WAVELET METHODS FOR THE INVERSION OF CERTAIN HOMOGENEOUS LINEAR OPERATORS IN THE PRESENCE OF NOISY DATA

A DISSERTATION

SUBMITTED TO THE DEPARTMENT OF STATISTICS

AND THE COMMITTEE ON GRADUATE STUDIES

OF STANFORD UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

By

Eric D. Kolaczyk

October 1994

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

David L. Donoho
(Principal Adviser)

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Iain M. Johnstone

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.

Richard A. Olshen

Approved for the University Committee on Graduate Studies:

Dean of Graduate Studies

Abstract

In this dissertation we explore the use of wavelets in certain linear inverse problems with discrete, noisy data. We observe discrete samples of a process $y(u) = (Kf)(u) + z(u)$, where K is a linear operator, z is a noise process, and f is a function we wish to recover from the data. In the problems that we consider, the inverse of K, K^{-1}, either does not exist or is poorly behaved. Such problems are termed *ill-posed* i.e., ones in which small changes in the data may lead to large changes in the recovered version of f. Our methods are most effective for problems where the operator K is homogeneous with respect to dilations, such as integration, fractional integration, convolution, and the Radon transform.

The theoretical framework in which we work is that of Donoho's (1992) Wavelet-Vaguelette Decomposition (WVD). The WVD uses wavelets and vaguelettes (almost wavelets) to decompose the operator K. Although this formally resembles the Singular Value Decomposition (SVD), the use of a wavelet basis means that functions f that exhibit spatial variability will be better represented with the WVD. When Kf is observed in the presence of noise, we can recover a denoised version of f by shrinking the wavelet coefficients towards zero. In the study of a simpler class of problems (where the operator K is the identity), Donoho & Johnstone show in a series of recent papers that wavelet shrinkage comes within a logarithmic factor of the optimal minimax risk, and significantly outperforms traditional linear methods in the case of a spatially variable function f. It is our goal to develop methods for inverse problems

that exhibit comparable advantages over traditional linear inversion techniques.

A primary motivation in our work is that of attacking the problem of reconstructing images from tomographic data using wavelets. In order to do so, we begin by developing tools for the calculation and use of Meyer wavelets. These wavelets have, in the frequency domain, compact support and a closed-form expression. Such characteristics make them particularly amenable to calculating the biorthogonal decomposition of operators that are well-understood in the frequency domain. We develop in detail algorithms for fast one- and two-dimensional Meyer wavelet transforms, and use these to calculate biorthogonal decompositions of certain self-adjoint, homogeneous operators. In particular, we consider the integration, fractional integration, and convolution operators.

With this background structure in place, we develop in detail algorithms for a biorthogonal decomposition of the Radon transform operator. Among other physical phenomena, this operator describes the mapping between a body or brain cross-section and the actual observed radiation counts in PET studies or the attenuation readings in X-ray transmission tomography. Our algorithms may be viewed as a multiresolution version of the traditional filtering of backprojected projections methods.

The biorthogonal decompositions of the Radon transform, differentiation, and fractional differentiation operators are finally used with wavelet shrinkage methods for recovery of signals and images in the context of inverse problems with noisy data. The shrinkage methods are adjusted so that the simple threshold used in non-inverse problems is replaced by a set of thresholds of increasing size. Such an adjustment is necessitated by the ill-posedness of the problems considered. We describe moderate-deviation probability calculations that give theoretical justification to these thresholds.

Acknowledgments

Contents

List of Tables

Chapter 1

Introduction and Background

1.1 Linear Inverse Problems

Linear inverse problems arise in a wide range of disciplines. Typically, they involve observed data from a process

$$y(u) = (Kf)(u) + z(u), \tag{1.1}$$

where K is a linear operator, z is a noise process, and f is a function. Usually f is assumed to be in $L^2(\mathbb{R}^d)$, for $d = 1$ or 2. The goal in these problems is to recover f from the observations $\{y(u)\}_{u\in\mathcal{U}}$ through some method of *inversion*. Reviews of such problems may be found in Bertero [4], O'Sullivan [42], and Wahba [57].

Examples that we examine in depth in this dissertation are

- Differentiation of a Noisy Signal (automatic control, velocity estimation)
- Inversion of Fractional Integral Transforms (Stereology)
- Inversion of the Radon Transform (Medical Imaging)
- Deconvolution (Imaging, signal analysis)

All these problems have the common features that the underlying operator, K, is homogeneous with respect to dilations and is most easily manageable in the frequency domain.

In statistical terminology, our task will be to estimate the underlying function f from a discrete set of observations that are both indirect and noisy. Since K is assumed to be known, the most obvious approach would be to apply the operator K^{-1} (or some discretization thereof) to the data. Unfortunately, in the problems that we consider, K^{-1} either is unstable or does not exist. Such problems are commonly called *ill-posed* inverse problems. Loosely speaking, a problem is ill-posed if small changes in the data may lead to large changes in the recovered version of f. Since we observe Kf in the presence of additive noise, this ill-posedness implies that using the same inversion method on two sets of data, sampled from the same underlying process, could yield drastically different reconstructions of f.

In attempting to deal with these conditions, researchers have devised numerous methods of *regularization*. As the recovery of higher frequency components of f is increasingly difficult, it is sought through these methods to recover the lower frequency components while damping out the high frequency effects of the noise. Such methods include, but are not limited to, Fourier filtering, representation by orthogonal series, iterative methods, and smoothing splines.

In this dissertation we look at a new set of tools for analyzing data: *wavelet methods*. In particular, we explore the use of these methods in the linear inverse problems specified above.

1.2 Why Wavelets ?

Wavelet methods involve the transformation of an object, say a signal or image, with respect to a basis of functions called *wavelets*, that are localized in both time and frequency. As a result of this special characteristic, wavelets have been found to be particularly adept at representing spatially variable objects. In this sense, they are somewhat complementary to Fourier methods, which are known to be rather inefficient in representing such objects.

Wavelet methods have many features that make them appealing for use in analyzing certain types of objects. First, they allow for the decomposition of an object into multiple resolutions (or scales). This is particularly advantageous for objects exhibiting a self-similarity when viewed at different magnifications. Second, the localization of wavelets in both time and frequency makes them quite useful for analyzing local features. Third, wavelets have been found to have excellent data compression capabilities for spatially variable objects, such as signals characterized by singularities or images determined primarily by a set of edges. Finally, simple nonlinear methods of denoising, based on thresholding of the wavelet coefficients, have been proven to be nearly optimal for a number of tasks across a wide range of function classes.

This last advantage refers primarily to the wavelet shrinkage methods of Donoho *et al.* [15]. Due to the compression advantages of wavelet methods, and the fact that discrete wavelet transforms are orthogonal, we have the following result: transforming a spatially variable function observed in the presence of Gaussian noise will yield a signal of the same length, with the relatively few large wavelet coefficients corresponding to the function, and the remaining smaller ones to the noise. These authors show in a series of papers that applying appropriate thresholds to the wavelet coefficients prior to inversion of the wavelet transform yields a denoised reconstruction of the function with many desirable characteristics. With the correct threshold level and thresholding function, their reconstructions will be at least as smooth as the object to be recovered, nearly optimal for the task of spatial adaptation, and will outperform traditional linear methods such as kernels, splines, and Fourier methods. The fact that the forward and inverse wavelet transforms of an n length signal can be accomplished with $O(n)$ operations means that these smoothing methods also admit fast algorithms.

The methods of Donoho *et al.* were developed and explored in the case of direct observations $y(u) = f(u) + z(u)$ i.e., where $K = I$ in (1.1). It is our goal in this dissertation to develop similar methods for the inverse problems mentioned above that exhibit comparable advantages over traditional linear inversion techniques.

1.3 Wavelets and Homogeneous Operators

The framework within which we work is that of the wavelet-vaguelette decomposition (WVD), a wavelet analogue of the singular value decomposition (SVD) [14]. The WVD uses wavelets and vaguelettes (almost wavelets) to decompose the operator K, instead of the eigenfunctions used by the SVD. This suggests that the WVD should be better at representing spatially variable functions. When the underlying operator is homogeneous with respect to dilations i.e.,

$$Kf_a = a^{\alpha}(Kf) \qquad \text{where} \qquad f_a(x) = f(ax), \tag{1.2}$$

the vaguelette functions (which decompose Kf-space) exhibit a scaling property similar to the wavelets. This property should enable the development of fast algorithms for the computation of the vaguelette coefficients, at which point methods similar to the wavelet shrinkage of Donoho *et al.* become feasible.

We concentrate on implementation of the WVD with the integration, fractional integration, and Radon transform operators. A primary motivation, and the main area of application, of our work is the task of reconstructing images from tomographic data. Often the primary reason for which such data are taken is to learn the edges of the objects being imaged. Thus, these images are ideal candidates for improvement via wavelet methods. In addition, the underlying operator is the Radon transform operator, which exhibits the homogeneity necessary for efficient algorithms for the WVD. As a first step in this direction, we begin by developing tools for the calculation and use of Meyer wavelets. These wavelets, in the frequency domain, have compact support and a closed-form expression. Such characteristics make them particularly amenable to calculating the biorthogonal decompositions of operators that are well-understood in the frequency domain.

We develop in detail algorithms for fast one- and two- dimensional Meyer wavelet transforms. While quite convenient for use with the Radon transform operator, they are especially useful for calculating biorthogonal decompositions of certain self-adjoint, homogeneous operators such as the integration, fractional integration, and convolution operators we consider. In the past, Meyer wavelets have not been used extensively for empirical work, being overshadowed by Daubechies wavelets and perhaps even considered impractical. However, the fast $O(n \log_2^2(n))$ algorithms we develop here make them practical, and their advantages in representing certain operators make them extremely attractive.

For the Radon transform operator, we devise a multiresolution analogue of the traditional filtering of backprojected projections method for the biorthogonal decomposition of this operator. When combined with a set of level-dependent thresholds, this analogue enables us to reconstruct images from tomographic data via wavelet shrinkage. The two types of data we consider in particular are those arising from parallel beam X-ray tomography and positron emission tomography. We use moderate deviation probability calculations to justify the thresholds in each case. Similar thresholds are calculated and combined with the corresponding biorthogonal decompositions in the cases of the integration, fractional integration, and convolution operators.

1.4 Related Work

Although the area is still in its infancy, there are already a number of researchers making sizable contributions to the use of wavelets in linear inverse problems.

In addition to Donoho's original work on the wavelet-vaguelette decomposition [14], there has been considerable work on operator biorthogonal decompositions. Lemarié [27] and Daubechies [9] have developed biorthogonal decompositions of the differentiation operator. Jawerth and Sweldens [24] [55] have decomposed variable-coefficient differential operators using biorthogonal decompositions. Also, Y. Wang [58] uses the WVD as a theoretical tool to decorrelate fractional Gaussian noise in function estimation problems from data with long-range dependence.

In the area of astronomy, Starck *et al.* [51] [52] [53] [54] have developed methods for image reconstruction in aperture synthesis and deconvolution problems that combine spherical wavelet transforms and the CLEAN algorithm. Further use of wavelets and multi-channel methods has been made by Galatsanos *et al.* in an effort to deblur images received from the (previously) defective Hubble Space Telescope.

Researchers at MIT recently have developed a sizable body of work using "multi-scale" methods, based on wavelets. This includes work by Luettgen [29] on regularization of ill-posed inverse problems, as well as Miller and Willsky [37] in an application to geology. Of particular interest to us is the work of Bhatia *et al.* [6], in which multiscale methods are used to perform tomographic reconstruction.

Other work in general tomographic reconstruction includes that of Peyrin *et al.* [44] and Sahiner and Yagle [47]. The latter have also explored the use of wavelets in limited angle tomography [48], as has Olson [41]. Finally, DeStephano and Olson [12] and Berenstein and Walnut [3] have used wavelets to localize the Radon transform in even dimensions. In a related direction, Biétrix *et al.* [7] and Letcher [28] have examined the use of wavelet methods in ultrasound imaging.

1.5 Overview

This thesis is arranged as follows. Chapter 2 contains a brief introduction to wavelets and wavelet methods as they are used here. We give a definition of wavelets and discuss the structure of orthonormal wavelet bases. We then define the associated discrete wavelet transform, in one and two dimensions. Next is a review of the key aspects of wavelet shrinkage methods. Finally, there is an introduction to the wavelet-vaguelette decomposition. In Chapter 3, we develop fast algorithms for one- and two-dimensional discrete, periodized Meyer wavelet transforms. Several applications of these algorithms are presented in Chapter 4, including the biorthogonal decomposition of self-adjoint, homogeneous operators. Chapter 5 is devoted to our work with the inversion of the Radon transform operator and its applications to tomographic data, including a detailed development of methods for computing a biorthogonal decomposition of the Radon transform and an assessment of the quality of these methods for noiseless data. In order to deal with noisy observations, we present sets of level-dependent thresholds corresponding to the settings of parallel beam X-ray tomography and positron emission tomography. The moderate deviation probability calculations used to justify these thresholds are shown. Finally, in Chapter 6 we summarize our work, offer some commments in the context of the contributions of others, and discuss future directions. The first appendix, A, is used to present the software tools implementing our methods and explain their usage. Appendix B describes the software that generated the illustrations herein, and Appendix C comments on issues regarding the implementation of our software tools with standard programming languages.

