmatrix} \phi_{l,k_0} \\ T_l\phi_{l,k_0} \\ \vdots \\ T_l^s\phi_{l,k_0} \end{pmatrix} \tag{5.5}$$

Since $\phi_{l,k_0}, T_l\phi_{l,k_0}, \ldots, T_l^s\phi_{l,k_0}$ are father wavelets (step functions) on mutually disjoint folders, they are linearly independent. So we only need to show the matrix

$$\begin{pmatrix} 1 & -1 & 0 & 0 & \ldots & 0 \\ 1 & 1 & -2 & 0 & \ldots & 0 \\ & & \vdots & & & \\ 1 & 1 & \ldots & 1 & 1 & -s \end{pmatrix} \tag{5.6}$$

has full rank. Deleting the first column of this matrix and we get a triangular matrix whose determinant is $\prod_{i=1}^{s}(-i)\neq 0$. This completes the proof. □

Similar to Chapter 3, we write the transform matrix in block form as

$$M=\begin{pmatrix}\phi_{1,1}\\ \psi_{l,k,j}\end{pmatrix}=\begin{pmatrix}\phi_{1,1}\\ (\psi_{2,k,j})\\ \vdots\\ (\psi_{L,k,j})\end{pmatrix} \tag{5.7}$$

$$j=1,2,\ldots,u_{l-1,k}-1; k=1,2,\ldots,n_{l-1}; l=2,3,\ldots,L$$

The mother wavelets $\{\psi_{l,k,j}\}$ are arranged in increasing order of l, k and j, with priority decreasing from l to k to j.

Theorem 5.4. The transform matrix given in (5.7) is invertible.

Proof. By Theorem 5.3, mother wavelets $\{\psi_{l,k,j}\}$ within each block are linearly independent. By Theorem 5.2, the blocks $\{\psi_{l,k,j}\}$ for $j = 1, 2, \ldots, u_{l-1,k} - 1; k = 1, 2, \ldots, n_{l-1}; l = 2, 3, \ldots, L$ and $\phi_{1,1}$ are mutually orthogonal. So the rows of the transform matrix are linearly independent. □

Finally, we point out a major difference between the two sets of wavelets defined in Chapter 3 and in this chapter. In Chapter 3, the mother wavelets are mutually orthogonal, between different levels or within the same level. This makes the transform matrix have mutually orthogonal rows. Therefore it becomes an orthogonal matrix after normalization (See Lemmas 3.2, 3.3 and Theorems 3.4, 3.6). The mother wavelets defined in this chapter, however, are not necessarily orthogonal to each other within the same level (See Example 5.1). Two mother wavelets are still orthogonal when they come from different level (See Theorems 5.2 and 5.3). The transform is still invertible (See Theorem 5.4), but after normalization it no longer has the form of an orthogonal matrix. Nevertheless, we still have the most important property, multiresolution analysis.

Theorem 5.5 (Multiresolution)**.** We have an orthogonal multiresolution analysis on on hierarchical tree (Definition 4.7) with the following choices.

(i) Father wavelet function $\phi = (1, 0, \ldots, 0) \in V$;

(ii) System generated by (5.1) - (5.3);

(iii) The nested spaces $\{V_l\}$ (Definition 4.1) and their orthogonal complements $\{W_l\}$ (Definition 4.2).

Proof. The wavelet generating process in (5.1) - (5.3) clearly satisfies (4.18) and (4.19). (4.22) and (4.23) are satisfied under the definitions of $\{V_l\}$ and $\{W_l\}$ (Definitions 4.1 and 4.2). By Lemma 5.1, (4.20) holds. By Theorem 5.3, (4.21) holds. □

5.3 Encoding Algorithm

The wavelet transform defined in this chapter can be realized by efficient encoding algorithm. We now outline the steps below, starting from definition of two operators.

Definition 5.1 (Sum operator)**.** The sum operator is a 1 by n matrix

$$S_n = \mathbf{1}_{1\times n} \tag{5.8}$$

Definition 5.2 (Detail operator)**.** The detail operator D_n has two steps. Given an input of size n by 2 $\begin{pmatrix} x_1 & y_1 \\ \vdots & \vdots \\ x_n & y_n \end{pmatrix}$. In the first step, divide the first column by the second column to get a column vector of size n $\begin{pmatrix} x_1/y_1 \\ \vdots \\ x_n/y_n \end{pmatrix}$. The second step is to left multiply by an $n-1$ by n matrix

$$\begin{pmatrix} 1 & -1 & 0 & 0 & 0 & \dots & 0 \\ 1 & 1 & -2 & 0 & 0 & \dots & 0 \\ 1 & 1 & 1 & -3 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 1 & 1 & \dots & 1 & 1 & -(n-1) \end{pmatrix} \tag{5.9}$$

The information passed along in our algorithm is folder in the form of n by 2 matrix. The first column records the sum of input signal value on all leaf nodes contained in the folder and the second column records the number of leaf nodes contained in the folder. Left multiplication by the sum operator combines subfolders into their parent folder on the next level. The first step of the detail operator calculates the average of each folder. The second step gives the difference, or detail at this level. The advantage of the wavelets built in this chapter lies in that all folders (as opposed to single leaf nodes) at each level are treated equally. After sum operator has been applied, we pass the result to the next level and treat it as a single folder. This allows us to use the same sum and detail operators repeatedly at all levels. The only thing we care about is the size of input, which is the folder size, so that we can apply the operator with the correct size.

On level L, we treat each leaf node as folder of size 1 and rewrite the input signal $\begin{pmatrix} f_1 \\ \vdots \\ f_N \end{pmatrix}$ as $\begin{pmatrix} f_1 & 1 \\ \vdots & \vdots \\ f_n & 1 \end{pmatrix}$. We then group its components into level $L-1$ folders. Then for each folder, we apply the sum and detail operators of the correct size. The detail operator yields detail at this level. The sum operator combines folders into their parent folder on the next level. This is what we pass along to the next level. We continue this process until reaching the root node, where the final result is the sum of entire input signal and its size in the form of a 1 by 2 matrix. We can easily calculate the average of entire input signal.

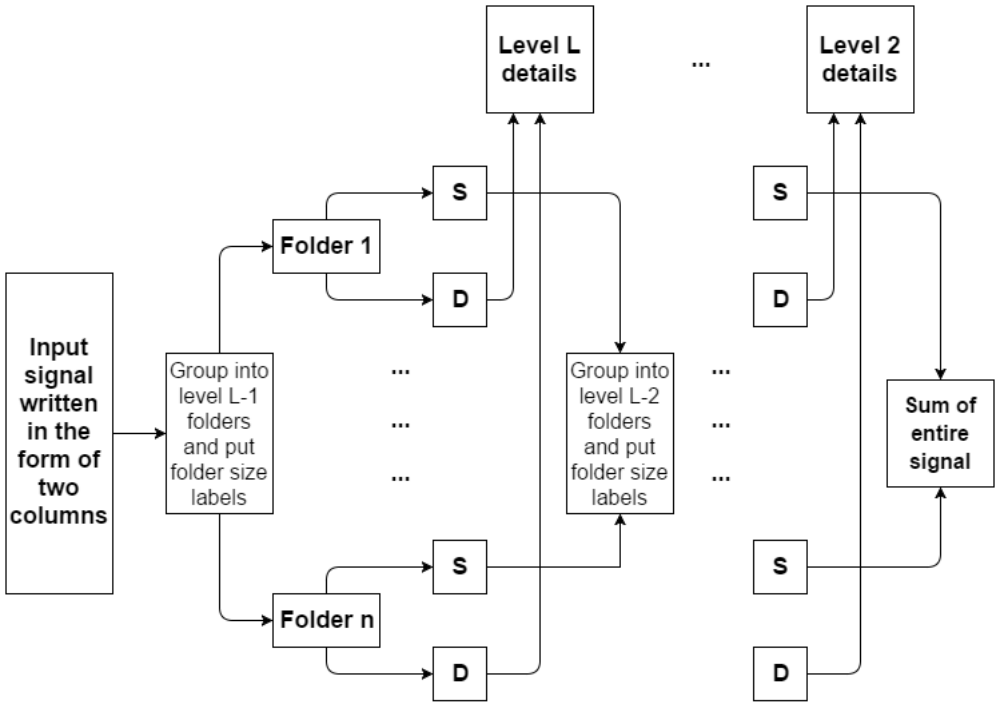

Figure 5.1: Encoding algorithm of wavelet transform given in Chapter 5

This process is summarized in a formal version as Algorithm 5.1. A flow chart for this algorithm is given in Figure 5.1.

5.4 Decoding Algorithm

For the reconstruction part, we want to go back from the average (or sum) of a folder plus the details of this folder to the average (or sum) of its subfolders. This process is not linear since we need to take care of subfolder sizes. As a result, we need to solve a linear system for each folder.

Definition 5.3 (Reconstruction operator)**.** Given input $m_1, m_2, \ldots, m_n$; s; $d_1, d_2, \ldots, d_{n-1}$. The reconstruction operator R solves the following linear

Algorithm 5.1 Tree Wavelet Encoding Algorithm

Task: Apply wavelet transform introduced in Chapter 5 to signal on hierarchical tree of level L (See Definitions 2.2 to 2.4).

Input: Signal f of size N; Tree structure specified by a sequence of vectors $u_l = (u_{l,1}, u_{l,2}, \ldots, u_{l,n_l})$ for $l = 1, 2, \ldots, L-1$ (See Definition 2.5).

Parameters: $s_{l,k}$ = sum of f over $X_{l,k}$; $m_{l,k} = |X_{l,k}|$; c is the cumulative counter for grouping; $M_{l-1,j}$ is the matrix for j-th folder at level $l-1$; $\left(d_{l-1,j}\right)$ is details in vector form of size $u_{l-1,j} - 1$; S and D are the sum and detail operators (See Definitions 5.1 and 5.2).

Initialization: $s_{L,k} = f_k$ and $m_{L,k} = 1$ for $k = 1, 2, \ldots, N$.

Main Iteration for $l = L, L-1, \ldots, 2$.

$c = 0$

Level l Iteration for $j = 1, 2, \ldots, n_{l-1}$

$M_{l-1,j} = \left(s_{l,k}, m_{l,k}\right)_{k=c+1,\ldots,c+u_{l-1,j}}$ %Group into next level folders.

$(s_{l-1,j}, m_{l-1,j}) = S_{u_{l-1,j}} M_{l-1,j}$ %Next level sum and folder size.

$\left(d_{l-1,j}\right) = D_{u_{l-1,j}} M_{l-1,j}$ %Details at this level.