Chapter 2

Review of Wavelets and Wavelet Methods

2.1 Orthonormal Wavelet Bases

Wavelets may be defined informally as *special collections of localized wiggles.* Wavelets are "local" in that they have compact support, or nearly so, in the time domain. They are "wiggles" in that their integral, and quite often higher order moments, are required to be zero. Finally, they are "special" in that when a specific set of dilations and translations of a single wavelet function are collected, they form an orthonormal basis of $L^2(\mathbb{R})$. The stunning popularity and usefulness of these relatively new bases stem from the fact that each basis function is localized to both a particular position and resolution (akin to frequency), and hence together they may be used to obtain a time-scale decomposition of a given function.

Speaking more carefully now, a function $\psi \in L^2(\mathbb{R})$ is defined to be a *wavelet of class m* (Meyer [35]) if

1. The set $\{\psi_{j,k}\}$ forms an orthonormal basis of $L^2(\mathbb{R})$, where

$\psi_{j,k}(x) = 2^{j/2}\psi(2^j x - k)$, $j, k \in \mathbb{Z}$.

2. $\psi^{(l)}(x) \in L^\infty(\mathbb{R})$ for $l = 0, \ldots, m$

3. $\psi^{(l)}(x)$ decreases rapidly as $x \to \pm\infty$, for $l = 0, \ldots, m$

4. $\int_{-\infty}^{\infty} x^l \psi(x) dx = 0$, for $l = 0, \ldots, m$

Property (2) dictates the regularity of ψ, and hence its localization in the frequency domain, while (3) dictates its localization in the time domain. Property (4) corresponds to the oscillation ('wiggling') of ψ. Note that the $\psi_{j,k}$ inherit these properties trivially.

The structure of the orthonormal basis in (1) is perhaps best described within the context of Mallat and Meyer's *multiresolution analysis* (MRA) [31] [34]. They define a MRA of $L^2(\mathbb{R})$ to be a set of increasing, closed subspaces, $\{V_j\}_{j\in\mathbb{Z}}$, such that

(a) $\cap_{-\infty}^{\infty} V_j = 0$ and $\overline{\cup_{-\infty}^{\infty} V_j} = L^2(\mathbb{R})$

(b) $f(\cdot) \in V_0 \Leftrightarrow f(2^j \cdot) \in V_j, \quad \forall j \in \mathbb{Z}$

(c) $\exists \phi \in V_0$ such that $\{\phi(\cdot - k)\}_{k\in\mathbb{Z}}$ forms an orthonormal basis of V_0

If we define the projection of $f \in L^2(\mathbb{R})$ onto V_j to be f_j, then properties (b) and (c) above imply that

$$f_j(x) = \sum_{k=-\infty}^{\infty} \langle \phi_{j,k}, f \rangle \phi_{j,k}(x), \tag{2.1}$$

where $\phi_{j,k}(x) = 2^{j/2}\phi(2^j x - k)$. The function f_j thus provides a course approximation to f that may be expressed in terms of $\{\phi_{j,k}\}_{k\in\mathbb{Z}}$. The detail gained in moving from f_j to f_{j+1} may be expressed as $d_j(x) = f_{j+1}(x) - f_j(x)$, which is contained in $W_j = V_{j+1} \ominus V_j$, the orthogonal complement of V_j in V_{j+1}. The subspace W_j possesses an orthonormal basis $\{2^{j/2}\psi(2^j \cdot -k)\}_{k\in\mathbb{Z}}$ for a wavelet ψ defined in terms of ϕ. One generally has $f_j(x) \to f(x)$ as $j \to \infty$. Hence from

$$f_{j_1}(x) = \sum_{k=-\infty}^{\infty} \langle \phi_{j_0,k}, f \rangle \phi_{j_0,k}(x) + \sum_{j_0 \le j < j_1} \sum_{k=-\infty}^{\infty} \langle \psi_{j,k}, f \rangle \psi_{j,k}(x)$$

and letting $j_1 \to \infty$, we see that we may write

$$f(x) = \sum_{k=-\infty}^{\infty} \langle \phi_{j_0,k}, f \rangle \phi_{j_0,k}(x) + \sum_{j \ge j_0} \sum_{k=-\infty}^{\infty} \langle \psi_{j,k}, f \rangle \psi_{j,k}(x), \tag{2.2}$$

for fixed $j_0 \in \mathbf{Z}$. Letting $j_0 \to -\infty$ gives us f expressed in terms of the orthonormal basis $\{\psi_{j,k}\}_{j,k \in \mathbf{Z}}$.

Numerous examples of pairs (ϕ, ψ) have been derived, often possessing various properties additional to the ones listed above. Figure 2.1 shows a few such pairs. Daubechies [10] gives explicit formulas and conditions for the creation of valid ϕ, often called the *scale function* or *father wavelet*, and the corresponding ψ, called the *mother wavelet*. A more complete explanation of these procedures, including strict definitions of the convergence involved, may be found in Daubechies [10] or Meyer [35]. Alternatively, Meyer [36] gives more of an overview of these topics within the general context of time-frequency and time-scale methods, and discusses a number of applications.

2.2 Discrete Wavelet Transforms

2.2.1 One-Dimensional Transforms

In the same way that the methods of Fourier analysis have been discretized and fast algorithms created for the computation of these discrete transforms, so too for wavelets. The orthonormal wavelet bases of the above section may be adapted to $L^2([0,1])$, either by periodization or boundary correction, and from there a discrete wavelet transform (DWT) may be defined.

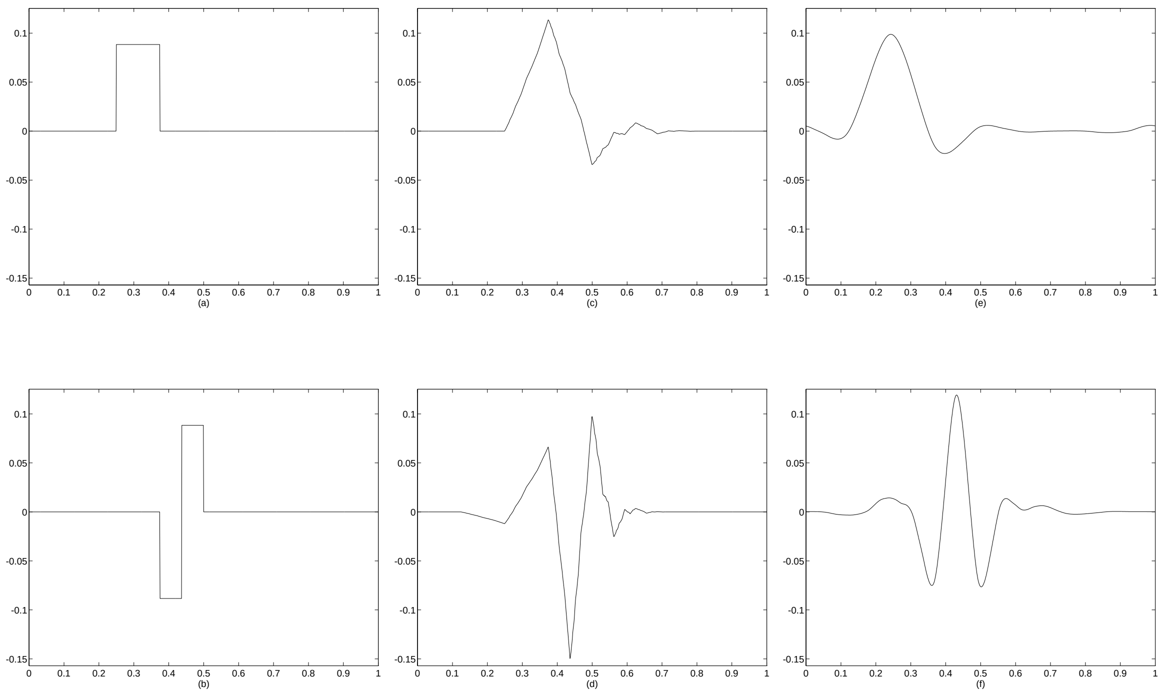


Figure 2.1: Selected pairs of 'father' and 'mother' wavelets at $j = 3$, $k = 3$. Figures (a) and (b) are the Haar wavelet pair, (c) and (d) the Daubechies6 pair, and (e) and (f) the Symmlet8 pair.

The DWT of a vector $\mathbf{x}$, of length n, may be thought of as the result, $\mathbf{w}$, of multiplication by an $n \times n$ orthogonal matrix $\mathcal{W}$ i.e., $\mathbf{w} = \mathcal{W}\mathbf{x}$. Each row of $\mathcal{W}$ may be thought of as samples of a particular wavelet at the positions t/n, $t = 0, 1, \ldots, n-1$. In fact, Donoho and Johnstone [17] note that

$$\sqrt{n}\ \mathcal{W}_{j,k}(t) \approx 2^{j/2}\psi(2^j \frac{t}{n} - k), \tag{2.3}$$

where $\mathcal{W}_{j,k}(t)$ is the $(t+1)st$ element of the $(2^j + k + 1)th$ row of $\mathcal{W}$. Since $\mathcal{W}$ is orthogonal, the original vector $\mathbf{x}$ may be recovered from $\mathbf{w}$ as $\mathbf{x} = \mathcal{W}^{-1}\mathbf{w}$.

The vector $\mathbf{w}$ is composed of a telescoping series of finite sets of empirical wavelet coefficients, corresponding to information at finer and finer resolutions. Specifically, let $n = 2^J$, for some $J > 0$, and fix $0 \leq C < J$. Then the first 2^C entries of $\mathbf{w}$ are associated with the father wavelet at level C and positions $k = 0, 1, \ldots, 2^C - 1$. These coefficients contain the course structure of or information in $\mathbf{x}$. Following these initial coefficients, the entries at $\{2^j+1,\ldots,2^{j+1}\}$ are associated with the mother wavelet at level j and positions $k=0,1,\ldots,2^j-1$. These coefficients contain the details, at finer and finer resolutions, left behind by the course level coefficients. Figure 2.2 shows a spatially variable function, ‘Bumps’, and its discrete wavelet transform. The x-axis of the coefficient plot roughly corresponds to the same positions in the plot of ‘Bumps’. The y-axis corresponds to resolution, course to fine, as one moves from top to bottom. Note the extreme sparseness of the transform, as well as the localization of the large coefficients to positions where ‘Bumps’ exhibits a singular behavior.

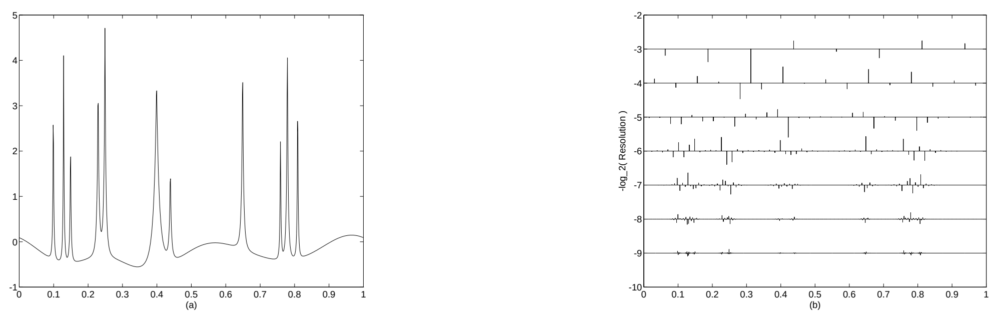


Figure 2.2: (a) The periodic function ‘Bumps’. (b) The Daubechies6 wavelet transform of ‘Bumps’.

Computation of the forward and inverse DWTs would naturally be expected to require $O(n^2)$ operations. Fortunately, a pyramid filtering algorithm due to Mallat [30] reduces this cost to only $O(n)$ operations. This is especially impressive when compared with the $O(n\log(n))$ operations required for the FFT. Mallat's algorithm works by expressing the action of $\mathcal{W}$ in terms of a pair of low and high pass filters, $\mathbf{L}$ and $\mathbf{H}$, whose forms are intimately connected to ϕ and ψ. Briefly put, the $(J-l)th$ level of detail coefficients in $\mathbf{w}$ is obtained by passing $\mathbf{x}$ through $\mathbf{L}$ $l-1$ times, followed by $\mathbf{H}$, downsampling by a factor of 2 at each stage. The final course level of coefficients is the result of applying $\mathbf{L}$ to $\mathbf{x}$ $J-C$ times, with downsampling. To recover $\mathbf{x}$ from $\mathbf{w}$, the process is simply reversed.

Empirical work in wavelets is most commonly done with compactly supported ϕ and ψ, since this yields filters $\mathbf{L}$ and $\mathbf{H}$ of finite length. In Chapter 3, we develop in detail algorithms for computing the DWTs for the Meyer wavelet, which has infinite support. These algorithms are decidedly different in nature from those of Mallat.

2.2.2 Two-Dimensional Transforms

The two-dimensional discrete wavelet transform takes an $n \times n$ matrix, $\mathbf{X}$, and outputs an $n \times n$ matrix $\mathbf{W}$ of empirical wavelet coefficients. This two-dimensional transform is based on an orthonormal wavelet basis of $L^2(I\!R^2)$, formed from all dyadic dilations and translations of the basis functions $\phi(x)\psi(y)$, $\psi(x)\phi(y)$, and $\psi(x)\psi(y)$. As a result, the two- dimensional matrix $\mathbf{W}$ of empirical wavelet coefficients may be broken down into blocks, similar to the manner in which the vector $\mathbf{w}$ was broken down into subvectors. This block structure is shown schematically in Figure 2.3. Again, take $n = 2^J$, fix $0 \le C < J$. The three largest blocks are of dimension $2^{J-1} \times 2^{J-1}$, corresponding to details at resolution $J-1$. The next three largest are of dimension $2^{J-2} \times 2^{J-2}$, at resolution level $J-2$, and so forth, down to the block in the upper left-hand corner, which is of dimension $2^C \times 2^C$ and contains the coarse level coefficients.

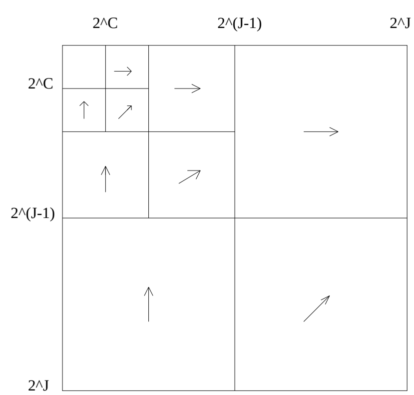


Figure 2.3: Schematic representation of the 2D wavelet transform coefficient matrix.

When $\mathbf{X}$ is a matrix of pixel strengths for an image,the upper right-hand corner of a given level of wavelet coefficients can be thought of as containing horizontal characteristics of $\mathbf{X}$, the lower left-hand corner, vertical characteristics, and the lower right-hand corner, diagonal. Note that the indexing in two dimensions is of the form $(j, (k_r, k_c))$, where j is the resolution index, k_r the position index for the rows, and k_c the position index for the columns. Computation of $\mathbf{W}$ from $\mathbf{X}$, is accomplished by applying the same filters $\mathbf{L}$ and $\mathbf{H}$ of the one-dimensional algorithms alternately on rows and columns at each stage. Inversion of $\mathbf{W}$ requires reversing the process.

2.3 Wavelet Shrinkage

In statistics, the most extensive study of wavelets and their usage has been made by Donoho and Johnstone [17] [18] [19] [20] and Donoho *et al.* [15] [16] in their *wavelet shrinkage* methods. As mentioned, these authors work with the direct problem of recovering a function f from its noisy observations. Specifically, they consider observations $y_i = f(i/n) + \sigma z_i$, for $i = 0, \ldots, n-1$, where the z_i are independent and identically distributed standard normals. A particular example of wavelet shrinkage is their proposal *VisuShrink*, involving three steps:

1. Compute the wavelet transform, $w = \mathcal{W}y$.
2. Apply a soft threshold, yielding $\tilde{w} = \eta_{t_n}(w)$.

   Here $\eta_t(\omega) = sgn(\omega)(|\omega| - t)_+$, and $t_n = \sqrt{2\log(n)}\ \frac{\sigma}{\sqrt{n}}$
3. Compute the inverse wavelet transform, $\tilde{f} = \mathcal{W}^{-1}\tilde{w}$

The term $\sqrt{2\log(n)}$ in the threshold t_n derives from the fact that for $Z_1, \ldots, Z_n$ i.i.d. standard normal,

$$\Pr\{\max|Z_i| > \sqrt{2\log(n)}\ \} \rightarrow 0$$

as $n \to \infty$. Since $\mathcal{W}$ is orthogonal, we have

$$w_{j,k} = \theta_{j,k} + \frac{\sigma}{\sqrt{n}} \; z_i' \quad \forall j, k$$

where $\theta_{j,k} \approx \langle f, \psi_{j,k} \rangle$ and the z_i' are again i.i.d. standard normals. Hence $\sqrt{2\log(n)}$, when adjusted by $\sigma/\sqrt{n}$, is a probabilistic upper bound on the noise level in wavelet space. Thresholding with this value should leave behind primarily the large underlying wavelet coefficients.

Figure 2.4 demonstrates the wavelet shrinkage process for the function 'Bumps'. Figure 2.4(a) shows 'Bumps' with Gaussian noise added, and Figure 2.4(b) shows the corresponding wavelet transform. Compare this latter figure to Figure 2.2(b). Note how the additive, Gaussian noise has been spread evenly throughout the coefficient levels by the wavelet transform. The majority of the few large coefficients are associated with the underlying function. This plot gives a visual motivation for using simple thresholds to denoise spatially variable signals that can be highly compressed by wavelet transforms. Figure 2.4(c) shows the wavelet coefficients after thresholding, and Figure 2.4(d) presents the result of inverting these coefficients. Note that the noise has been effectively suppressed while still resolving each of the peaks.

In addition to such empirical evidence of success, Donoho and Johnstone showed extensively in a series of papers that this apparently simple nonlinear method has a whole host of theoretically optimal properties and advantages over traditional linear methods. First, the estimate $\tilde{f}$ will be at least as smooth as f with high probability, indicating that wavelet shrinkage should not introduce spurious artifacts into the reconstruction. Second, the risk associated with $\tilde{f}$ will be within a $\log^2(n)$ factor of the "ideal" risk one could achieve with piecewise polynomial fits and prior knowledge of the best knot placement. This suggests that their estimate is near-optimal for spatial adaptation. Third, $\tilde{f}$ will achieve a risk within a logarithmic factor of the optimal minimax risk, for either global or pointwise estimation, across a broad range

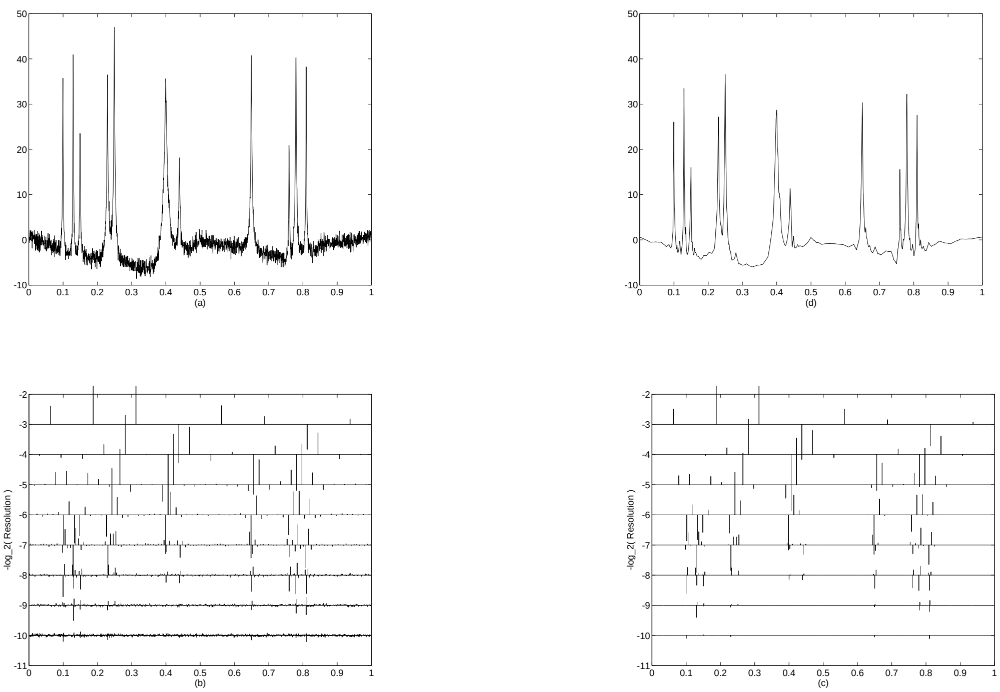


Figure 2.4: Example of function reconstruction using wavelet shrinkage. Figure (a) shows the function 'Bumps' with Gaussian noise added, while Figure (b) shows the corresponding wavelet transform. The result of passing the wavelet coefficients through a soft threshold is displayed in Figure (c). The reconstruction of 'Bumps', obtained by inverting the thresheld coefficients, appears in Figure (d).

of function classes, using a variety of loss functions.

It is these properties and the fact that they imply that wavelet shrinkage outperforms such traditional linear methods as kernels, splines, and Fourier methods, that motivates us to explore their usage in linear inverse problems. Our hope is to develop and implement efficient methods to use wavelet shrinkage in the linear inverse problems already mentioned that will exhibit similar advantages over traditional linear methods of inversion.

2.4 The Wavelet-Vaguelette Decomposition

We have already observed the additional difficulties one encounters in moving from a direct reconstruction problem to an indirect one. In the wavelet framework this difficulty may be expressed as follows. Setting aside, for the moment, considerations of discreteness and noise, suppose we would like to have information on the function f, but only have information on Kf, where K is a linear operator. For a selected orthonormal wavelet basis, $\{\psi_{j,k}\}_{j,k\in\mathbb{Z}}$, the coefficients $\langle Kf, \psi_{j,k}\rangle$ will generally not be equal to $\langle f, \psi_{j,k}\rangle$. Hence, although f may be a function that can be efficiently represented by wavelets, we cannot recover f from

$$\sum_{j\in\mathbb{Z}}\sum_{k\in\mathbb{Z}}\langle Kf, \psi_{j,k}\rangle\psi_{j,k},$$

(which is, of course, actually just Kf for well behaved K).

Donoho [14] created the wavelet-vaguelette decomposition (WVD) in response to these difficulties. His framework allows one to represent f by an orthonormal wavelet basis, and to represent Kf by an associated near-orthonormal basis of 'almost' wavelets, called vaguelettes. In relating the coefficients of these two bases, one obtains a biorthogonal decomposition of the operator K and the ability to represent the information in f in terms of wavelets. In other words, the result is a wavelet analogue of the singular value decomposition (SVD). In cases where f is a spatially variable function, the WVD should better recover the information in f from Kf than the SVD.

We summarize briefly the derivation of the WVD. Let $\mathcal{D}(K)$ and $\mathcal{R}(K^*)$ be the domain and range of K and its adjoint K^*, respectively. K is assumed to be an operator, not necessarily bounded, from $L^2(dt)$ into $L^2(du)$. Let $\{\psi_\lambda\}$ be a specific orthonormal wavelet basis, such that

$$\psi_\lambda \in \mathcal{D}(K)\cap\mathcal{R}(K^*)$$

for all λ in some index set Λ. Then there exist bounded linear functionals $c_\lambda : L^2(du) \to I\!R$ such that

$$c_\lambda(Kf) = \langle f, \psi_\lambda \rangle,$$

and for all finite sums

$$f = \sum_\lambda \langle f, \psi_\lambda \rangle \psi_\lambda$$

we have the formula

$$f = \sum_\lambda c_\lambda(Kf) \psi_\lambda$$

which reproduces f from Kf.

The equivalence of $c_\lambda(Kf)$ and $\langle f, \psi_\lambda \rangle$ may be seen by the following argument. Under the given conditions, there exists a bounded linear functional $s_\lambda : L^2(dt) \to I\!R$ with Riesz representer σ_λ, such that $s_\lambda(K^*Kf) = \langle f, \psi_\lambda \rangle$. This allows us to write

$$\begin{aligned} \langle \psi_\lambda, f \rangle &= s_\lambda(K^*Kf) \\ &= \langle \sigma_\lambda, K^*Kf \rangle \\ &= [K\sigma_\lambda, Kf] \\ &:= [\gamma_\lambda Kf] \end{aligned}$$

When K is a homogeneous, linear operator, scaling the collection of functions $\{\gamma_\lambda\}_{\lambda \in \Lambda}$ by resolution dependent constants yields a near-orthonormal basis of vaguelettes. These vaguelettes will have localization and cancellation properties similar to the wavelets, as well as being represented as dilations and translations of a single 'mother' vaguelette.

From an empirical point of view the WVD provides us with a method of obtaining (approximate) reconstructions of f from observations of Kf. If Kf is observed in the presence of Gaussian noise, as in (1.1), we should be able to obtain a denoised reconstruction

$$\tilde{f} = \sum_{\lambda} \eta_{t_n}(c_\lambda(y))\psi_\lambda$$

by applying thresholds to the empirical coefficients. In order to accomplish such a task, two primary challenges face us. First, efficient algorithms need to be designed for calculating the empirical coefficients associated with a given operator K. As noted before, simply taking DWTs of the data will not yield the desired result. Second, due to the ill-posedness of the problems we consider, the simple threshold of Donoho and Johnstone must be replaced by a set of resolution dependent thresholds.