$c = c + u_{l-1,j}$

End Level l Iteration

End Main Iteration

Output: Sum of entire signal is $s_{1,1}$; Average of entire signal is $s_{1,1}/m_{1,1}$; Level l details are $\left(d_{l-1,j}\right)$ for $j = 1, 2, \ldots, n_{l-1}$.

system for $a_1, a_2, \ldots, a_n$.

$$\begin{pmatrix} m_1 & m_2 & m_3 & \ldots & m_{n-1} & m_n \\ 1 & -1 & 0 & 0 & \ldots & 0 \\ 1 & 1 & -2 & 0 & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & 1 & \ldots & 1 & 1 & -(n-1) \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ \vdots \\ a_n \end{pmatrix} = \begin{pmatrix} s \\ d_1 \\ d_2 \\ \vdots \\ d_{n-1} \end{pmatrix} \tag{5.10}$$

$$R_n(m_1, m_2, \ldots, m_n; s; d_1, d_2, \ldots, d_{n-1}) = (a_1, a_2, \ldots, a_n) \tag{5.11}$$

When we use reconstruction operator R, given a folder, $m_1, m_2, \ldots, m_n$ are subfolder sizes; s is the sum over this folder; $d_1, d_2, \ldots, d_{n-1}$ are details of this folder. The output $a_1, a_2, \ldots, a_n$ are average over subfolders.

The general solution for the reconstruction operator is as follows.

$$a_1 = \frac{s + \sum_{k=2}^{n} D_k m_k}{\sum_{k=1}^{n} m_k} \text{ and } a_i = a_1 - D_i \text{ for } i = 2, \ldots, n \tag{5.12}$$

where

$$D_i = \frac{1}{i-1} d_{i-1} + \sum_{j=1}^{i-2} \frac{1}{j(j+1)} d_j \text{ for } i = 2, \ldots, n \tag{5.13}$$

Or, recursively,

$$D_2 = d_1 \text{ and } D_{i+1} = D_i + \frac{1}{i}(d_i - d_{i-1}) \text{ for } i = 2, \ldots, n-1 \tag{5.14}$$

At a fixed level, we are given the sum of a folder the associated details generated when this folder splits into its subfolders. Obtain the corresponding subfolder sizes from the tree structure and apply the reconstruction operator. We get the averages of signal over the subfolders and they can be converted into sums. Repeat this process over all folders at this level and then repeat

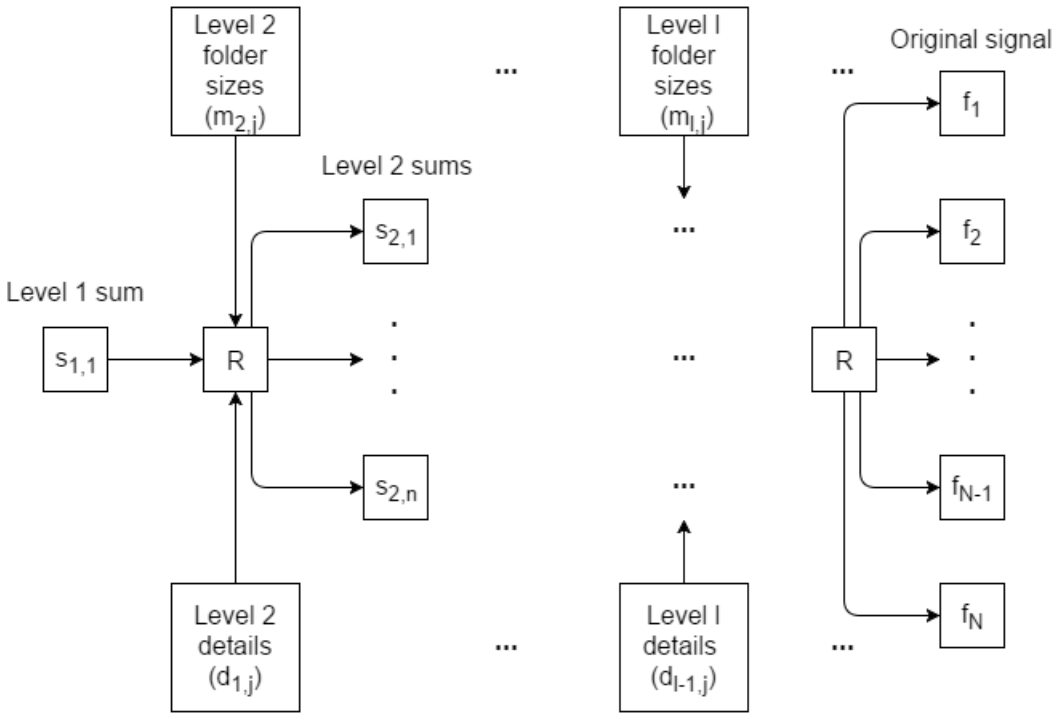

Figure 5.2: Decoding algorithm of wavelet transform given in Chapter 5

over all levels. The final result is bottom level sums over folder size 1, which is the original signal. For access to tree structure, we can use either folder size version (Definition 2.4) or vector sequence version (Definition 2.5). They can be converted into each other easily. For simplicity, we use both in the algorithm. This process is summarized in a formal version as Algorithm 5.2. A flow chart for this algorithm is given in Figure 5.2.

In applications, sometimes we want to stop at a certain level and use the average at this level for each folder as an approximate for the entire signal. With this decoding algorithm, we can perform this easily.

Algorithm 5.2 Tree Wavelet Decoding Algorithm

Task: Reconstruct wavelet (introduced in Chapter 5) encoded signal to original signal f on hierarchical tree of level L (See Definitions 2.2 to 2.4).

Input: Sum of entire signal $s_{1,1}$; Details at each level in vector form $\left(d_{l,j}\right)$ of size $u_{l,j} - 1$ (See Algorithm 5.1); Tree structure specified by folder sizes $\{m_{l,k}\}$ for $l = 1, 2, \ldots, L; k = 1, 2, \ldots, n_l$ (See Definitions 2.4) and a sequence of vectors $u_l = (u_{l,1}, u_{l,2}, \ldots, u_{l,n_l})$ for $l = 1, 2, \ldots, L-1$ (See Definition 2.5).

Parameters: $s_{l,k}$ = sum of f over $X_{l,k}$; $a_{l,k}$ = average of f over $X_{l,k}$; c is the cumulative counter for grouping; R is the reconstruction operator (See Definition 5.3).

Main Iteration for $l = 1, 2, \ldots, L-1$.

$c = 0$

Level l Iteration for $j = 1, 2, \ldots, n_l$

$$\left(a_{l+1,k}\right)_{k=c+1,\ldots,c+u_{l,j}} = R_{u_{l,j}}\left(\left(m_{l+1,k}\right)_{k=c+1,\ldots,c+u_{l,j}}; s_{l,j}; \left(d_{l,j}\right)\right)$$

%Reconstruct average for subfolders of $X_{l,j}$.

$s_{l+1,k} = m_{l+1,k}a_{l+1,k}$ %Convert average into sum.

$c = c + u_{l,j}$

End Level l Iteration

End Main Iteration

Output: Original signal $f_k = s_{L,k}$ for $k = 1, 2, \ldots, N$.

5.5 Estimate of Complexity

We briefly examine the time complexity of Algorithms 5.1 and 5.2 in this section.

For Algorithm 5.1, number of additions, multiplications and divisions during Level l Iteration is counted as follows. Subtraction is counted as addition since these operations are executed in similar manner. We follow the notation used in Algorithm 5.1.

1. $M_{l-1,j} = \left(s_{l,k}, m_{l,k} \right)_{k=c+1,\ldots,c+u_{l-1,j}}$

   No actual computation needed.

2. $(s_{l-1,j}, m_{l-1,j}) = S_{u_{l-1,j}} M_{l-1,j}$

   Additions: $2(u_{l-1,j} - 1)$.

3. $\left(d_{l-1,j} \right) = D_{u_{l-1,j}} M_{l-1,j}$

   Additions: $1 + 2 + \cdots + (u_{l-1,j}) = \frac{1}{2} u_{l-1,j}^2 - \frac{1}{2} u_{l-1,j}$;

   Multiplications: $u_{l-1,j} - 2$; Divisions: $u_{l-1,j}$.

4. $c = c + u_{l-1,j}$

   Additions: 1.

The above steps are iterated for $j = 1, 2, \ldots, n_{l-1}$ and then for $l = L, L-1, \ldots, 2$. Using Equation (2.2) $\sum_{k=1}^{n_l} u_{l,k} = n_{l+1}$, we obtain the total number of operations needed as following.

$$\text{Additions: } \sum_{l=L}^{2} \sum_{j=1}^{n_{l-1}} \left(\frac{1}{2} u_{l-1,j}^2 + \frac{3}{2} u_{l-1,j} - 1 \right) \tag{5.15}$$

$$\text{Multiplications: } \sum_{l=L}^{2}\sum_{j=1}^{n_{l-1}}(u_{l-1,j}-2)=\sum_{l=L}^{2}n_l-2(L-1) \tag{5.16}$$

$$\text{Divisions: } \sum_{l=L}^{2}\sum_{j=1}^{n_{l-1}}u_{l-1,j}=\sum_{l=L}^{2}n_l \tag{5.17}$$

Therefore, number of multiplications and divisions needed is approximately equal to the number of nodes in the tree $\sum_{l=1}^{L}n_l$.

For additions, we consider a "best case" scenario, the perfect k-ary tree. In this case, every node (except leaf nodes on the bottom level) has k child nodes. Thus we have

$$u_{l-1,j}=k \text{ and } n_l=k^{l-1} \tag{5.18}$$

The number of additions needed is

$$\begin{aligned}
&\sum_{l=L}^{2}\sum_{j=1}^{n_{l-1}}(\frac{1}{2}u_{l-1,j}^2+\frac{3}{2}u_{l-1,j}-1)\\
=&\sum_{l=L}^{2}\sum_{j=1}^{k^{l-2}}(\frac{1}{2}k^2+\frac{3}{2}k-1)\\
=&\sum_{l=L}^{2}(\frac{1}{2}k^l+\frac{3}{2}k^{l-1}-k^{l-2})\\
=&\frac{k^{L-1}-1}{k-1}(\frac{1}{2}k^2+\frac{3}{2}k-1)
\end{aligned} \tag{5.19}$$

Note that the bottom level L has k^{L-1} nodes, which is the number of input signal length N. Thus for fixed k, Equation (5.19) is of order $O(N)$.

In the case of perfect k-ary tree, Equations (5.16) and (5.17) can also

be simplified as

$$\text{Multiplications: } \sum_{l=L}^{2} n_l - 2(L-1) = \frac{k^L - 1}{k-1} - 2L + 1 \tag{5.20}$$

$$\text{Divisions: } \sum_{l=L}^{2} n_l = \frac{k^L - 1}{k-1} - 1 \tag{5.21}$$

Thus, in this case, number of multiplications and divisions are also of order $O(N)$.