We shall begin by addressing the first challenge in the next chapter. All of the operators we consider in this dissertation (and hence their biorthogonal decompositions) are most easily managed in the frequency domain. This would suggest that, all else being equal, we should choose a basis of wavelets that possess a similar property. The compactly supported wavelets of Daubechies that are currently used in most empirical work do not have this property. An obvious substitute is the Meyer wavelet, which possesses a closed form in the frequency domain. Unfortunately, as the Meyer wavelet has infinite support in the time domain, it does not lend itself well to computation with the typical pyramid filtering algorithms. In Chapter 3 we develop in detail a computational infrastructure for the use of Meyer wavelets with one and two dimensional discrete problems.

Chapter 3

Meyer Wavelet Transforms

3.1 Overview

In this chapter we present our algorithms for fast $O(n \log_2^2(n))$ forward and inverse discrete, periodized Meyer wavelet transforms. These algorithms are decidedly different in character from the $O(n)$ algorithms used to compute the same transforms using Daubechies wavelets. Instead of convolving the data in the time domain with finite length filters in a pyramidal fashion, our algorithms take place in the frequency domain and associate each level of coefficients with a projection of the signal onto a particular frequency band. We use the folding, polarity, and projection concepts of Coifman & Meyer [8], Auscher, Weiss, & Wickerhauser [1], and Meyer [36] to represent each projection in terms of a local trigonometric basis (also see Malvar [32] and Wickerhauser [59]). It is with respect to these trigonometric bases that the actual computations are done, yielding a set of algorithms composed of specific discrete sine and cosine transforms, coupled with simple folding and windowing operations.

We will derive our algorithms in the context of one-dimensional data. We refer the reader to Section 3.6 for a discussion of our implementation of similar algorithms in two dimensions. The object to be transformed, $\{f_0, \ldots, f_{n-1}\}$, will be a vector of length $n = 2^J$, for integer $J > 3$, composed of equispaced observations of a function f at $\{0, \frac{1}{n}, \ldots, \frac{n-1}{n}\}$. The vector of wavelet coefficients will be expressed as

$$\{\beta^{\circ}_{C,0}, \ldots, \beta^{\circ}_{C,2^C-1}, \alpha^{\circ}_{C,0}, \ldots, \alpha^{\circ}_{C,2^C-1}, \ldots\ldots, \alpha^{\circ}_{J-1,0}, \ldots, \alpha^{\circ}_{J-1,2^{J-1}-1}\},$$

where $\{\beta^{\circ}_{C,k}\}_{k=0}^{2^C-1}$ is the coarse level of coefficients, and $\{\alpha^{\circ}_{j,k}\}_{k=0}^{2^j-1}$ is the jth detail level, for $j = C, \ldots, J-1$ and $C \geq 3$.

The forward Meyer wavelet transform (FMWT) works by first moving to the frequency domain and separating the signal into real and imaginary parts. From here, a set of windowing, folding, and trigonometric transform operations are computed for each level of coefficients. The general form of the FMWT follows.

FMWT Outline

1. Compute FFT of the data.
2. Separate into real and imaginary parts.
3. For each level of wavelet coefficients,
   (a) Window the vectors for each part over the appropriate frequency band. Keep only elements within the 'support' of the window.
   (b) Fold each resulting vector according to specific polarities.
   (c) Compute a discrete sine or cosine transform of each folded vector.
   (d) Combine local sine and cosine coefficients to obtain wavelet coefficients for this level.

Here Fold and Window refer to specific operations to be described in full below. The concept of Polarity will also be described. We adopt the convention of printing the names of such concepts in a slightly thicker font, in order to alert the reader to their importance in the work that follows, as well as the fact that their definitions, given below, may differ slightly from those used in other areas of the literature.

In the FMWT, the frequency band onto which the signal is to be windowed varies with coefficient level j, and is roughly equal to the frequency domain support of the wavelets underlying that level. The folding and trigonometric transforms are intimately connected with the particular sine/cosine basis used to represent certain projections defined in terms of the windowing operation, and the associated symmetries thereof. The specifics of Step 3 will be presented, for the coarse, detail, and finest detail levels of coefficients, in Section 3.4.

The inverse Meyer wavelet transform (IMWT) algorithm works approximately in the reverse fashion. The goal is to recover a projection of the signal onto a particular frequency band from each level of wavelet coefficients. Summing the projections across all frequency bands yields the complete signal in the frequency domain, allowing recovery of the original signal via the Fourier transform. The general form of the IMWT follows.

IMWT Outline

1. For each level of wavelet coefficients,
   (a) Separate into local sine and cosine coefficients.
   (b) Compute inverse discrete sine and cosine transforms.
   (c) Unfold resulting vectors according to specific polarities.
   (d) Window unfolded vectors.
   (e) Extend to obtain projections of real and imaginary parts of the Fourier transform of the signal onto the discrete circle.
2. Sum across all projections to recover Fourier transform of signal.

3. Compute IFFT to recover original signal.

Here, again, Unfold and Extend refer to specific operations to be described in full below. For each level, the wavelet coefficients are 'separated' so as to yield two sets of local sine and cosine coefficients. To these coefficient sets we apply the inverse transform, and the resulting vectors are unfolded and windowed, to produce the relevant projection. For the detail levels, the windows used in projecting are symmetric and bimodal. As a result, we compute only the projection of one mode and then extend by symmetry (or antisymmetry). The specifics of Step 1 will be presented, for the coarse, detail, and finest detail of coefficients, in Section 3.5.

We begin in Section 3.2 by presenting certain fundamental concepts relating to local trigonometric bases, as well as software tools to implement them with data. We then derive necessary relationships between Meyer wavelet coefficients and certain local trigonometric coefficients in Section 3.3. Our approach will be to begin with the case of functions on $L^2(\mathbb{R})$, continue with that of $L^2([0,1])$, and finish with functions on the discrete circle. Sections 3.4 and 3.5, as mentioned, present the details of the main components of the forward and inverse algorithms. In Section 3.6, we briefly outline the manner in which two-dimensional Meyer wavelet transforms may be computed using algorithms similar to the one-dimensional setting. Finally, in Section 3.7 we discuss our implementation of these various algorithms in Matlab.

3.2 Concepts

In this section we define basic concepts underlying the local trigonometric bases of Coifman & Meyer [8] that will be fundamental in creating our algorithms. Our notation will closely parallel that of Auscher, Weiss, & Wickerhauser [1]. For each concept, we create a corresponding software tool to implement that concept in our algorithms. This approach is similar to that of Wickerhauser [60]. These tools are written in a pseudo-code manner, imitating Knuth's WEB language [25], to allow for easier reading. Implementation of these tools in a standard programming language may be accomplished fairly easily, as we have done using the Matlab language. A more detailed discussion of these issues appears in Appendix C.

The concepts in this section are presented in a natural order, beginning with the definition of a bell function, continuing with the use of such bell functions to create 'smooth' orthogonal projections, and finishing with the representation of these projections by means of local trigonometric bases. We then discuss the reverse of this process, whereby the projection of a function is recovered from its local trigonometric coefficients.

3.2.1 Bell Functions

Given an interval $I = [\alpha, \beta]$ and two constants $\epsilon, \epsilon' > 0$, a *bell function over I* is defined as

$$b_I(x) \equiv s_\epsilon(x-\alpha) \cdot c_{\epsilon'}(x-\beta),$$

where

$$s_\epsilon(x) = \sin\Big[\int_{-\infty}^{x} v_\epsilon(t)dt\Big] \qquad c_{\epsilon'}(x) = \cos\Big[\int_{-\infty}^{x} v_{\epsilon'}(t)dt\Big]$$

and $v_\epsilon(\cdot)$ is an even, non-negative function with support in $[-\epsilon, \epsilon]$. Such a function $b_I(\cdot)$ can be characterized by its behavior on the three subintervals $[\alpha-\epsilon, \alpha+\epsilon]$, $[\alpha+\epsilon, \beta-\epsilon']$, and $[\beta-\epsilon', \beta+\epsilon']$. In the middle interval b_I is just equal to 1, while on the left and right intervals it is defined entirely by $s_\epsilon(x-\alpha)$ and $c_{\epsilon'}(x-\beta)$, respectively.

The use of 'cos' and 'sin' may be motivated as follows. Looking ahead to the smooth projections of the next subsection, we would like to obtain the property that if I and J are adjacent intervals, $J = [\beta, \gamma]$ with constants $\epsilon', \epsilon'' > 0$, then

$$b_{I\cup J}^2 = b_I^2 + b_J^2, \tag{3.1}$$

where

$$b^2_{I\cup J} \equiv s_\epsilon(x-\alpha)\cdot c_{\epsilon''}(x-\gamma).$$

Choosing the function v_ϵ in such a way that $\int_{-\infty}^{x} v_\epsilon(t)dt$ has a simple closed form, we introduce the following software tool to window a vector $\mathtt{f}$=($\mathtt{f}_1, \ldots, \mathtt{f}_n$) by a discrete version of the bell function b_I.

---

define `y` $\equiv$ `WINDOW( f, bI, [a,b], (e,e') )`

{*Compute Windowed Version of* `f`}

Variables of `WINDOW` $\equiv$

`f` := *input vector of length* n

`bI` := *(string) name of bell function with which to window*

`[a,b]` := *(real) endpoints of bell interval*

`(e,e')`:= *(real) overlap to left and right of* `a` *and* `b` *of bell function*

`y` := *output vector of length* $m = card([\mathtt{a}-\mathtt{e}, \mathtt{b}+\mathtt{e}'] \cap \boldsymbol{Z})$

This tool multiplies the input vector `f` by the bell function `bI` and stores the resulting nonzero values (i.e. those in the support of b_I) in the array `y`.

3.2.2 Smooth Projections

Given a bell function b_I over an interval I and a function $f \in L^2(I\!R)$, the *smooth (orthogonal) projection of f onto I* is defined as

$$(P_I f)(x) = b_I^2(x)f(x) \pm b_I(x)b_I(2\alpha - x)f(2\alpha - x) \pm b_I(x)b_I(2\beta - x)f(2\beta - x). \quad (3.2)$$

This projection is 'smooth' as compared to χ_I, because b_I overlaps I slightly, tending to zero quickly. Note that actually *four* projections are defined by (3.2), depending on the two choices of $\pm$ in front of the second and third terms. This pair of choices is called the *polarity of the projection P_I*, or sometimes the *polarity of I*, and is written

$(\pm, \pm)$. The polarity of a projection becomes particularly important when attempting to combine projections of two adjacent intervals, as we shall want to do in Section 3.3. Specifically, for adjacent intervals I and J defined as above, we have the relation

$$P_{I \cup J} = P_I + P_J$$

if P_I and P_J have opposite polarities at β. This follows directly from Equations (3.2) and (3.1).

We define no software tool explicitly for this concept of projection, although implicitly the concept is crucial for the material in Section 3.3.

3.2.3 Folding

We would like to represent the projections we have defined above in terms of a set of smooth basis functions. As a first step in this direction, we use the symmetries/antisymmetries induced by our definition to introduce the concept of folding. Consider writing the projection $(P_I f)$ as

$$\begin{aligned} (P_I f)(x) &= b_I(x)\{\ b_I(x) f(x) \pm b_I(2\alpha - x) f(2\alpha - x) \pm b_I(2\beta - x) f(2\beta - x)\ \} \\ &\equiv b_I(x) S(x). \end{aligned} \tag{3.3}$$

The function S is supported on $[\alpha - \epsilon, \beta + \epsilon']$ and has symmetries and/or antisymmetries determined by the polarity of I. Specifically, S will be symmetric (antisymmetric) about α in $[\alpha - \epsilon, \alpha + \epsilon]$ if $-$ $(+)$ is chosen for the second term. The behavior about β is determined similarly by the choice of sign before the third term.

Because of these symmetries/antisymmetries, it is sufficient to know S on the interval I alone. The restriction of S to I is called the *folded version of f with respect to b_I*. We create a folding tool to calculate an analog of S for discrete data,

---

define `fldy` $\equiv$ `FOLD( y, [a,b], (e,e'), pol )`

{*Fold Windowed Vector* }

Variables of function `FOLD` $\equiv$

`y` := *vector of length* m *resulting from* `WINDOW`

`[a,b],(e,e')` := *defined as in* `WINDOW`

`pol` := *(string) polarity of* `[a,b]`

`fldy` := *vector of length* $card([\mathtt{a},\mathtt{b}] \cap \mathbb{Z})$

This tool takes `y`, the windowed version of `f`, and folds it in a manner consistent with the definition of $S(\cdot)$.