Now we consider Algorithm 5.2. From Equations (5.12) to (5.14), number of additions, multiplications and divisions for a reconstruction operator of size n is counted as follows. Subtraction is counted as addition.

1. $D_2 = d_1$

   No actual computation needed.

2. $D_{i+1} = D_i + \frac{1}{i}(d_i - d_{i-1})$ for $i = 2, \ldots, n-1$

   Additions: $2(n-2)$; Divisions: $n-2$.

3. $$a_1 = \frac{s + \sum_{k=2}^{n} D_k m_k}{\sum_{k=1}^{n} m_k}$$

   Additions: $2(n-2)$; Multiplications: $n-1$; Divisions: 1.

4. $a_i = a_1 - D_i$ for $i = 2, \ldots, n$

   Additions: $n-1$.

5. Total: additions: $5n-9$; multiplications: $n-1$; divisions: $n-1$.

Therefore, for folder size $u_{l-1,j}$, we need the following total number of

operations.

$$\text{Additions: } \sum_{l=2}^{L}\sum_{j=1}^{n_{l-1}}(5u_{l-1,j}-9) = 5\sum_{l=2}^{L} n_l - 9(L-1) \tag{5.22}$$

$$\text{Multiplications: } \sum_{l=2}^{L}\sum_{j=1}^{n_{l-1}}(u_{l-1,j}-1) = \sum_{l=2}^{L} n_l - (L-1) \tag{5.23}$$

$$\text{Divisions: } \sum_{l=2}^{L}\sum_{j=1}^{n_{l-1}}(u_{l-1,j}-1) = \sum_{l=2}^{L} n_l - (L-1) \tag{5.24}$$

So the time complexity of Algorithm 5.2 is on the order of the number of nodes in the hierarchical tree $\sum_{l=1}^{L} n_l$. In the case of a perfect k-ary tree, similar to Equations (5.20) and (5.21), these are of order $O(N)$, where N is the length of signal (bottom level nodes).

CHAPTER 6
TREE CUT AND EXTENSION

6.1 Tree Cut

For processing of signals on graphs, in general, change of graph structure is no easy task. Such change in most scenarios leads to large scale recalculation. For spectral domain methods, it is usually necessary to recalculate eigenvalues and eigenvectors. For vertex domain methods, removing a single vertex can result in huge structure change and rework of the entire wavelet transform. See, for example, [9], [12], [18] and [38].

From Sections 5.3 and 5.4, we can see that cutting off the bottom level of a hierarchical tree will not affect the wavelet transform in a global fashion. Because in the algorithms we only care about the information passed along. Therefore, we would like to define some operations (cut and extension) on hierarchical tree structure and explore their relationship with wavelet transform. However, we cannot just delete the bottom level and keeps the transform matrix, as can been seen in the following example.

Example 6.1. The wavelet transform matrix for Figure 2.1 is

$$M_1 = \begin{pmatrix} 1/9 & 1/9 & 1/9 & 1/9 & 1/9 & 1/9 & 1/9 & 1/9 & 1/9 \\ 1/2 & 1/2 & -1/4 & -1/4 & -1/4 & -1/4 & 0 & 0 & 0 \\ 1/2 & 1/2 & 1/4 & 1/4 & 1/4 & 1/4 & -2/3 & -2/3 & -2/3 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & -3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & -2 \end{pmatrix} \tag{6.1}$$

If we delete the bottom level of the tree, the wavelet transform matrix associated with the new tree becomes

$$M_2 = \begin{pmatrix} 1/3 & 1/3 & 1/3 \\ 1 & -1 & 0 \\ 1 & 1 & -2 \end{pmatrix} \tag{6.2}$$

Note that the rows of M_2 are not scalar multiples of the first three rows of M_1. In M_2, three level 2 folders are treated equally, while in M_1 they have different weights. The reason is that when we cut off the bottom level some information (folder sizes) is lost. Motivated by Algorithm 5.1, we add weights to all folders of the tree, in an attempt to preserve information when the graph structure is changed. This will lead to smooth transition of signals and encoded signals, as shown in Sections 6.3 and 6.4.

Definition 6.1 (Weighted hierarchical tree)**.** For a hierarchical tree defined in Section 2.1, a weight function $w(l,k)$ is assigned to each root node of folder $X_{l,k}$ that satisfies the following conditions.

1. $w(l,k)$ is positive real valued;
2. If folder $X_{l,k}$ splits into subfolders $\{X_{l+1,k_j}\}$ for $j = 1, 2, \ldots, n$, then $\sum_{j=1}^{n} w(l+1, k_j) = w(l,k)$.

A hierarchical tree with such a weight function is called a weighted hierarchical tree.

Some observations.

- Note that unlike general weighted graphs, the weights are assigned to nodes, not edges of the tree.

- Any hierarchical tree as defined in Section 2.1 can be easily turned into a weighted one by letting the weight equal to folder size. Thus, this is a generalized definition of hierarchical tree.
- The representation by folder sizes, as in Definition 2.4, is replaced by weight function. The vector representation in Definition 2.5 can still be used. But they no long uniquely determines a weighted hierarchical tree, as they do not contain information on weights. Thus, we will use both weights and Definition 2.5 to represent a weighted hierarchical tree. In some applications, vector representation plus the bottom level weights is enough, as the other weights can be calculated by the second condition in Definition 6.1.

With the weights assigned, we can define cut on tree.

Definition 6.2 (Tree cut)**.** A cut on a weighted hierarchical tree (Definition 6.1) is the removal of its bottom level nodes and the edges linking to them, together with the weights assign to them. The weights for the other nodes remain unchanged.

It is obvious that both conditions in Definition 6.1 are satisfied after the cut, thus the result of a cut is still a weighted hierarchical tree. Under the representation of Definition 2.5, the cut is simply throwing away the bottom level vector u_{L-1}.

Example 6.2. See Figure 6.1. We use the tree structure Figure 2.1 as an

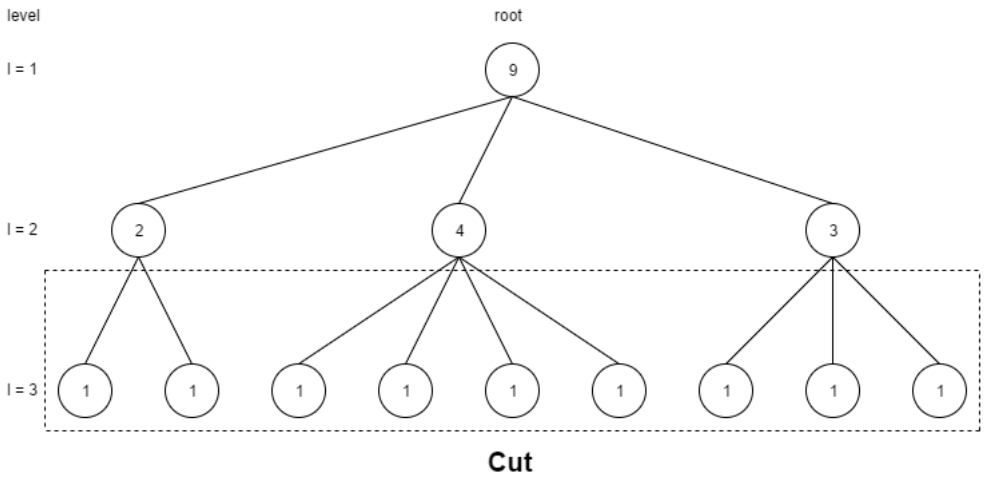

Figure 6.1: Cut on Weighted Hierarchical Tree

example. Assign weights equal to folder sizes, and we have a weighted tree. That is

$$w(3,k) = 1 \text{ for } k = 1, 2, \ldots, 9 \tag{6.3}$$

$$w(2,1) = 2, w(2,2) = 4, w(2,3) = 3 \text{ and } w(1,1) = 9 \tag{6.4}$$

$$u_2 = (2,4,3) \text{ and } u_1 = (3) \tag{6.5}$$

After a cut, we have two levels (four nodes) left. Level 2 becomes the new bottom level. The weights are carried over and the bottom level vector u_2 is discarded. The new representation is

$$w(2,1) = 2, w(2,2) = 4, w(2,3) = 3 \text{ and } w(1,1) = 9 \text{ and } u_1 = (3) \tag{6.6}$$

This process is shown in Figure 6.1 with weights given inside each node.

6.2 Wavelet Transform

In order to build up wavelet transform for weighted hierarchical tree, we need to take weights into consideration. This can be done by changing the father wavelets $\phi_{l,k}$. In (4.5) and (4.12), they are average function on $X_{l,k}$ with sum equal to 1. A natural transition to the weighted tree is to replace the folder size $m_{l,k}$ by its weight $w(l,k)$.

$$\phi_{l,k} = \frac{1}{w(l,k)} \mathbf{1}_{X_{l,k}} \tag{6.7}$$

Theorem 6.1 (Father wavelets as basis for V_l)**.** Let V_l be defined as in Definition 4.1. Then $\{\phi_{l,k} | k = 1, 2, \ldots, n_l\}$ given in (6.7) is a basis for V_l.

Proof. Similar to the proof of Theorem 4.6. $\{\phi_{l,k} | k = 1, 2, \ldots, n_l\}$ are constant functions on folders $X_{l,k}$. So any signal in V_l can be written as a linear combination of $\phi_{l,k}$. And it is obvious that $\phi_{l,k}$ are linearly independent, since they have mutually disjoint supports. □

Mother wavelets are generated in a similar way to (5.3). Suppose folder $X_{l-1,k}$ splits into subfolders $\{Y_{l,k,j}\}, j = 1, 2, \ldots, u_{l-1,k}$ on level l. $Y_{l,k,1}$ is the first subfolder, which is also a level l folder. Let the ϕ_{l,k_0} be the father wavelet associated with this level l folder. The mother wavelets for these subfolders are generated as

$$\psi_{l,k,j} = \sum_{n=1}^{j} \phi_{l,k_0+n-1} - j\phi_{l,k_0+j} \tag{6.8}$$

Theorem 6.2 (Mother wavelets as basis for U_l)**.** Let $\psi_{l,k,j}$ be defined in (6.8). Let U_{l-1} be defined as $\text{span}(\psi_{l,k,j}|j = 1, 2, \ldots, u_{l-1,k}, k = 1, 2, \ldots, n_l)$.

(i) $\psi_{l,k,j}$ are linearly independent.