3.2.4 Local Trigonometric Transforms

The following four *local sine and cosine bases* of Coifman and Meyer form orthonormal bases of $P_I L^2(I\!R)$, the image of $L^2(I\!R)$ under P_I, when I has the indicated polarity.

$$\begin{aligned}
(LST-I)\ \ (-,-) \quad &\Rightarrow \quad \Big\{ \sqrt{\frac{2}{|I|}} b_I(x) \sin[k \frac{\pi}{|I|}(x-\alpha)] \Big\}, \ \ k = 1,2,\ldots \\
(LST-III)\ \ (-,+) \quad &\Rightarrow \quad \Big\{ \sqrt{\frac{2}{|I|}} b_I(x) \sin[\frac{2k+1}{2} \frac{\pi}{|I|}(x-\alpha)] \Big\}, \ \ k = 0,1,2,\ldots \\
(LCT-I)\ \ (+,+) \quad &\Rightarrow \quad \Big\{ \frac{1}{\sqrt{|I|}} b_I(x), \sqrt{\frac{2}{|I|}} b_I(x) \cos[k \frac{\pi}{|I|}(x-\alpha)] \Big\}, \ \ k = 1,2,\ldots \\
(LCT-III)\ \ (+,-) \quad &\Rightarrow \quad \Big\{ \sqrt{\frac{2}{|I|}} b_I(x) \cos[\frac{2k+1}{2} \frac{\pi}{|I|}(x-\alpha)] \Big\}, \ \ k = 0,1,2,\ldots
\end{aligned}$$

These bases are arrived at by choosing a trigonometric system for $L^2(I)$ whose basis functions share the polarity of I, and windowing them with the bell b_I. The orthonormality of the local trigonometric system follows from that of the trigonometric system and the symmetries/antisymmetries of S.

For example, if I has the polarity $(-,+)$, the system

$$\left\{ \sqrt{\frac{2}{|I|}} \sin[\frac{2k+1}{2}\frac{\pi}{|I|}(x-\alpha)] \right\} \tag{3.4}$$

is an orthonormal basis of $L^2(I)$ whose basis functions share the same polarity. The orthonormality of the LST-III system above then follows from the relation

$$\begin{aligned} & \int_{\alpha-\epsilon}^{\beta+\epsilon'} g(x) b_I(x) \sin\Big[\frac{2k+1}{2}\frac{\pi}{|I|}(x-\alpha)\Big]\,dx \\ = & \int_{\alpha}^{\beta} S(x) \sin\Big[\frac{2k+1}{2}\frac{\pi}{|I|}(x-\alpha)\Big]\,dx, \end{aligned} \tag{3.5}$$

where $g(x) = b_I(x)S(x)$ is a function in $P_I L^2(I\!R)$, and $S(\cdot)$ is a function supported on I with polarity $(-,+)$. See Auscher, Weiss, and Wickerhausser [1] for a more complete proof.

The terminology 'LST-III' derives from similar terminology in electrical engineering (see Rao and Yip [45]), which defines four types of discrete sine transforms (DST) and four types of discrete cosine transforms (DCT). The discrete analogue of the basis in (3.4), where $x \in \{1, \ldots, n\}$ and $|I| = n$, is what engineers would describe as being of type DST-III. When we insert the bell b_I, we refer to the new basis as being of type LST-III.

The identity (3.5) says literally that a local sine (cosine) transform of the projection of f onto I may be computed by windowing f, folding the result, and then calculating a sine (cosine) transform of the appropriate type, as determined by the polarity. Inspired by this algorithm for the continuous case, we can similarly represent the projection of a vector $\mathtt{f}=(\mathtt{f}_1, \ldots, \mathtt{f}_n)$ in terms of LST (LCT) basis functions that are completely parallel to those given at the start of this section. The *only* difference is that x now runs through a discrete set of integers, $\{1, \ldots, n\}$, rather than an interval of the real line. The resulting transformations have all the orthogonality and symmetry/antisymmetry properties that were present in the continuous case, but in addition are susceptible to easy computational implementation.

Specifically, after applying the tool `WINDOW` to the discrete vector `f` and applying `FOLD` to the result, we are left with the discrete analogue of $S(\cdot)$ i.e., `fldy`. The discrete local trigonometric coefficients of the projection of `f` are then calculated from `fldy` as

$$\texttt{dltcoefs} = \begin{cases} \texttt{DST\_I}(\texttt{fldy}), & \text{if } (-,-) \\ \texttt{DST\_III}(\texttt{fldy}), & \text{if } (-,+) \\ \texttt{DCT\_I}(\texttt{fldy}), & \text{if } (+,+) \\ \texttt{DCT\_III}(\texttt{fldy}), & \text{if } (+,-) \end{cases}$$

Precise definitions of the discrete trigonometric transforms may be found in Rao & Yip [45]. These authors also explain how the discrete sine and cosine transforms of types II and III may be computed using the fast Hartley transform, while those of type I may be computed using the fast Fourier transform.

3.2.5 Inverting Local Trigonometric Transforms

Having represented a projection in terms of local trigonometric bases, we can similarly recover that projection by reversing the process, since the sine (cosine) transforms, as well as the folding operation, are invertible. In particular, given the discrete local trigonometric coefficients, `dltcoefs`, from the above subsection, the folded, windowed version of `f` is calculated as

$$\texttt{fldy} = \begin{cases} \texttt{DST\_I}(\texttt{dltcoefs}), & \text{if } (-,-) \\ \texttt{DST\_II}(\texttt{dltcoefs}), & \text{if } (-,+) \\ \texttt{DCT\_I}(\texttt{dltcoefs}), & \text{if } (+,+) \\ \texttt{DCT\_II}(\texttt{dltcoefs}), & \text{if } (+,-). \end{cases}$$

Note that the DST-I and DCT-I are their own inverses (unitary transforms), while the DST-II and DCT-II are the inverses of the DST-III and DCT-III, respectively.

We next unfold the vector `fldy` using a tool that simply reverses the operation of `FOLD`, which we now define.

define `y` $\equiv$ `UNFOLD( fldy, [a,b], (e,e'), pol )`

{ *Unfold Folded, Windowed Vector* }

Variables of function `UNFOLD` $\equiv$

`fldy` := *vector of length* $card([\mathtt{a},\mathtt{b}] \cap \boldsymbol{Z})$

`[a,b],(e,e')` := *defined as in* `WINDOW`

`pol` := *(string) polarity of* `[a,b]`

`y` := *vector of length* $m = card([\mathtt{a}-\mathtt{e}, \mathtt{b}+\mathtt{e}'] \cap \boldsymbol{Z})$

The variables of `UNFOLD` are the same as those of `FOLD`, except that the places of `y` and `fldy` are switched. As the resulting vector `y` is the discrete analogue of $S(\cdot)$, we see by consulting Equation 3.3 that the projection of `f` onto I is now obtained by applying `WINDOW` to `y`.

3.2.6 Extension

In the next section, we will be interested in representing projections of the Fourier transform of a real function f in terms of local trigonometric coefficients. It will be in establishing links between these coefficients and the Meyer wavelet coefficients that we arrive at our Meyer wavelet transform algorithms.

If we write the Fourier transform of $f \in L^2(\mathbb{R})$ as $\hat{f} = f_1 + \imath f_2$, and define $-I \equiv [-\beta, -\alpha]$, note that projections of f_1 and f_2 onto $-I$ may be obtained directly from their projections onto I using the relations

$$(P_{-I}f_1)(x) = (P_I f_1)(-x) \qquad \forall x \in (-\infty, 0)$$

and

$$(P_{-I}f_2)(x) = -(P_I f_2)(-x) \qquad \forall x \in (-\infty, 0).$$

These relations follow from the fact that for real functions f, the real and imaginary parts of $\hat{f}$ will be even and odd, respectively. Note that the window associated with the projections onto $-I$ will be the symmetric reflection about zero of the window associated with the projections onto I.

These results motivate us to define the following tool for extending the projection of a vector `f` onto the discrete analogue of the interval I to that of the union of intervals $-I \cup I$.

---

define `extproj` $\equiv$ `EXTEND( proj, [a,b], (e,e'), type )`

{*Extend Projection to Negative Values* }

Variables of function `EXTEND` $\equiv$

`proj` := *projection onto [a,b]. Vector of length card*$([\mathsf{a}-\mathsf{e}, \mathsf{b}+\mathsf{e}'] \cap \boldsymbol{Z})$.

`[a,b], (e,e')` := *as defined in* `WINDOW`

`type` := *(string) type of extension, 'even' or 'odd'*

`extproj` := *extended projection. Vector of length* $2\times$ *length*(`proj`)

3.3 Coefficient Relationships and Identities

In this section, we derive certain relationships between the 'mother' and 'father' Meyer wavelet coefficients and particular local sine and cosine coefficients. We justify these relationships using the projection, folding, and polarity concepts of Section 3.2.

3.3.1 Mother Meyer Wavelet

We begin with the 'mother' Meyer wavelet. This wavelet, or appropriate dilations thereof, underlies each of the detail levels of coefficients. What follows is a derivation of the relationship between the 'mother' Meyer wavelet coefficients of a given level, and sets of particular local sine and cosine coefficients. As our goal is to use these relationships in creating our forward and inverse algorithms, we begin with the case of functions in $L^2(\mathbb{R})$, continue with functions in $L^2([0,1])$, and finish with functions on the discrete circle.

Functions in $L^2(\mathbb{R})$

For a function $f \in L^2(\mathbb{R})$, we let $\hat{f}(\xi) = \int_{-\infty}^{\infty} f(x)e^{-i2\pi x\xi}dx$ denote the Fourier transform of f at ξ. We define the 'mother' Meyer wavelet, ψ, in the frequency domain as

$$\hat{\psi}(\xi) = \begin{cases} e^{-i\pi\xi}\sin(\frac{\pi}{2}\nu(3|\xi|-1)), & \text{if } 1/3 < |\xi| \leq 2/3 \\ e^{-i\pi\xi}\cos(\frac{\pi}{2}\nu(\frac{3}{2}|\xi|-1)), & \text{if } 2/3 < |\xi| \leq 4/3 \\ 0, & \text{otherwise} \end{cases}$$

where $\nu(\cdot)$ is a smooth function such that

$$\nu(x) = \begin{cases} 0, & \text{if } x \leq 0 \\ 1, & \text{if } x > 1 \end{cases}$$

and $\nu(x) + \nu(1-x) = 1$. Figure 3.1 shows a graph of ψ and $|\hat{\psi}|$ when ν is chosen to be $\nu(x) = x^4(35 - 84x + 70x^2 - 20x^3)$ on $[0,1]$. This wavelet is a $C^{\infty}(\mathbb{R})$ function, has compact support in the frequency domain, and has a Fourier transform that is a member of $C^{4-\delta}$. In general, the regularity of $\hat{\psi}$ will be same as that of ν.

Letting $b_{I_0}(\xi) \equiv |\hat{\psi}(\xi)|$, we can write

$$\begin{aligned} \hat{\psi}(\xi) &= b_{I_0}(\xi)e^{-i\pi\xi} \\ &= b_{I_0}(\xi)\{\cos(\pi\xi) - i\sin(\pi\xi)\}. \end{aligned}$$

In the language of Section 3.2, the function $b_{I_0}(\cdot)$ is a bell function over the interval $I_0 = [\frac{1}{2}, 1]$ (as well as its symmetric reflection, $[-1, -\frac{1}{2}]$). The bell b_{I_0} is only supported on I_0 in a 'moral' sense, in that it is allowed to overlap the interval by $\epsilon_0 = \frac{1}{6}$ to the left and $\epsilon_0' = 2\epsilon_0 = \frac{1}{3}$ to the right. More generally, defining the wavelet at resolution

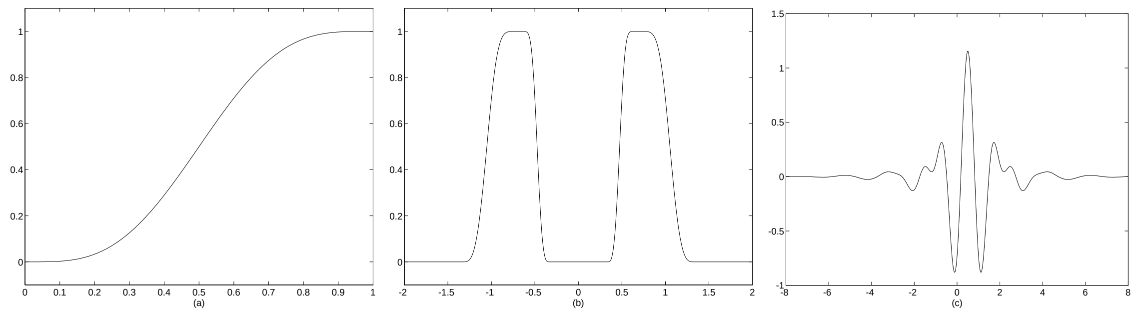


Figure 3.1: (a) The auxiliary function $\nu(x) = x^4(35 - 84x + 70x^2 - 20x^3)$. (b) The modulus of the associated 'mother' Meyer wavelet, $|\hat{\psi}|$. (c) The 'mother' Meyer wavelet, ψ, in the time domain.

level j and position k as

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

and $b_{I_j}(\cdot) \equiv b_{I_0}(2^{-j}\cdot)$ as the corresponding bell function over $I_j = [2^{j-1}, 2^j]$, with $\epsilon_j = 2^j/6$ and $\epsilon'_j = 2^j/3$, we can write

$$\begin{aligned}\hat{\psi}_{j,k}(\xi) &= 2^{-j/2} b_{I_j}(\xi)\{\cos(2\pi(k+\frac{1}{2})2^{-j}\xi) - i\sin(2\pi(k+\frac{1}{2})2^{-j}\xi)\} \\ &\equiv u_{j,k}(\xi) - iv_{j,k}(\xi). \end{aligned} \tag{3.6}$$

Note that the cosine and sine contributions of $u_{j,k}$ and $v_{j,k}$ have, respectively, odd/even and even/odd symmetries about the endpoints of I_j. This suggests, as has been shown explicitly by Meyer [36] and AWW [1], that the Meyer wavelets can be represented in the frequency domain through their real and imaginary parts, by using local sine and cosine bases of $(-,+)$ and $(+,-)$ polarities (LST-III and LCT-III in Section 3.2). It is in examining this relationship in greater detail, as well as translating it to the case of periodic and discrete functions, that we derive the relations necessary for our algorithms.