(ii) $\{\psi_{l,k,j}\}$ form a basis for space U_{l-1}

Proof. Similar to proof of Theorem 5.3. Write mother wavelets generated by (6.8) into matrix form and we have

$$\begin{pmatrix} 1 & -1 & 0 & \ldots & 0 \\ 1 & 1 & -2 & \ldots & 0 \\ & & \vdots & & \\ 1 & 1 & \ldots & 1 & -s \end{pmatrix} \begin{pmatrix} \phi_{l+1,k_0} \\ \phi_{l+1,k_0+1} \\ \vdots \\ \phi_{l+1,k_0+s} \end{pmatrix} \tag{6.9}$$

The vector on the right side consists of father wavelets on the same level, so they are linearly independent. We only need to show the matrix on the left side has full rank. Deleting the first column of this matrix and we get a triangular matrix whose determinant is $\prod_{i=1}^{s}(-i) \neq 0$. This completes the proof. □

Theorem 6.3 (Multiresolution)**.** Let V_l be defined as in Definition 4.1 and U_l as in Theorem 6.2. Then $V_{l+1} = V_l \oplus U_l$.

Proof. Obviously, both V_l and U_l are subset of V_{l+1}. Following the proof of Theorem 4.8, $\dim(V_{l+1}) = n_{l+1}$ and $\dim(V_l) = n_l$. From (6.8) and Theorem 6.2, U_l has the same number of basis vectors as W_l in Definition 4.2. So $\dim(U_l) = \dim(W_l) = n_{l+1} - n_l$. Thus we have

$$\dim(V_{l+1}) = \dim(V_l) + \dim(U_l). \tag{6.10}$$

Now it suffices to show that basis vectors of V_l are not in U_l. If $m \neq k$, $\phi_{l,m}$ and $\psi_{l,k,j}$ have disjoint support. Thus we only need to show $\phi_{l,k}$ and $\{\psi_{l+1,k,j} | j = 1, 2, \ldots, u_{l,k}\}$ are linearly independent. Same as in (6.8), we denote the father wavelet function associated with the first subfolder of $X_{l,k}$ by ϕ_{l+1,k_0} and let $s = u_{l,k} - 1$. Put $\phi_{l,k}$ and $\{\psi_{l+1,k,j} | j = 1, 2, \ldots, u_{l,k}\}$ into the matrix form and we obtain

$$\begin{pmatrix} \frac{w(l+1,k_0)}{w(l,k)} & \frac{w(l+1,k_0+1)}{w(l,k)} & \cdots & \cdots & \frac{w(l+1,k_0+s)}{w(l,k)} \\ 1 & -1 & 0 & \ldots & 0 \\ 1 & 1 & -2 & \ldots & 0 \\ & & \vdots & & \\ 1 & 1 & \ldots & 1 & -s \end{pmatrix} \begin{pmatrix} \phi_{l+1,k_0} \\ \phi_{l+1,k_0+1} \\ \vdots \\ \vdots \\ \phi_{l+1,k_0+s} \end{pmatrix} \tag{6.11}$$

In (6.10), the vector on the right side consists of father wavelets on the same level, so they are linearly independent. Now we need to show the matrix on the left side has full rank.

$$\begin{aligned} &\det \begin{pmatrix} \frac{w(l+1,k_0)}{w(l,k)} & \frac{w(l+1,k_0+1)}{w(l,k)} & \cdots & \cdots & \frac{w(l+1,k_0+s)}{w(l,k)} \\ 1 & -1 & 0 & \ldots & 0 \\ 1 & 1 & -2 & \ldots & 0 \\ & & \vdots & & \\ 1 & 1 & \ldots & 1 & -s \end{pmatrix} \\ &= \sum_{n=0}^{s} \frac{w(l+1,k_0+n)}{w(l,k)} \prod_{i=1}^{s} (-i) \end{aligned} \tag{6.12}$$

The sum $\sum_{n=0}^{s} \frac{w(l+1,k_0+n)}{w(l,k)}$ is positive since weight function is positive by Definition 6.1. The product $\prod_{i=1}^{s} (-i)$ is obviously not zero. So the determinant is non zero. This completes the proof. □

Note that in this case for, detail spaces U_l in general is not orthogonal to

V_l. U_l is the linear complement of V_l in V_{l+1}, but not necessarily the orthogonal complement. We still have $(\oplus_{l=1}^{L-1} U_l) \oplus V_1 = V_L = V$ and the transform is thus invertible. Similar to (5.7) in Chapter 5, we write the transform matrix in block form as

$$M = \begin{pmatrix} \phi_{1,1} \\ \psi_{l,k,j} \end{pmatrix} = \begin{pmatrix} \phi_{1,1} \\ (\psi_{2,k,j}) \\ \vdots \\ (\psi_{L,k,j}) \end{pmatrix} \tag{6.13}$$

$$j = 1, 2, \ldots, u_{l-1,k} - 1; k = 1, 2, \ldots, n_{l-1}; l = 2, 3, \ldots, L$$

The algorithms for wavelet transform on weighted hierarchical trees are similar to the ones introduced in Chapter 5. The tree structure is now specified by vector sequence $\{u_l\}$ and bottom level weights $w(L, k)$. The weights on the other levels can be calculated through the algorithm. The sum operator and detail operator remain unchanged and their sizes are still determined by $\{u_l\}$. For the information passed along in the algorithm, folder size $m_{l,k}$ is replaced by weight function $w(l, k)$. In the reconstruction part, we also change the folder size to weight for input to the reconstruction operator.

Algorithm 6.1 Weighted Tree Wavelet Encoding Algorithm

Task: Apply wavelet transform introduced in Chapter 6 to signal on weighted hierarchical tree of level L (See Definition 6.1).

Input: Signal f of size N; Tree structure specified by a sequence of vectors $u_l = (u_{l,1}, u_{l,2}, \ldots, u_{l,n_l})$ for $l = 1, 2, \ldots, L-1$ (See Definition 2.5) and bottom level weights $w(L,k)$ for $k = 1, 2, \ldots, N$ (See Definition 6.1).

Parameters: $s_{l,k}$ = sum of f over $X_{l,k}$; c is the cumulative counter for grouping; $M_{l-1,j}$ is the matrix for j-th folder at level $l-1$; $\left(d_{l-1,j}\right)$ is details in vector form of size $u_{l-1,j} - 1$; S and D are the sum and detail operators (See Definitions 5.1 and 5.2).

Initialization: Set $s_{L,k} = f_k$ and get $w(L,k)$ from input for $k = 1, 2, \ldots, N$.

Main Iteration for $l = L, L-1, \ldots, 2$.

$c = 0$

Level l Iteration for $j = 1, 2, \ldots, n_{l-1}$

$M_{l-1,j} = \left(s_{l,k}, w(l,k)\right)_{k=c+1,\ldots,c+u_{l-1,j}}$ %Group into next level folders.

$\left(s_{l-1,j}, w(l-1,j)\right) = S_{u_{l-1,j}} M_{l-1,j}$ %Next level sum and weight.

$\left(d_{l-1,j}\right) = D_{u_{l-1,j}} M_{l-1,j}$ %Details at this level.

$c = c + u_{l-1,j}$

End Level l Iteration

End Main Iteration

Output: Sum of entire signal is $s_{1,1}$; Average of entire signal is $s_{1,1}/w(1,1)$; Level l details are $\left(d_{l-1,j}\right)$ for $j = 1, 2, \ldots, n_{l-1}$.

Algorithm 6.2 Weighted Tree Wavelet Decoding Algorithm

Task: Reconstruct wavelet (introduced in Chapter 6) encoded signal to original signal f on weighted hierarchical tree of level L (See Definition 6.1).

Input: Sum of entire signal $s_{1,1}$; Details at each level in vector form $\left(d_{l,j}\right)$ of size $u_{l,j} - 1$ (See Algorithm 6.1); Tree structure specified by a sequence of vectors $u_l = (u_{l,1}, u_{l,2}, \ldots, u_{l,n_l})$ for $l = 1, 2, \ldots, L-1$ (See Definition 2.5) and weights $w(l,k)$ for $l = 1, 2, \ldots, L, k = 1, 2, \ldots, n_l$.

Parameters: $s_{l,k}$ = sum of f over $X_{l,k}$; $a_{l,k}$ = average of f over $X_{l,k}$; c is the cumulative counter for grouping; R is the reconstruction operator (See Definition 5.3).

Main Iteration for $l = 1, 2, \ldots, L-1$.

$c = 0$

Level l Iteration for $j = 1, 2, \ldots, n_l$

$$\left(a_{l+1,k}\right)_{k=c+1,\ldots,c+u_{l,j}} = R_{u_{l,j}}\left(\left(w(l+1,k)\right)_{k=c+1,\ldots,c+u_{l,j}}; s_{l,j}; \left(d_{l,j}\right)\right)$$

%Reconstruct average for subfolders of $X_{l,j}$.

$s_{l+1,k} = w(l+1,k)a_{l+1,k}$ %Convert average into sum.

$c = c + u_{l,j}$

End Level l Iteration

End Main Iteration

Output: Original signal $f_k = s_{L,k}$ for $k = 1, 2, \ldots, N$.

6.3 Tree Cut and Encoded Signal

Our goal is to establish a simple relationship between cut on a tree and the corresponding wavelet encoded signal. From the Algorithms 6.1 and 6.2, it can be seen that cutting off the bottom level of a weighted tree is equivalent to throwing away the bottom level details of the encoded signal. In order to properly reflex the structure change of tree cut, we first define the associated cut on original signal and encoded signal.

Definition 6.3 (Cut on signal)**.** Let $f = (f(x_1), f(x_2), \ldots, f(x_N))$ be a signal on a weighted hierarchical tree (Definition 6.1) of level L. A cut on f associated with the tree cut (Definition 6.2) is

$$\hat{f} = (\sum_{x_i \in X_{L-1,1}} f(x_i), \sum_{x_i \in X_{L-1,2}} f(x_i), \ldots, \sum_{x_i \in X_{L-1,n_{L-1}}} f(x_i)) \tag{6.14}$$

So cutting a signal is to to group its components into level $L-1$ folders and take sum over each folder. The size of the cut signal is the number of level $L-1$ folders n_{L-1}.

Definition 6.4 (Cut on encoded signal)**.** Let g be a wavelet encoded signal (by transform matrix (6.12)) on a weighted hierarchical tree (Definition 6.1) of level L. A cut on g associated with the tree cut (Definition 6.2) $\hat{g}$ is the removal of level L details, or the last $n_L - n_{L-1}$ components.

Cut on encoded signal is simply throwing a way its last $n_L - n_{L-1}$ components, which are level L details. In applications, this step does not involve any actual computation.

Theorem 6.4 (Tree cut related to encoded signal cut)**.** Let a weighted hierarchical tree with weights $w(l,k)$ be represented by $\{m_{l,k}|l=1,2,\ldots,L,k=1,2,\ldots,n_l\}$ and $\{u_l|l=1,2,\ldots,L-1\}$ (Definitions 2.4 and 2.5). Let $\{m'_{l,k}|l=1,2,\ldots,L-1,k=1,2,\ldots,n_l\}$ and $\{u_l|l=1,2,\ldots,L-2\}$ be the representation of a tree cut (Definition 6.2). Let M and $\hat{M}$ be the wavelet transform matrices (Equation (6.12)) for the original tree and its cut. Let f and g be a signal on the original tree and its wavelet transform. Let $\hat{f}$ and $\hat{g}$ be the cut signal and cut encoded signal (Definitions 6.3 and 6.4). Then

$$\hat{g} = \widehat{Mf} = \hat{M}\hat{f} \tag{6.15}$$

Proof. For simplicity, we follow the notations in Algorithm 6.1 for the proof. By Definition 6.2, the cut does not change the weights on levels $L-1$ and up. By definition 6.4, the j-th component of $\hat{f}$ together its weight is

$$\left(\sum_{x_k\in X_{L-1,j}} f(x_k), \sum_{x_k\in X_{L-1,j}} w(L,k)\right) = M_{L-1,j} \tag{6.16}$$

Now compare M and $\hat{M}$. The last $n_L - n_{L-1}$ rows of M are removed. The first n_{L-1} rows of $\hat{M}$ are similar to M but are squeezed due to the change in folder sizes. However, by definition cut on the tree does not change vector representation $\{u_l\}$ or weights $w(l,k)$ for levels $L-1$ and up. And Algorithm 6.1 is independent on folder size during the Main Iteration step (See Definitions 5.1 and 5.2 for sum and detail operator). It only depends on vector representation $\{u_l\}$ and weights $w(l,k)$. Therefore, Main Iteration produces the same result as the original signal for iteration steps $L-1$ and up. This is exactly what we

have for $\hat{g} = \widehat{Mf}$. □

Example 6.3. We give an example of Theorem 6.4, based on the weighted hierarchical tree shown in Figure 6.2, with weights shown in nodes. The wavelet transform matrix for the original 3-level tree is

$$M = \begin{pmatrix} 1/15 & 1/15 & 1/15 & 1/15 & 1/15 & 1/15 & 1/15 & 1/15 & 1/15 \\ 1/4 & 1/4 & -1/6 & -1/6 & -1/6 & -1/6 & 0 & 0 & 0 \\ 1/4 & 1/4 & 1/6 & 1/6 & 1/6 & 1/6 & -2/5 & -2/5 & -2/5 \\ 1 & -1/3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & -1/2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 1/2 & -2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/2 & 1/2 & 1 & -3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1/3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1/3 & -2 \end{pmatrix} \tag{6.17}$$

After the cut, the wavelet transform matrix is

$$\hat{M} = \begin{pmatrix} 1/15 & 1/15 & 1/15 \\ 1/4 & -1/6 & 0 \\ 1/4 & 1/6 & -2/5 \end{pmatrix} \tag{6.18}$$

It can be seen that $\hat{M}$ is the first 3 rows of M with squeezed folder size. Now let a signal on the original tree be

$$f = (3, 6, 4, 2, 3, 2, 5, 3, 2)^T \tag{6.19}$$

The encoded signal is

$$g = Mf = (30/15, 9/4 - 11/6, 9/4 + 11/6 - 20/5, 1, 1, -3, 0, 4, 2)^T \tag{6.20}$$

The cut signal is

$$\hat{f} = (9, 11, 10)^T \tag{6.21}$$

$$\hat{M}\hat{f} = (30/15, 9/4 - 11/6, 9/4 + 11/6 - 20/5) \tag{6.22}$$

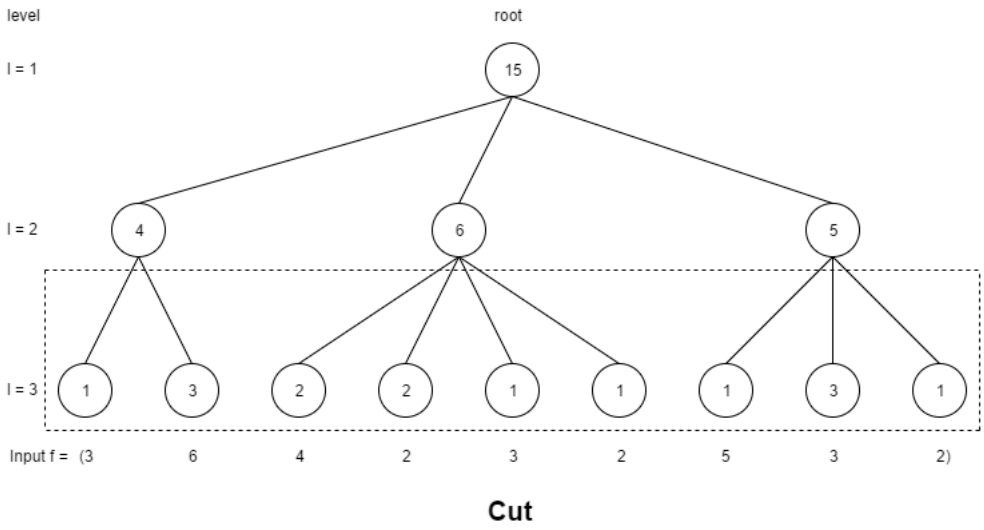

Figure 6.2: Cut on Weighted Hierarchical Tree and an Input Signal

This is exactly the first three components of the encoded signal g, which is the cut encoded signal $\hat{g} = \widehat{Mf}$.

In Theorem 6.4 and Example 6.3, we have shown a simple relationship between cut on tree and cut on encoded signal. This has been made possible due to the weights we assigned to each node, which retains essential information on tree structure when it is cut. In applications, there is no actual computation needed. For example, when we ignore small details on bottom level, Theorem 6.4 guarantees that this is equivalent to ignoring the bottom level of the tree, with appropriate change on the input signal. The change of input signal is already one step in the encoding algorithm. There is no need to recalculate wavelet transform.

6.4 Tree Extension

The opposite operation of tree cut is tree extension, which is adding a new level below the bottom level of an existing tree. We give the definitions below. As expected, a simple relationship between tree extension and encoded signal extension can be established.

Definition 6.5 (Tree extension)**.** An extension of a weighted hierarchical tree (Definition 6.1) of level L is the addition of a new level of nodes below the bottom level, together with the weights assigned to them, and the edges linking to them such that the following conditions are satisfied.

1. $w(L+1,k)$ is positive real valued for $k = 1, 2, \ldots, n_{L+1}$;
2. $\sum_{y_k \in X_{L,j}} w(L+1,k) = w(L,j)$;

where the new function space is on the new leaf nodes $y_1, y_2, \ldots, y_{n_{L+1}}$.

When we add a new level of nodes, we need to ensure that the extended tree is also a weighted hierarchical tree. That is, the two conditions in Definition 6.1 are satisfied. Essentially, the weights on bottom level nodes are distributed among the nodes of the new level added below.

Definition 6.6 (Signal extension)**.** Let $f = (f(x_1), f(x_2), \ldots, f(x_N))$ be a signal on a weighted hierarchical tree (Definition 6.1) of level L. An extension of f associated with the tree extension (Definition 6.5) is

$$\tilde{f} = (\tilde{f}(y_1), \tilde{f}(y_2), \ldots, \tilde{f}(y_{N_{L+1}})) \tag{6.23}$$

that satisfies the following condition

$$f(x_j) = \sum_{y_k \in X_{L,j}} \tilde{f}(y_k) \text{ for } j = 1, 2, \ldots, n_L \tag{6.24}$$

Definition 6.7 (Encoded signal extension)**.** Let g be a wavelet encoded signal (by transform matrix (6.12)) of original signal f on a weighted hierarchical tree (Definition 6.1) of level L. An extension of g associated with the tree extension (Definition 6.5) $\tilde{g}$ is the addition of level $L+1$ details to g as the last $n_{L+1} - n_L$ components. The added details are generated by $\tilde{f}$ (Definition 6.6).

$$\tilde{g} = (g, (d_{L,1}), (d_{L,2}), \ldots, (d_{L,n_L})) \tag{6.25}$$

where $(d_{L,j}) = D_{u_{L,j}} M_{L,j}$ are defined as in Algorithm 6.1.

Definitions 6.6 and 6.7 are parallel to Definitions 6.3 and 6.4. Extension followed by cut dose not a signal or encoded signal.

Theorem 6.5 (Cut of extension)**.** Suppose we perform an extension followed by a cut on a weighted hierarchical tree. Let f and g be a signal on the original tree and its wavelet transform. Then

$$\hat{\tilde{f}} = f \text{ and } \hat{\tilde{g}} = g \tag{6.26}$$

Proof. This is obvious from Definitions 6.3, 6.4, 6.6 and 6.7. □

Note that if we perform cut first and then extension, we do not get $\tilde{\hat{f}} = f$ and $\tilde{\hat{g}} = g$. Since the extension adds arbitrary structure on the bottom level that is not necessarily the same as the original tree.

As one would expect, tree extension has two equivalent realizations. The first is to add of details to the encoded signal. The second is to extend the input signal and the transform matrix, and then to recalculate encoded signal. This is summarized in the following theorem.

Theorem 6.6 (Tree extension related to encoded signal extension)**.** Let a weighted hierarchical tree with weights $w(l,k)$ be represented by $\{m_{l,k}|l = 1,2,\ldots,L, k = 1,2,\ldots,n_l\}$ and $\{u_l|l = 1,2,\ldots,L-1\}$ (Definitions 2.4 and 2.5). Let $\{m'_{l,k}|l = 1,2,\ldots,L+1, k = 1,2,\ldots,n_l\}$ and $\{u_l|l = 1,2,\ldots,L\}$ be the representation of a tree extension (Definition 6.5). Let M and $\tilde{M}$ be the wavelet transform matrices (Equation (6.12)) for the original tree and its extension. Let f and g be a signal on the original tree and its wavelet transform. Let $\tilde{f}$ and $\tilde{g}$ be the extended signal and extended encoded signal (Definitions 6.6 and 6.7). Then

$$\tilde{g} = \widetilde{Mf} = \tilde{M}\tilde{f} \tag{6.27}$$

Proof. We follow the notations in Algorithm 6.1 for the proof. $\tilde{M}$ consists of two blocks M_1 and M_2.

$$\tilde{M} = \begin{pmatrix} M_1 \\ M_2 \end{pmatrix} \tag{6.28}$$

The top n_L rows M_1 are extended version of M and the bottom $n_{L+1} - n_L$ rows M_2 are the one level detail blocks for the extended level $L+1$ details. By Definition 6.7, $M_2\tilde{f} = ((d_{L,1}),(d_{L,2}),\ldots,(d_{L,n_L}))$. Thus it suffices to show $M_1\tilde{f} = g$. Given input $\tilde{f}$ in Algorithm 6.1, the Main Iterations only depend on

weights $w(l,k)$ and vector representation u_l. These remain unchanged between original tree and its extension. Therefore, Algorithm 6.1 produces the same result for $M_1\tilde{f}$ and Mf. That is, $M_1\tilde{f} = Mf = g$. □

In applications, sometimes we need to add more data to an existing graph structure. According to Theorem 6.6, this can be done efficiently if we have the structure of a weighted hierarchical tree. All we need to do is the following.