Note that the wavelet coefficient at level 0 and position $k \in \mathbf{Z}$ may be expressed as

$$\alpha_{0,k} = \langle f, \psi_{0,k}\rangle = \int_{\mathbb{R}} \Re\hat{f}(\xi) u_{0,k}(\xi) d\xi - \int_{\mathbb{R}} \Im\hat{f}(\xi) v_{0,k}(\xi) d\xi.$$

It follows directly that for the collection of wavelet coefficients at level 0 we have,

$$\frac{\alpha_{0,k} + \alpha_{0,-(k+1)}}{2} = \int_{\mathbb{R}} \Re\hat{f}(\xi) u_{0,k}(\xi) d\xi \tag{3.7}$$

and

$$\frac{\alpha_{0,k} - \alpha_{0,-(k+1)}}{2} = -\int_{\mathbb{R}} \Im\hat{f}(\xi) v_{0,k}(\xi) d\xi. \tag{3.8}$$

The right hand sides of (3.7) and (3.8) appear to be 'almost' local *cosine* and *sine* coefficients, respectively. Actually, local *sine* and *cosine* coefficients may be obtained if we modulate the left-hand sides by $2(-1)^{k+1}$. Specifically,

$$\begin{aligned} c_{0,k} &= \frac{(-1)^{k+1}}{2}\{\alpha_{0,k} + \alpha_{0,-(k+1)}\} \\ &= \int_0^\infty \Re\hat{f}(\xi)\sqrt{\frac{2}{|I_0|}} b_{I_0}(\xi) \sin[(k+1/2)\frac{\pi}{|I_0|}(\xi - \frac{1}{2})] d\xi \end{aligned}$$

and

$$\begin{aligned} d_{0,k} &= \frac{(-1)^{k+1}}{2}\{\alpha_{0,k} - \alpha_{0,-(k+1)}\} \\ &= \int_0^\infty \Im\hat{f}(\xi)\sqrt{\frac{2}{|I_0|}} b_{I_0}(\xi) \cos[(k+1/2)\frac{\pi}{|I_0|}(\xi - \frac{1}{2})] d\xi, \end{aligned}$$

are local sine and cosine coefficients corresponding to the projections of the positive real and imaginary parts of $\hat{f}$ onto the interval I_0. Writing these projections as $P_{I_0}(\Re\hat{f})$ and $P_{I_0}(\Im\hat{f})$, they may be recovered from the level 0 wavelet coefficients, $\{\alpha_{0,k}\}_{k\in\mathbf{Z}}$, as

$$P_{I_0}(\Re\hat{f})(\xi) = \sqrt{\frac{2}{|I_0|}} b_{I_0}(\xi) \sum_{k=0}^{\infty} c_{0,k} \sin[(k+1/2)\frac{\pi}{|I_0|}(\xi - \frac{1}{2})]$$

and

$$P_{I_0}(\Im\hat{f})(\xi) = \sqrt{\frac{2}{|I_0|}} b_{I_0}(\xi) \sum_{k=0}^{\infty} d_{0,k} \cos[(k+1/2)\frac{\pi}{|I_0|}(\xi - \frac{1}{2})].$$

It can be shown similarly that the projections $P_{I_j}(\Re\hat{f})$ and $P_{I_j}(\Im\hat{f})$ onto the interval I_j are recoverable from the level j wavelet coefficients, $\{\alpha_{j,k}\}_{k\in Z}$. Since the projections are orthogonal i.e.,

$$P_{I_j} P_{I_{j'}} = 0 \qquad \forall j \neq j',$$

and adjacent intervals have opposite polarities at the shared endpoint, we obtain the projections of $\Re\hat{f}$ and $\Im\hat{f}$ onto the interval $[0, 2^M]$ by summing projections as

$$P_{[0,2^M]} = \sum_{j=-\infty}^{M} P_{I_j}.$$

Note that care should be taken to recall that we are in $L^2(I\!R)$, and so equality of the two sides above must be interpreted in that sense.

We recover all of $\hat{f}$ (and hence f itself) as

$$\hat{f} = \lim_{M\to\infty} \{ \; P_{[-2^M,2^M]}(\Re\hat{f}) + \imath P_{[-2^M,2^M]}(\Im\hat{f}) \; \},$$

where

$$P_{[-2^M,2^M]}(\Re\hat{f})(\xi) = \begin{cases} P_{[0,2^M]}(\Re\hat{f})(\xi), & \text{if } \xi \geq 0 \\ P_{[0,2^M]}(\Re\hat{f})(-\xi), & \text{if } \xi < 0 \end{cases}$$

and

$$P_{[-2^M,2^M]}(\Im\hat{f})(\xi) = \begin{cases} P_{[0,2^M]}(\Im\hat{f})(\xi), & \text{if } \xi \geq 0 \\ -P_{[0,2^M]}(\Im\hat{f})(-\xi), & \text{if } \xi < 0. \end{cases}$$

Functions in $L^2([0,1])$

Let f now be a periodic function in $L^2([0,1])$, and define the periodized Meyer wavelet on $[0,1]$ to be

$$\psi^{\circ}_{j,k}(x) \equiv \sum_{l} \psi_{j,k}(x+l),$$

for $j \in \mathbf{Z}^+$ and $k = 0, 1, \ldots, 2^j-1$. Figure 3.2 shows some examples of these periodized wavelets. Note that the collection $\{\psi^{\circ}_{j,k}\}$ forms a complete orthonormal system. If we now let $\hat{f}(l) = \int_0^1 f(x)e^{-i2\pi xl}dx$, it follows that

$$\begin{aligned} \hat{\psi}^{\circ}_{j,k}(l) &= \int_0^1 \psi^{\circ}_{j,k}(x)e^{-i2\pi lx}dx \\ &= \int_0^1 \{\sum_{l'} \psi_{j,k}(x+l')\}e^{-i2\pi lx}dx \\ &= \int_{-\infty}^{\infty} \psi_{j,k}(\xi)e^{-i2\pi l\xi}d\xi \\ &= \hat{\psi}_{j,k}(l). \end{aligned}$$

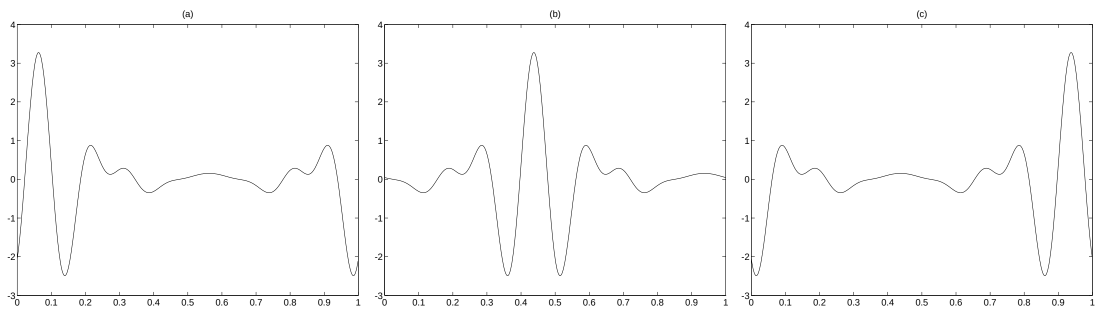


Figure 3.2: Figures (a)-(c) show the periodized, 'mother' Meyer wavlets $\psi^\circ_{3,0}$, $\psi^\circ_{3,3}$, and $\psi^\circ_{3,7}$.

So $\hat{\psi}^\circ_{j,k}(l) = u_{j,k}(l) - i v_{j,k}(l)$, similar to the case in $L^2(I\!\!R)$, and hence

$$\begin{aligned}\alpha^\circ_{j,k} &= \int_0^1 f(x)\psi^\circ_{j,k}(x)dx \\ &= \sum_{l\in\mathbf{Z}} \Re\hat{f}(l)u_{j,k}(l) - \sum_{l\in\mathbf{Z}} \Im\hat{f}(l)v_{j,k}(l).\end{aligned}$$

In order to obtain relations analogous to (3.7) and (3.8), we first define the convention

$$\ominus(k+1) \equiv 2^j - k - 1,$$

which accounts for the fact that k only runs over a finite number of values in the periodic case. With this convention, we may write

$$\frac{\alpha^\circ_{j,k} + \alpha^\circ_{j,\ominus(k+1)}}{2} = \sum_{l\in\mathbf{Z}} \Re\hat{f}(l)u_{j,k}(l)$$

and

$$\frac{\alpha^\circ_{j,k} - \alpha^\circ_{j,\ominus(k+1)}}{2} = -\sum_{l\in\mathbf{Z}} \Im\hat{f}(l)v_{j,k}(l).$$

Modulating again by $2(-1)^{k+1}$, from the wavelet coefficients $\{\alpha^\circ_{j,k}\}_{k=0}^{2^j-1}$ we get the local sine and cosine coefficients

$$\{c^\circ_{j,k}\}_{k=0}^{2^{j-1}-1} \quad \text{and} \quad \{d^\circ_{j,k}\}_{k=0}^{2^{j-1}-1}.$$

Note that at each level there are only half as many local sine (cosine) coefficients as there are wavelet coefficients, due to the circular fashion in which we have defined $\ominus(k+1)$.

Let I_j° be defined as the *integer* points belonging to the I_j of the previous subsection; but now $P_{I_j^\circ}$ is understood to mean projection onto the set I_j°. Using the local sine and cosine coefficients, we can recover the projections $P_{I_j^\circ}(\Re\hat{f})$ and $P_{I_j^\circ}(\Im\hat{f})$ as

$$P_{I_j^\circ}(\Re\hat{f})(l) = \sqrt{\frac{2}{|I_j^\circ|}} b_{I_j^\circ}(l) \sum_{k=0}^{2^{j-1}-1} c_{j,k}^\circ \sin[(k+1/2)\frac{\pi}{|I_j^\circ|}(l-2^{j-1})]$$

and

$$P_{I_j^\circ}(\Im\hat{f})(l) = \sqrt{\frac{2}{|I_j^\circ|}} b_{I_j^\circ}(l) \sum_{k=0}^{2^{j-1}-1} d_{j,k}^\circ \cos[(k+1/2)\frac{\pi}{|I_j^\circ|}(l-2^{j-1})],$$

for $l \geq 0$. We extend by symmetry and antisymmetry, respectively, to obtain projections on I_j° and its reflection about 0. Finally, in a manner similar to above, we can recover $\hat{f}$ and hence f itself by summing the projections for all $j \in \mathbf{Z}$.

The fact that the operation $P_{I_j^\circ}$ is indeed a projection, as defined in the previous section, with all of the related properties, can be shown rigorously through a set of arguments revolving around the Poisson summation formula. Such an approach involves *unwrapping* the periodic functions on $[0,1]$ to the real line, and proving the existence of a typical commutative diagram relating the projections in the two settings.

Functions on the Discrete Circle

Suppose now that we observe data $\{f(t/n)\}_{t=0}^{n-1}$, where $n = 2^J$ for some $J \geq 2$. Define the discrete Fourier transform of f to be

$$\hat{f}^D(m) = \frac{1}{n}\sum_{t=0}^{n-1} f(t/n)e^{-i2\pi tm/n}.$$

Then for $m \geq 0$ and $j \leq J-1$,

$$\hat{\psi}_{j,k}^{\circ D}(m) = \frac{1}{n}\sum_{t=0}^{n-1} \psi_{j,k}^\circ(\frac{t}{n})e^{-i2\pi tm/n}$$

$$
\begin{aligned}
&= \frac{1}{n}\sum_{t=0}^{n-1}\{\sum_{l\in\mathbb{Z}} \hat{\psi}^{\circ}_{j,k}(l)e^{i2\pi lt/n}\}e^{-i2\pi tm/n} \\
&= \frac{1}{n}\sum_{t=0}^{n-1}\{\sum_{l\in\mathbb{Z}} \hat{\psi}_{j,k}(l)e^{i2\pi lt/n}\}e^{-i2\pi tm/n}.
\end{aligned}
$$

The fact that the support of $\hat{\psi}_{j,k}$ is $(-\frac{4}{3}2^j, -\frac{2^j}{3}) \cup (\frac{2^j}{3}, \frac{4}{3}2^j)$ implies that $\hat{\psi}_{j,k}(l) = 0$ for $|l| \geq \lceil \frac{n}{3} \rceil$ and $j = 0, \ldots, J-2$. So we write

$$
\begin{aligned}
\hat{\psi}^{\circ D}_{j,k}(m) &= \frac{1}{n}\sum_{t=0}^{n-1}\{\sum_{l=-n/2+1}^{n/2} \hat{\psi}_{j,k}(l)e^{i2\pi lt/n}\}e^{-i2\pi tm/n} \\
&= \frac{1}{n}\sum_{l=-n/2+1}^{n/2} \hat{\psi}_{j,k}(l)\{\sum_{t=0}^{n-1} e^{i2\pi\frac{t}{n}(l-m)}\} \\
&= \hat{\psi}_{j,k}(m),
\end{aligned}
$$

for $0 \leq m \leq n/2$. Extending to the second half of the discrete circle by Hermitian symmetry gives us the result that

$$
\hat{\psi}^{\circ D}_{j,k}(m) = \hat{\psi}_{j,k}(m) = \hat{\psi}^{\circ}_{j,k}(m)
$$

for $-n/2 < m \leq n/2$ and $j = 0, \ldots, J-2$.

This equivalence of the Fourier transform of ψ in all three settings has important consequences for the development of algorithms. If we assume that our sample $\{f(t/n)\}_{t=0}^{n-1}$ is from a trigonometric polynomial f of degree at most n, then we see that

$$
\begin{aligned}
\alpha^{\circ D}_{j,k} &\equiv \sum_{0}^{n-1} \hat{f}^D(l)\overline{\hat{\psi}^{\circ D}_{j,k}(l)} \\
&= \sum_{-n/2+1}^{n/2} \hat{f}(l)\overline{\hat{\psi}^{\circ}_{j,k}(l)} \\
&= \sum_{l\in\mathbb{Z}} \hat{f}(l)\overline{\hat{\psi}^{\circ}_{j,k}(l)} \\
&= \alpha^{\circ}_{j,k},
\end{aligned}
$$

for $j = 0, 1, \ldots, J-2$ and $k = 0, 1, \ldots, 2^j - 1$. This means that the results of the previous section, relating levels of wavelet coefficients to appropriate projections, carry over directly to the case of functions on the discrete circle. Furthermore, the equivalence of $\hat{\psi}^{\circ D}_{j,k}$ and $\hat{\psi}_{j,k}$ on $\{-\frac{n}{2}+1, \ldots, \frac{n}{2}\}$ implies that no special modifications of $\hat{\psi}_{j,k}$ (whose definition is known) are necessary for coding computer algorithms for discrete data, as discussed in Section 3.7.

3.3.2 Father Meyer Wavelet

Recall that in practice, wavelet algorithms are designed with a coarse level of approximation, followed by a set of increasingly finer details. For the coarse level of coefficients, it is not the 'mother' wavelet itself that is used, but rather the 'father' wavelet or scale function. Below we derive relations for the 'father' Meyer wavelet similar to those of the 'mother' wavelet. However, a number of subtle differences appear as we proceed, due to the difference in the nature of the bell functions involved.

Functions in $L^2(\mathbb{R})$

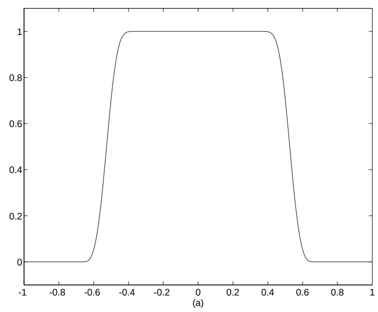
The 'father' Meyer wavelet is written in the frequency domain as

$$\hat{\phi}(\xi) = \begin{cases} 1, & \text{if } |\xi| \leq 1/3 \\ \cos(\frac{\pi}{2}\nu(3|\xi|-1)), & \text{if } 1/3 < |\xi| \leq 2/3 \\ 0, & \text{otherwise} \end{cases}$$

and it may be checked by consulting the list of properties in AWW that $\hat{\phi}$ also qualifies as a bell function, this time on the interval $\tilde{I}_0 = [-\frac{1}{2}, \frac{1}{2}]$. Figure 3.3 shows the 'father' Meyer wavelet in both the frequency and time domains, using the same auxiliary function ν as in Figure 3.1.

Similar to Equation (3.6) we can write for $\phi_{j,k}(x) = 2^{j/2}\phi(2^j x - k)$,

$$\hat{\phi}_{j,k}(\xi) \quad = \quad 2^{-j/2}\hat{\phi}(2^{-j}\xi)\{\cos(2\pi k 2^{-j}\xi) - i\sin(2\pi k 2^{-j}\xi)\}$$



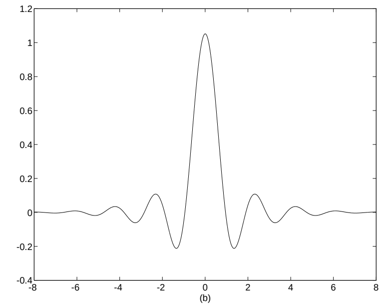


Figure 3.3: (a) The modulus of the 'father' Meyer wavelet, $|\hat{\phi}|$. (b) The 'father' Meyer wavelet, ϕ, in the time domain.

$$\equiv \quad \tilde{u}_{j,k}(\xi) - i\tilde{v}_{j,k}(\xi)$$

In this case the cosine and sine contributions to $\tilde{u}_{j,k}(\cdot)$ and $\tilde{v}_{j,k}(\cdot)$ have even/even and odd/odd symmetries about the endpoints of $\tilde{I}_j = [-2^{j-1}, 2^{j-1}]$, indicating $(+,+)$ and $(-,-)$ polarities, respectively. This suggests that the real and imaginary parts of a given level of 'father' wavelets (in the frequency domain) may be expressed in terms of the local cosine and sine systems LCT-I and LST-I of Section 3.2.

As in the above subsection, we spell out this relationship in terms of the corresponding coefficients. Expressing the coarse level wavelet coefficients as

$$\beta_{j,k} = \langle f, \phi_{j,k} \rangle = \int_{I\!R} \Re \hat{f}(\xi)\tilde{u}_{j,k}(\xi)d\xi - \int_{I\!R} \Im \hat{f}(\xi)\tilde{v}_{j,k}(\xi)d\xi,$$

for $f \in L^2(I\!R)$, local cosine and sine coefficients may be constructed from them, corresponding to the real and imaginary parts of $\hat{f}$, respectively. Letting $\tilde{I}_j = [-2^{j-1}, 2^{j-1}]$ and $b_{\tilde{I}_j}(\xi) = \hat{\phi}(2^{-j}\xi)$, we write the cosine coefficients, $\{\tilde{c}_{j,k}\}_{k\in\mathbb{Z}}$, as

$$\begin{aligned}
\tilde{c}_{j,0} &= \beta_{j,0} = \int_{I\!R} \Re \hat{f}(\xi) \frac{1}{\sqrt{|\tilde{I}_j|}} b_{\tilde{I}_j}(\xi)d\xi \\
\tilde{c}_{j,2k} &= \frac{(-1)^k}{\sqrt{2}}\{\beta_{j,k} + \beta_{j,-k}\} \\
&= \int_{I\!R} \Re \hat{f}(\xi) \sqrt{\frac{2}{|\tilde{I}_j|}} b_{\tilde{I}_j}(\xi) \cos[k\frac{\pi}{|\tilde{I}_j|}(\xi + 2^{j-1})]d\xi \\
\tilde{c}_{j,2k+1} &= 0,
\end{aligned}$$

and the sine coefficients as

$$\begin{aligned}
\tilde{d}_{j,0} &= 0 \\
\tilde{d}_{j,2k} &= \frac{(-1)^{k+1}}{\sqrt{2}}\{\beta_{j,k} - \beta_{j,-k}\} \\
&= \int_{I\!R} \Im\hat{f}(\xi)\sqrt{\frac{2}{|\tilde{I}_j|}}b_{\tilde{I}_j}(\xi)\sin[k\frac{\pi}{|\tilde{I}_j|}(\xi + 2^{j-1})]d\xi \\
\tilde{d}_{j,2k+1} &= 0,
\end{aligned}$$

for $k \geq 1$.

Note that although the process for obtaining these coefficients is similar to that of the 'mother' wavelets, the results differ in a number of ways. The key difference is that the projection of $\Re\hat{f}$ will now correspond to local *cosine* coefficients while that of $\Im\hat{f}$ will correspond to local *sine* coefficients. Also, the coefficients $\tilde{c}_{j,k}$ and $\tilde{d}_{j,k}$ are zero for all odd values of k. At even values of k they may be non-zero, and note that they are formed from coarse wavelet coefficients at locations k and $-k$, as opposed to k and $-(k+1)$ with the 'mother' wavelets. Finally, whereas in dealing with the 'mother' wavelet coefficients our aim was to recover a projection on $[2^{j-1}, 2^j]$ and extend it to $[-2^j, -2^{j-1}]$ by symmetry arguments, here the only interval of interest is $[-2^{j-1}, 2^{j-1}]$, to which our projection is already tailored.

The projections of $\Re\hat{f}$ and $\Im\hat{f}$ onto $\tilde{I}_j$ may be recovered by forming

$$P_{\tilde{I}_j}(\Re\hat{f})(\xi) = \frac{\tilde{c}_{j,0}}{\sqrt{|\tilde{I}_j|}}b_{\tilde{I}_j}(\xi) + \sqrt{\frac{2}{|\tilde{I}_j|}}b_{\tilde{I}_j}(\xi)\sum_{k=1}^{\infty}\tilde{c}_{j,k}\cos[k\frac{\pi}{|\tilde{I}_j|}(\xi + 2^{j-1})]$$

and

$$P_{\tilde{I}_j}(\Im\hat{f})(\xi) = \sqrt{\frac{2}{|\tilde{I}_j|}}b_{\tilde{I}_j}(\xi)\sum_{k=1}^{\infty}\tilde{d}_{j,k}\sin[k\frac{\pi}{|\tilde{I}_j|}(\xi + 2^{j-1})]$$

for all $\xi \in I\!R$.

Functions in $L^2([0,1])$

As we did with the 'mother' wavelets, begin by defining the periodized 'father' Meyer wavelet on $[0,1]$ as

$$\phi^\circ_{j,k}(x) = \sum_{l\in\mathbf{Z}} \phi_{j,k}(x+l),$$

for $j \in \mathbf{Z}^+$ and $k = 0, 1, \ldots, 2^j - 1$. Figure 3.4 shows some examples. We again have

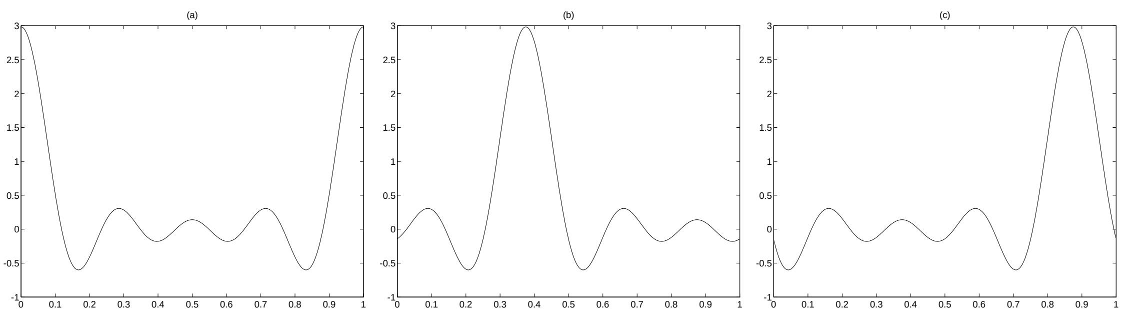


Figure 3.4: Figures (a)-(c) show the periodized, 'father' Meyer wavlets $\phi^\circ_{3,0}$, $\phi^\circ_{3,3}$, and $\phi^\circ_{3,7}$.

the result that

$$\hat{\phi}^\circ_{j,k}(l) = \hat{\phi}_{j,k}(l) \qquad \forall l \in \mathbf{Z},$$

and because the range of k is again limited, it is necessary to employ the circular convention $\ominus k \equiv 2^j - k$.

With these definitions and results in hand, it may be shown that a set of local cosine and sine coefficients,

$$\{\tilde{c}^\circ_{j,k}\}_{k=0}^{2^{j+1}-1} \quad \text{and} \quad \{\tilde{d}^\circ_{j,k}\}_{k=0}^{2^{j+1}-1},$$

may be formed from the coarse coefficients

$$\beta^\circ_{j,k} = \int_0^1 f(x)\phi^\circ_{j,k}(x)dx = \sum_{l\in\mathbf{Z}} \Re\hat{f}(l)\tilde{u}_{j,k}(l) - \sum_{l\in\mathbf{Z}} \Im\hat{f}(l)\tilde{v}_{j,k}(l).$$

Note that there are a total of $2^{j+1}+1$ coefficients of each type, as opposed to 2^{j-1} in the case of the 'mother' Meyer wavelet. However, half of these are zero, similar to the $L^2(\mathbb{R})$ case.