1. Assign new weights that best represent data added and ensure that the new weights conform to the weighted tree structure. This is the extension of tree and input signal (Definitions 6.5 and 6.6).
2. Calculate new bottom level details and add them to the encoded signal (Definition 6.7).

There is no need to recalculate wavelet transform or change its algorithm. All the other information is carried over smoothly.

6.5 Classification

Similar to Section 4.5, we do a brief classification for multiresolution wavelets on weighted hierarchical trees. They are not necessarily orthogonal multiresolution. In Theorem 6.3, we have offered a constructive approach to multiresolution wavelets on weighted trees. It is part of our separate classification result below. By comparison to the case of classic wavelets in $L^2(\mathbb{R}^d)$, it is interesting to note that there is not an analogous known classification for

the classic wavelet families, not even for the case $d = 1$. See See [7], [13] and [28]. Both Theorem 6.3 and our parallel classification result are new.

For simplicity, we require that the nested subspaces V_l be defined as in Definition 4.1. The detail spaces U_l are not necessarily orthogonal to V_l, instead they are just linear complements to V_l in V_{l+1}. That is $V_{l+1} = V_l \oplus U_l$.

Let the father wavelets be $\phi_{l,k} = \dfrac{1}{w(l,k)}\mathbf{1}_{X_{l,k}}$, defined as in (6.7). By Theorem 6.1, they form a basis for V_l. Now we look for potential mother wavelets $\{q_{l,n}\}$ that forms a basis for U_{l-1}. Since $U_{l-1} \subset V_l$, each $q_{l,n}$ must be a linear combinations of $\{\phi_{l,k}\}$. By counting dimension of space, $\dim(U_{l-1}) = \dim(V_l) - \dim(V_{l-1}) = n_l - n_{l-1}$. Therefore, they can be written as

$$q_{l,n} = \sum_{k=1}^{n_l} a_{l,n,k}\phi_{l,k} \tag{6.29}$$

for $n = 1, 2, \ldots, n_l - n_{l-1}$. Here the coefficients $a_{l,n,k}$, depending on l, n and k, are some constants to be determined. Put in the matrix form, we have

$$\begin{pmatrix} q_{l,1} \\ \vdots \\ q_{l,n_l-n_{l-1}} \end{pmatrix} = \begin{pmatrix} a_{l,1,1} & \cdots & a_{l,1,n_l} \\ \vdots & \vdots & \vdots \\ a_{l,n_l-n_{l-1},1} & \cdots & a_{l,n_l-n_{l-1},n_l} \end{pmatrix} \begin{pmatrix} \phi_{l,1} \\ \vdots \\ \phi_{l,n_l} \end{pmatrix} \tag{6.30}$$

Write the matrix as A_l and we require it has full rank, so that $\{q_{l,n}\}$ are linearly independent.

Now we need $V_l = V_{l-1} \oplus U_{l-1}$. Since V_{l-1} is spanned by its basis $\{\phi_{l-1,j}\}$, we need each $\phi_{l-1,j}$ is not a linear combination of $\{q_{l,n}\}$. Write $\phi_{l-1,j}$

as a linear combination of $\{\phi_{l,k}\}$ and we have

$$\phi_{l-1,j} = \sum_{X_{l,k} \subset X_{l-1,j}} \frac{w(l,k)}{w(l-1,j)} \phi_{l,k} \tag{6.31}$$

Write this in matrix form and we obtain

$$\phi_{l-1,j} = W(l-1,j) \begin{pmatrix} \phi_{l,1} \\ \vdots \\ \phi_{l,n_l} \end{pmatrix} \tag{6.32}$$

where $W(l-1,j)$ is a row matrix that is supported on $X_{l-1,j}$.

Put (6.29) and (6.31) together, we have

$$\begin{pmatrix} \phi_{l-1,1} \\ \vdots \\ \phi_{l-1,n_{l-1}} \\ q_{l,1} \\ \vdots \\ q_{l,n_l-n_{l-1}} \end{pmatrix} = \begin{pmatrix} & W(l-1,1) & \\ & \vdots & \\ & W(l-1,n_{l-1}) & \\ a_{l,1,1} & \cdots & a_{l,1,n_l} \\ \vdots & \vdots & \vdots \\ a_{l,n_l-n_{l-1},1} & \cdots & a_{l,n_l-n_{l-1},n_l} \end{pmatrix} \begin{pmatrix} \phi_{l,1} \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ \phi_{l,n_l} \end{pmatrix} \tag{6.33}$$

Therefore, the requirement for potential mother wavelets $\{q_{l,n}\}$ to generate a multiresolution is that the above matrix has full rank. Or equivalently, each $W(l-1,j)$ as a row vector is linearly independent to A_l.

CHAPTER 7
THRESHOLDING

Thresholding is a common practice is classic discrete signal processing. In classic wavelet transforms, low level details are usually quite small, thus throwing away some details under certain value will not cause significant loss of signal quality. At the same time, we can achieve higher rate of compression. We can naturally extend this to the setting of hierarchical trees. If we assume that data collected in the bottom (close to bottom) level folders are closely related, we would expect insignificant loss of signal quality.

The following is a theoretical upper bound for the errors induced by thresholding. In this case, we use vector norm defined in (2.4).

7.1 An Upper Bound

Theorem 7.1. Let M be the transform matrix defined in Chapter 3. Let f be a signal and g be its transform. $\tilde{g}$ is g after thresholding and $\tilde{f} = M^{-1}\tilde{g}$. An upper bound of error is given by

$$\|f - \tilde{f}\| \leq \sqrt{N}\|g - \tilde{g}\| \tag{7.1}$$

Proof. Let $\|\cdot\|_{op}$ denote the operator norm of a matrix.

Let $MM^T = D = diag(d_1, \ldots, d_N)$.

$$\|f - \tilde{f}\| = \|M^{-1}(g - \tilde{g})\| \leq \|M^{-1}\|_{op}\|g - \tilde{g}\| \tag{7.2}$$

$$\|M^{-1}\|_{op} = \text{ maximal singular value of } M^{-1} \tag{7.3}$$

$$\|M^{-1}\|_{op}^2 = \text{ maximal eigenvalue of } (M^{-1})^T M^{-1} \tag{7.4}$$

By Theorem 3.5., $(M^{-1})^T M^{-1} = (M^T D^{-1})^T M^{-1} = D^{-1} M M^{-1} = D^{-1}$

$$\|M^{-1}\|_{op} = (\text{minimal of } \sqrt{d_i})^{-1} = \sqrt{d_1}^{-1} = \sqrt{N} \tag{7.5}$$

□

7.2 Some Examples

We choose a specific signal and apply various thresholds on both wavelets defined in Chapter 3 and Chapter 5. See Appendix A.3 for the MATLAB code.

In the following example, we build two sets of wavelet basis from Chapter 3 and Chapter 5, both based on the tree structure given in Figure 2.1. Our choice of signal is $f = (12, 14, 1.5, 3.5, 4, 2, 8, 6, 7.5)$. We choose this on the assumption that data collected in the same low level folder are more closely related. In this case, their value is close. Thresholds of 0.5, 1 and 2 are tested.

The results show that the wavelet basis in Chapter 5 reproduces better signals after thresholding. The reason is that our choice of input signal has similar value inside each bottom level folder. Therefore, by treating each folder equally with Chapter 5 wavelet basis, we lose less information through thresholding. In practice, data similarity inside low level folder is a property we can often assume.

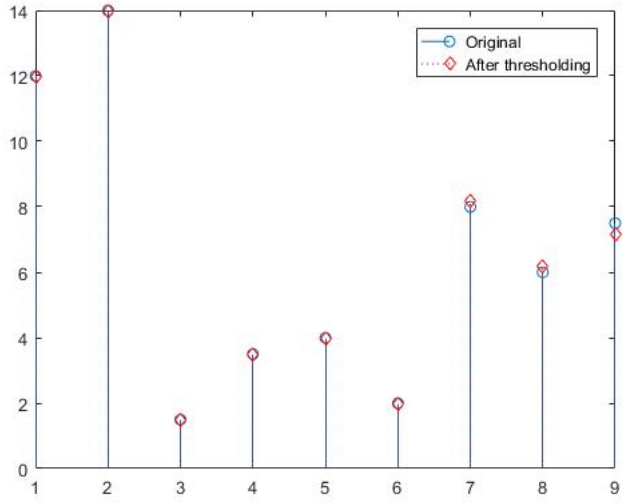

Figure 7.1: Chapter 3 wavelets, threshold = 0.5

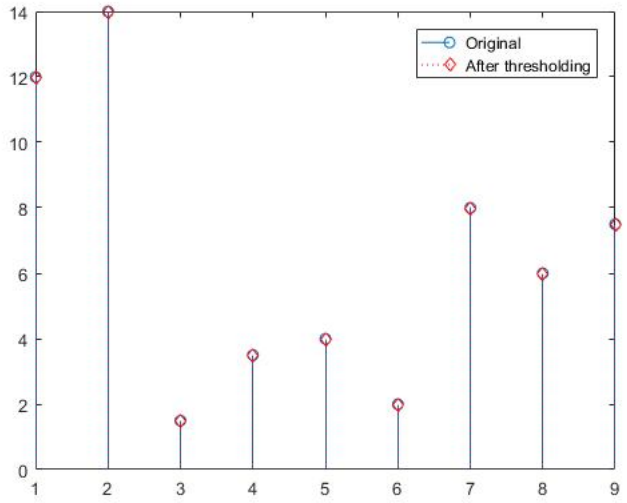

Figure 7.2: Chapter 5 wavelets, threshold = 0.5

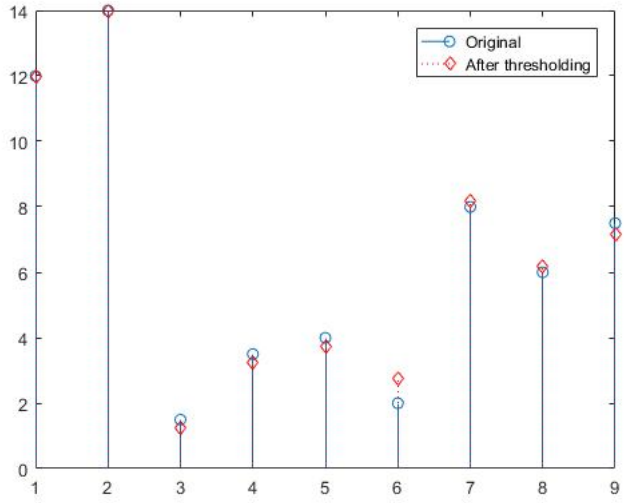

Figure 7.3: Chapter 3 wavelets, threshold = 1

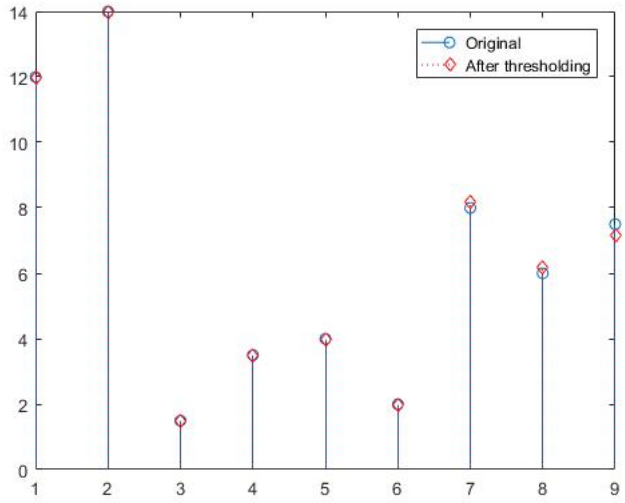

Figure 7.4: Chapter 5 wavelets, threshold = 1

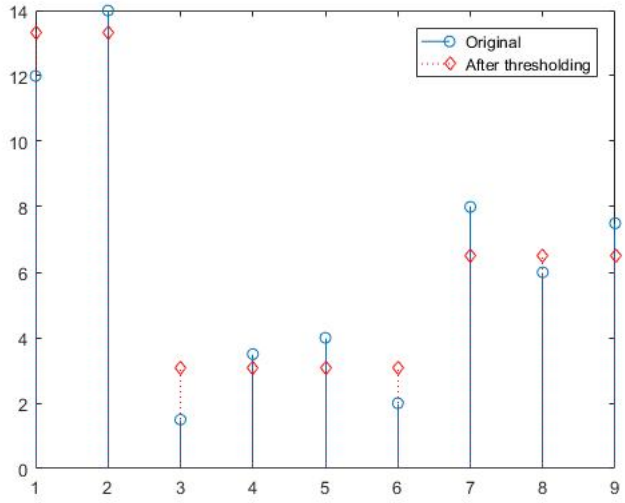

Figure 7.5: Chapter 3 wavelets, threshold = 2

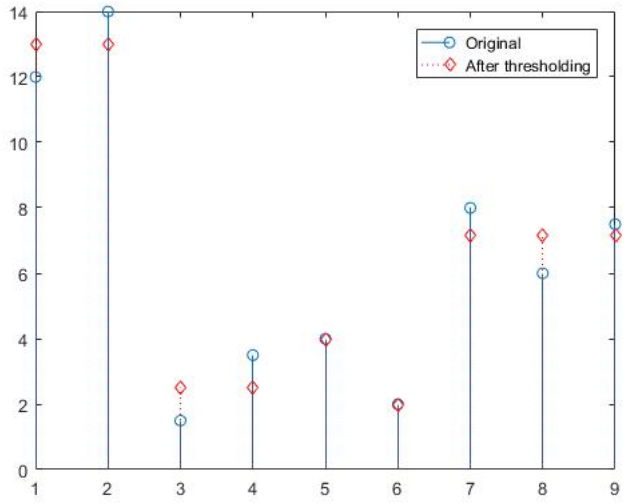

Figure 7.6: Chapter 5 wavelets, threshold = 2

CHAPTER 8
SUMMARY AND FUTURE WORK

8.1 Summary of This Work

We have accomplished three main goals in this work.

The first is construction of multiresolution analysis on hierarchical tree. Following the a wavelet basis proposed in [17], we give the mathematics background and details in the construction of this wavelet. Motivated by the classic wavelet theory, we make modifications and define common operators, such as scaling and translation, in the new setting. And then we define a multiresolution analysis on hierarchical tree and prove the above wavelet basis yields am orthogonal multiresolution. All these have been achieved in close analogue to the classic case, so that many good properties can be carried over. Finally, we make a classification for all possible choices that lead to an orthogonal multiresolution.

For the purpose of realization, we need efficient algorithm to carry out wavelet transform. The above wavelet basis that involves recalculating relative weights at each level seems inadequate in this regard. Therefore, we build up another wavelet basis and prove that it also yields an orthogonal multiresolution. This one can be realized by efficient algorithms, for both encoding and decoding. We briefly estimated the time complexity of both algorithms, which turns out to be of order $O(N)$.

The third is establish a link between encoded signal and change of tree structure. We extend to the definition of weighted hierarchical tree and build a wavelet transform on it. This transform, as we have shown, also yields a multiresolution, although no longer orthogonal. Tree cut is then defined, and a simple relation between encoded signal and tree cut established. This means we need almost no real world computation when a tree is cut. Similarly, tree extension is defined and its link to encoded signal given. In this case, we only need to calculate the details generated by the extension without any change to other data. We also make a classification for all possible choices that lead to a multiresolution on weighted hierarchical trees.

Tree is a special category of graphs, since it can be seen as a building block for a general graph. We have studied signals and their wavelet transform on hierarchical trees, a special kind of tree. Therefore, we have made a first step effort to link the classic wavelet theory to signals on general graph.

8.2 Future Research

- We introduced a new wavelet basis in Chapter 5. This is based on the assumption that data collected in the same low level folder will share some similarity, such as close in value. From Section 4.5, the requirement for an orthogonal multiresolution is that the matrix

$$\begin{pmatrix} 1 & -1 & 0 & 0 & \dots & 0 \\ 1 & 1 & -2 & 0 & \dots & 0 \\ & & \vdots & & & \\ 1 & 1 & \dots & 1 & 1 & -s \end{pmatrix}$$

has full rank and zero sum on each row. We can build more wavelet basis to suit data coming from various source. For example, if we expect data to have a linear relation within low level folders, we can try a matrix like the following since it preserves the first momentum.

$$\begin{pmatrix} 1 & -1 & 0 & 0 & \dots & 0 \\ 1 & 2 & -3 & 0 & \dots & 0 \\ & & \vdots & & & \\ 1 & 2 & 3 & \dots & s-1 & -\dfrac{s(s-1)}{2} \end{pmatrix}$$

Another interesting choice is

$$\begin{pmatrix} 1 & -1 & 0 & 0 & \dots & 0 \\ 0 & 1 & -1 & 0 & \dots & 0 \\ & & \vdots & & & \\ 0 & 0 & 0 & \dots & 1 & -1 \end{pmatrix}$$

- In Section 5.5, we only examine the time complexity of Algorithms 5.1 and 5.2 for the "best case" scenario, a perfect k-ary tree. This gives $O(N)$ result, where N is signal length. In more general setting, a hierarchical tree might be unbalanced with number of child nodes fluctuating from node to node. An estimate of complexity for a general hierarchical tree is to be given.
- In Section 4.5, we have made requirements for the nested subspaces and father wavelets before classification. If these requirements are removed, we might have a broader choice of wavelet basis.
- We introduced tree cut and extension in Chapter 6. These are done for a complete level to keep all leaf nodes on the bottom level. If we no long

require leaf nodes to appear only on the bottom level, we will have more freedom in tree structure change, such as cut and extension on a few nodes instead of a level. This might be desirable in some applications. We need to make proper change for the wavelet construction.

- For every graph, there is a subgraph that is a tree and contains all vertices of the graph. It is called a spanning tree. Every finite graph has a spanning tree. Thus it is a natural idea to construct wavelets on a spanning tree and then extend that to the whole graph. We have introduced wavelets on a special kind of tree, hierarchical tree, in this work. So the next step is to find a link between hierarchical tree and general spanning tree.
- Infinite tree or graph can be seen an extension of the finite case by letting the size go to infinity. Proper changes need to be made and not all properties can be carried over. Examples include Bratteli diagrams and hypercube of rank n. See [6].

APPENDIX A
MATLAB CODES

A.1 Wavelet Transform Matrix for Chapter 3

```
function M = MW1(n)

% This function calculates the bottom level mother wavelets

% defined in Chapter 3.


M = zeros(n,n+1);

for i = 1:n

    for j = 1:i

        M(i,j) = 1/i;

    end

    M(i,i+1) = -1;

end


function [A,B] = WoT1(v)

% This function calculates the wavelet transform and inverse

% transform matrices defined in Chapter 3 of signals on a

% hierarchical tree of level 3.

% The input v is a vector of positive integers that gives the

% second level folder structure.

% The output A is the transform matrix and B is the inverse
```

```
% transform matrix.


n = sum(v); m = length(v);

% size of signal and number of bottom level folders


A = zeros(n);


A(1,:) = ones(1,n)./n;

% level 1 wavelet


for i = 1:m-1

    for j = 1:sum(v(1:i))

        A(i+1,j) = 1/sum(v(1:i));

    end

    for j = sum(v(1:i))+1:sum(v(1:i+1))

        A(i+1,j) = -1/v(i+1);

    end

end

% level 2 wavelets


u = v;

for i = 1:m
```

```
    u(i) = sum(v(1:i))-i;
end
u = [0 u];
for i = 1:m
    A(m+u(i)+1:m+u(i+1),sum(v(1:i-1))+1:sum(v(1:i))) =
    MW1(v(i)-1);
end
% level 3 wavelets


D = A*A';
% diagonal matrix


B = A'/D;
% inverse transform matrix
```

A.2 Wavelet Transform Matrix for Chapter 5

```
function M = MW2(n)
% This function calculates the bottom level mother wavelets
% defined in Chapter 5.
M = zeros(n,n+1);
for i = 1:n
    for j = 1:i
```

```
        M(i,j) = 1;
    end
    M(i,i+1) = -i;
end

function [A,B] = WoT2(v)
% This function calculates the wavelet transform and inverse
% transform matrices defined in Chapter 5 of signals on a
% hierarchical tree of level 3.
% The input v is a vector of positive integers that gives the
% second level folder structure.
% The output A is the transform matrix and B is the inverse
% transform matrix.

n = sum(v); m = length(v);
% size of signal and number of bottom level folders

A = zeros(n);

A(1,:) = ones(1,n)./n;
% level 1 wavelet

for i = 1:m-1
```

```
    for j = 1:i
        for n = 1:v(j)
        A(i+1,n+sum(v(1:j-1))) = 1/v(j);
        end
    end
    for j = sum(v(1:i))+1:sum(v(1:i+1))
        A(i+1,j) = -1/v(i+1);
    end
end
% level 2 wavelets


u = v;
for i = 1:m
    u(i) = sum(v(1:i))-i;
end
u = [0 u];
for i = 1:m
    A(m+u(i)+1:m+u(i+1),sum(v(1:i-1))+1:sum(v(1:i))) =
    MW2(v(i)-1);
end
% level 3 wavelets
```

```
B = inv(A);
% inverse transform matrix
```

A.3 Thresholding of Signal

```
function Threshold(threshold,wavelet)
% This function gives some examples of wavelet transform on
% trees with thresholding.
% t is the threshold put on the transformed signal.
% wavelet = 1 for Chapter 3 wavelets and wavelet = 2 for Chapter
% 5 wavelets.


v = [2 4 3];
% tree structure
if wavelet == 1
    [A,B] = WoT1(v);
end
if wavelet == 2
    [A,B] = WoT2(v);
end
% transform and inverse transform matrices
f = [12 14 1.5 3.5 4 2 8 6 7.5]';
% input signal
```

```
g = A*f;

% transformed signal


t = threshold;

% threshold

g1 = g;

for i = 1:length(g)

    if abs(g1(i)) <= t

        g1(i) = 0;

    end

end

% transformed signal after thresholding

f1 = B*g1;

% reconstructed signal after thresholding

stem (f)

hold on

stem (f1,':diamondr')

hold off

legend('Original','After thresholding')


clear;
```

REFERENCES

[1] Paul S Addison. *The illustrated wavelet transform handbook: introductory theory and applications in science, engineering, medicine and finance*. CRC press, 2002.

[2] Daniel Alpay, Palle Jorgensen, and Izchak Lewkowicz. Extending wavelet filters: infinite dimensions, the nonrational case, and indefinite inner product spaces. In *Excursions in harmonic analysis. Volume 2*, Appl. Numer. Harmon. Anal., pages 69–111. Birkhäuser/Springer, New York, 2013.

[3] Itai Benjamini, Russell Lyons, Yuval Peres, and Oded Schramm. Uniform spanning forests. *Ann. Probab.*, 29(1):1–65, 2001.

[4] Christophe P. Bernard, Stéphane G. Mallat, and Jean-Jacques Slotine. Scattered data interpolation with wavelet trees. In *Curve and surface fitting (Saint-Malo, 2002)*, Mod. Methods Math., pages 59–64. Nashboro Press, Brentwood, TN, 2003.

[5] Ahmad Biniaz, Prosenjit Bose, Anil Maheshwari, and Michiel Smid. Plane geodesic spanning trees, Hamiltonian cycles, and perfect matchings in a simple polygon. In *Topics in theoretical computer science*, volume 9541 of *Lecture Notes in Comput. Sci.*, pages 56–71. Springer, [Cham], 2016.

[6] Ola Bratteli. Inductive limits of finite dimensional c*-algebras. *Transactions of the American Mathematical Society*, 171:195–234, 1972.

[7] Ola Bratteli and Palle Jorgensen. *Wavelets through a looking glass: the world of the spectrum*. Springer Science & Business Media, 2013.

[8] Siheng Chen, Aliaksei Sandryhaila, and Jelena Kovačević. Sampling theory for graph signals. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3392–3396. IEEE, 2015.

[9] Siheng Chen, Rohan Varma, Aliaksei Sandryhaila, and Jelena Kovačević. Discrete signal processing on graphs: Sampling theory. *IEEE Transactions on Signal Processing*, 63(24):6510–6523, 2015.

[10] Xiuyuan Cheng, Xu Chen, and Stéphane Mallat. Deep Haar scattering networks. *Inf. Inference*, 5(2):105–133, 2016.

[11] Maureen Clerc and Stéphane Mallat. Estimating deformations of stationary processes. *Ann. Statist.*, 31(6):1772–1821, 2003.

[12] Mark Crovella and Eric Kolaczyk. Graph wavelets for spatial traffic analysis. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 3, pages 1848–1857. IEEE, 2003.

[13] Ingrid Daubechies. *Ten lectures on wavelets*, volume 61 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1992.

[14] Ingrid Daubechies, Olof Runborg, and Jing Zou. A sparse spectral method for homogenization multiscale problems. *Multiscale Model. Simul.*, 6(3):711–740, 2007.

[15] Dorin E. Dutkay and Palle E. T. Jorgensen. Wavelets on fractals. *Rev. Mat. Iberoam.*, 22(1):131–180, 2006.

[16] Dorin Ervin Dutkay, John Haussermann, and Palle E. T. Jorgensen. Atomic representations of Cuntz algebras. *J. Math. Anal. Appl.*, 421(1):215–243, 2015.

[17] Matan Gavish, Boaz Nadler, and Ronald R Coifman. Multiscale wavelets on trees, graphs and high dimensional data: Theory and applications to semi supervised learning. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 367–374, 2010.

[18] David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011.

[19] Ami Harti. Discrete multi-resolution analysis and generalized wavelets. *Applied numerical mathematics*, 12(1):153–192, 1993.

[20] Norden E. Huang, Ingrid Daubechies, and Thomas Y. Hou. Adaptive data analysis: theory and applications. *Philos. Trans. A*, 374(2065):20150207, 3, 2016.

[21] Arne Jensen and Anders la Cour-Harbo. *Ripples in mathematics: the discrete wavelet transform*. Springer Science & Business Media, 2001.

[22] Palle ET Jorgensen. *Analysis and probability: wavelets, signals, fractals*, volume 234. Springer Science & Business Media, 2006.

[23] Palle ET Jorgensen and Myung-Sin Song. Comparison of discrete and continuous wavelet transforms. In *Encyclopedia of Complexity and Systems Science*, pages 1163–1177. Springer, 2009.

[24] Nathan Kahl. On constructing rational spanning tree edge densities. *Discrete Appl. Math.*, 213:224–232, 2016.

[25] Mikhail Klin and Štefan Gyürki. *Selected topics from algebraic graph theory*. Belianum, Vydavatel'stvo Univerzity Mateja Bela, Banská Bystrica, 2015. Lecture notes.

[26] Ann B Lee, Boaz Nadler, and Larry Wasserman. Treelets: an adaptive multi-scale basis for sparse unordered data. *The Annals of Applied Statistics*, pages 435–471, 2008.

[27] Alessandro Maddaloni and Carol T. Zamfirescu. A cut locus for finite graphs and the farthest point mapping. *Discrete Math.*, 339(1):354–364, 2016.

[28] Stephane Mallat. *A wavelet tour of signal processing: the sparse way*. Academic press, 2008.

[29] Fionn Murtagh. The haar wavelet transform of a dendrogram. *Journal of Classification*, 24(1):3–32, 2007.

[30] Sunil K Narang and Antonio Ortega. Lifting based wavelet transforms on graphs. In *Proceedings: APSIPA ASC 2009: Asia-Pacific Signal and Information Processing Association, 2009 Annual Summit and Conference*, pages 441–444. Asia-Pacific Signal and Information Processing Association, 2009 Annual Summit and Conference, International Organizing Committee, 2009.

[31] Sunil K Narang and Antonio Ortega. Perfect reconstruction two-channel wavelet filter banks for graph structured data. *IEEE Transactions on Signal Processing*, 60(6):2786–2799, 2012.

[32] Ha Q Nguyen and Minh N Do. Downsampling of signals on graphs via maximum spanning trees. *IEEE Transactions on Signal Processing*, 63(1):182–191, 2015.

[33] Seongmin Ok, R. Bruce Richter, and Carsten Thomassen. Liftings in Finite Graphs and Linkages in Infinite Graphs with Prescribed Edge-Connectivity. *Graphs Combin.*, 32(6):2575–2589, 2016.

[34] Alan V Oppenheim and Ronald W Schafer. *Discrete-time signal processing*. Pearson Higher Education, 2010.

[35] Alan V Oppenheim, Alan S Willsky, and Syed Hamid Nawab. *Signals and systems*, volume 2. Prentice-Hall Englewood Cliffs, NJ, 1983.

[36] Idan Ram, Michael Elad, and Israel Cohen. Generalized tree-based wavelet transform. *IEEE Transactions on Signal Processing*, 59(9):4199–4209, 2011.

[37] Idan Ram, Michael Elad, and Israel Cohen. Redundant wavelets on graphs and high dimensional data clouds. *IEEE Signal Processing Letters*, 19(5):291–294, 2012.

[38] Aliaksei Sandryhaila and José MF Moura. Discrete signal processing on graphs. *IEEE transactions on signal processing*, 61(7):1644–1656, 2013.

[39] Aliaksei Sandryhaila and Jose MF Moura. Big data analysis with signal processing on graphs: Representation and processing of massive data sets with irregular structure. *IEEE Signal Processing Magazine*, 31(5):80–90, 2014.

[40] Aliaksei Sandryhaila and Jose MF Moura. Discrete signal processing on graphs: Frequency analysis. *IEEE Transactions on Signal Processing*, 62(12):3042–3054, 2014.

[41] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE Signal Processing Magazine*, 30(3):83–98, 2013.

[42] Javier Soria and Pedro Tradacete. Best constants for the Hardy-Littlewood maximal operator on finite graphs. *J. Math. Anal. Appl.*, 436(2):661–682, 2016.

[43] Krishnaiyan Thulasiraman, Subramanian Arumugam, Andreas Brandstädt, and Takao Nishizeki, editors. *Handbook of graph theory, combinatorial optimization, and algorithms*. Chapman & Hall/CRC Computer and Information Science Series. CRC Press, Boca Raton, FL, 2016.

[44] Nicolas Tremblay and Pierre Borgnat. Graph wavelets for multiscale community mining. *IEEE Transactions on Signal Processing*, 62(20):5227–5239, 2014.

[45] Guy Wolf, Stéphane Mallat, and Shihab Shamma. Rigid motion model for audio source separation. *IEEE Trans. Signal Process.*, 64(7):1822–1831, 2016.

[46] Junfeng Wu, Ziyang Meng, Tao Yang, Guodong Shi, and Karl Henrik Johansson. Sampled-data consensus over random networks. *IEEE Trans. Signal Process.*, 64(17):4479–4492, 2016.

[47] Zhihua Zhang and Palle E. T. Jorgensen. Modulated Haar wavelet analysis of climatic background noise. *Acta Appl. Math.*, 140:71–93, 2015.

Index