The projections of $\Re\hat{f}$ and $\Im\hat{f}$ onto $\tilde{I}_j^\circ$ (the *integer* points belonging to $\tilde{I}_j$) may be recovered from the corresponding local trigonometric coefficients as

$$P_{\tilde{I}_j^\circ}(\Re\hat{f})(l) = \frac{\tilde{c}_{j,0}^\circ}{\sqrt{|\tilde{I}_j^\circ|}} b_{\tilde{I}_j^\circ}(l) + \sqrt{\frac{2}{|\tilde{I}_j^\circ|}} b_{\tilde{I}_j^\circ}(l) \sum_{k=1}^{2^{j+1}-1} \tilde{c}_{j,k}^\circ \cos[k\frac{\pi}{|\tilde{I}_j^\circ|}(l+2^{j-1})]$$

and

$$P_{\tilde{I}_j^\circ}(\Im\hat{f})(l) = \sqrt{\frac{2}{|\tilde{I}_j^\circ|}} b_{\tilde{I}_j^\circ}(l) \sum_{k=1}^{2^{j+1}-1} \tilde{d}_{j,k}^\circ \sin[k\frac{\pi}{|\tilde{I}_j^\circ|}(l+2^{j-1})]$$

for *all* integers $l \in \mathbf{Z}$.

Functions on the Discrete Circle

Define our sample $\{f(t/n)\}_{t=0}^{n-1}$ as above. Since the support of $\hat{\phi}_{j,k}$ is $[-2^{j+1}/3, 2^{j+1}/3]$, it may be shown similar to the case of the 'mother' wavelets that

$$\hat{\phi}_{j,k}^{\circ D}(m) = \hat{\phi}_{j,k}^\circ(m) = \hat{\phi}_{j,k}(m)$$

for $0 \le m \le n/2$ and $j = 0, 1, \ldots, J-2$. Extension by Hermitian symmetry gives us the same result on the entire discrete circle. This equivalence then implies that

$$\beta_{j,k}^{\circ D} = \beta_{j,k}^\circ,$$

for $j = 0, 1, \ldots, J-2$ and $k = 0, 1, \ldots, 2^j - 1$, in the case where f is a trigonometric polynomial of degree at most n. This suggests that we may incorporate directly the formulas relating coarse level coefficients of periodic functions and their projections onto $\tilde{I}_j^\circ$ into an algorithm for discrete data.

In particular, having such a result for both the 'father' and 'mother' wavelets implies that we may recover $\hat{f}^D$ from its discrete wavelet transform as

$$\hat{f}^D = \{\ P_{\tilde{I}_C^\circ}(\Re\hat{f}^D) + \sum_{j=C}^{J-1} P_{I_j^\circ}(\Re\hat{f}^D)\ \} + \imath\{\ P_{\tilde{I}_C^\circ}(\Im\hat{f}^D) + \sum_{j=C}^{J-1} P_{I_j^\circ}(\Im\hat{f}^D)\ \}.$$

3.4 Forward Meyer Wavelet Transform

In this section we present detailed algorithms for Step 3 of the FMWT algorithm. Steps 1 and 2 involve taking the fast Fourier transform of a signal vector and separating the result into real and imaginary parts. Step 3 forms the bulk of the algorithm, and consists of a series of windowing, folding, and discrete trigonometric operations that correspond to computing local trigonometric transforms of a projection of the signal onto a particular frequency band. The resulting local trigonometric coefficients are combined to yield the wavelet coefficients for a particular resolution level.

As mentioned in Section 3.3, the details of these operations vary depending on the level. Specifically, the coarse level and the detail levels will use different windows in projecting the data, corresponding to the special wavelets that underlie those levels. In addition, a truncated version of the detail window is used for the finest level of detail coefficients, due to the finiteness of the data. Each of these windows, depending on whether they window the real or imaginary parts of the signal, will have a particular polarity. This polarity dictates the form of the local trigonometric basis used to represent the projection. See Section 3.3 for more details.

This section is broken down into three subsections, corresponding to the coarse, detail, and fine-detail levels of coefficients. Within each subsection, we first present the stages of Step 3 in algorithmic form, and then proceed to discuss each stage in detail.

3.4.1 Coarse Level Coefficients

We calculate the coarse level wavelet coefficients of f as a combination of local trigonometric coefficients. These latter coefficients are those associated with a pair of local trigonometric transforms of two smooth projections of $\hat{f}$ onto the frequency domain support (morally) of the level C 'father' Meyer wavelets. The window (or bell function) defining these projections is proportional to the modulus of these same wavelets. The *local* trigonometric coefficients are obtained by calculating *ordinary* trigonometric coefficients of a version of $\hat{f}$ that has been windowed and folded in an energy preserving fashion.

Coarse Level Coefficient Algorithm

3. For the coarse coefficients at level C,

   (a) Window both $\{\Re\hat{f}\}$ and $\{\Im\hat{f}\}$ by $b^D_{\tilde{I}^\circ_C}$, producing compactly supported sequences

   BR = WINDOW(Ref̂, Mod_Father_C, $[-2^{C-1}, 2^{C-1}]$, $(\frac{2^C}{6}, \frac{2^C}{6})$)

   BI = WINDOW(Imf̂, Mod_Father_C, $(-2^{C-1}, 2^{C-1})$, $(\frac{2^C}{6}, \frac{2^C}{6})$)

   (b) Fold ...

   (i) $\{b^D_{\tilde{I}^\circ_C}\Re\hat{f}\}$: FBR = FOLD(BR, $[-2^{C-1}, 2^{C-1}]$, $(\frac{2^C}{6}, \frac{2^C}{6})$, (+,+))

   (ii) $\{b^D_{\tilde{I}^\circ_C}\Im\hat{f}\}$: FBI = FOLD(BI, $(-2^{C-1}, 2^{C-1})$, $(\frac{2^C}{6}, \frac{2^C}{6})$, (-,-))

   (c) Compute ...

   (i) DCT-I of the folded, windowed real part of $\hat{f}$:

   CR = DCT_I(FBR)

   (ii) DST-I of the folded, windowed imaginary part of $\hat{f}$:

   CI = DST_I(FBI)

   (d) Combine the DCT-I and DST-I coefficients CR and CI

Window

The data are multiplied by the modulus of the (unscaled) 'father' Meyer wavelet, in the frequency domain, at resolution level C i.e.,

$$b_{\tilde{I}^\circ_C}(l) = |\hat{\phi}(2^C l)| \qquad l \in \mathbf{Z},$$

which as a function of $x \in \mathbb{R}$ is essentially supported on the interval $\tilde{I}_C \equiv [-2^{C-1}, 2^{C-1}]$. We use the notation $\tilde{I}_C^\circ \equiv \tilde{I}_C \cap \mathbf{Z}$ in naming the window, as opposed to $\tilde{I}_C$, to emphasize the point that we are actually working with discrete data and not functions on $\mathbb{R}$. An adjustment is made to $b_{\tilde{I}_C^\circ}$ at the endpoints of $\tilde{I}_C$, to account for the discreteness of the data in the folding of Step 3(b). Specifically, we use the window

$$b^D_{\tilde{I}_C^\circ}(l) = \begin{cases} b_{\tilde{I}_C^\circ}(l), & \text{if } l \neq \pm 2^{C-1} \\ 1, & \text{if } l = \pm 2^{C-1} \end{cases}$$

Fold

The windowing of Step 3(a) yields two vectors of length

$$2^C + 2\lfloor \frac{2^C}{6} \rfloor + 1,$$

corresponding to data at indices centered about the set $\tilde{I}_C \cap \mathbf{Z}$. We use folding to reduce these vectors to signals completely on the set $\tilde{I}_C \cap \mathbf{Z}$. Computing specific discrete trigonometric transforms in Step 3(c) of the results of these windowing and folding operations will yield the desired local trigonometric coefficients.

We fold the outer $\lfloor \frac{2^C}{6} \rfloor$ points (on both the left and right) of the vectors

$$\{b^D_{\tilde{I}_C^\circ} \Re \hat{f}\} \qquad \text{and} \qquad \{b^D_{\tilde{I}_C^\circ} \Im \hat{f}\}$$

about the points indexed by the endpoints $l = \pm 2^{C-1}$ of $\tilde{I}_C$. For the windowed real part, this folding induces symmetries about the endpoints, corresponding to a $(+,+)$ polarity. For the windowed imaginary part, antisymmetries are induced by first multiplying the subvectors to be folded by -1, resulting in a $(-,-)$ polarity. These choices of polarities is predetermined by the local cosine and sine basis functions with which one can represent the projections of the real and imaginary parts of $\hat{f}$, respectively, onto $\tilde{I}_C^\circ$.

Note that we fold about integer indices or *at* data points, which differs from similar algorithms (in a different context) of Wickerhauser [60] where the folding is done about half-integers or *between* data points. It is this folding *at* actual data points that necessitates the adjustment to our window at those points. Note too that the folded vector resulting from $\Im \hat{f}$ must be zero at the endpoints $\pm 2^{C-1}$ to satisfy the antisymmetry conditions.

Discrete Trigonometric Transforms

We use discrete cosine and sine transformations to compute local trigonometric coefficients from the windowed, folded data, so that in Step 3(d) we may combine them to obtain the coarse level wavelet coefficients.

From the windowed real part, we want to compute local cosine coefficients

$$\tilde{c}^{\circ}_{C,k} = \sum_{l=-2^{C-1}-\lfloor \frac{2^C}{6} \rfloor}^{2^{C-1}+\lfloor \frac{2^C}{6} \rfloor} \Re \hat{f}(l) \sqrt{\frac{2}{|\tilde{I}^{\circ}_C|}} b^D_{\tilde{I}^{\circ}_C}(l) \cos\Big[k \frac{\pi}{|\tilde{I}^{\circ}_C|}(l + 2^{C-1})\Big],$$

for $k = 0, 1, \ldots, 2^{C+1} - 1$. Letting $S^r_C(\cdot)$ represent the folded version of the windowed real part, and setting $m = l + 2^{C-1}$ and $N = |\tilde{I}^{\circ}_C| \equiv 2^C$, we may instead write this as

$$\tilde{c}^{\circ}_{C,k} = \sqrt{\frac{2}{N}} \sum_{m=0}^{N} S^r_C(m) \cos\Big[\frac{\pi k m}{N}\Big].$$

Up to normalization by $1/\sqrt{2}$ at the endpoints,this is simply the first discrete cosine transform (DCT-I) of $\{S^r_C(m)\}_{m=0}^{2^C}$, and can be computed easily using the FFT.

From the windowed imaginary part, we want to compute local sine coefficients,

$$\tilde{d}^{\circ}_{C,k} = \sum_{l=-2^{C-1}-\lfloor \frac{2^C}{6} \rfloor}^{2^{C-1}+\lfloor \frac{2^C}{6} \rfloor} \Im \hat{f}(l) \sqrt{\frac{2}{|\tilde{I}^{\circ}_C|}} b^D_{\tilde{I}^{\circ}_C}(l) \sin\Big[k \frac{\pi}{|\tilde{I}^{\circ}_C|}(l + 2^{C-1})\Big],$$

for $k = 1, \ldots, 2^{C+1} - 1$. Defining m and N as above, and letting $S^i_C(\cdot)$ represent the folded version of the windowed imaginary part of the data, we may write this as

$$\tilde{d}^{\circ}_{C,k} = \sqrt{\frac{2}{N}} \sum_{m=1}^{N-1} S^i_C(m) \sin\Big[\frac{\pi k m}{N}\Big],$$

which is the first discrete sine transform (DST-I) of $\{S^i_C(m)\}_{m=1}^{2^C-1}$, and can also be computed using the FFT.

Combining the Local Trigonometric Coefficients

The final step in computing the coarse level wavelet coefficients, $\{\beta^{\circ}_{C,k}\}_{k=0}^{2^C-1}$, is to combine the local sine and cosine coefficients following the relations derived in Section 3.3. Specifically,

$$\beta^{\circ}_{C,k} = \begin{cases} \tilde{c}^{\circ}_{C,0}, & \text{if } k = 0 \\ \frac{(-1)^k}{\sqrt{2}}(\tilde{c}^{\circ}_{C,2k} - \tilde{d}^{\circ}_{C,2k}), & \text{if } k = 1, \ldots, 2^{C-1} \\ \frac{(-1)^k}{\sqrt{2}}(\tilde{c}^{\circ}_{C,2^{C+1}-2k} + \tilde{d}^{\circ}_{C,2^{C+1}-2k}), & \text{if } k = 2^{C-1}+1, \ldots, 2^C - 1 \end{cases}$$

3.4.2 Detail Level Coefficients

We calculate the jth detail level wavelet coefficients of f as a combination of local trigonometric coefficients. Due to symmetry considerations, in this case the latter coefficients are associated with a pair of local trigonometric transforms of two smooth projections of the *positive* part of $\hat{f}$ onto the *positive* frequency domain support of the level j 'mother' Meyer wavelets. The window defining these projections is the modulus of these same wavelets. Again, we calculate the required *local* trigonometric coefficients by calculating *ordinary* trigonometric transforms of windowed, folded versions of $\hat{f}$.

Detail Level Coefficient Algorithm

3. For detail coefficient level $j = C, \ldots, J-2$,

   (a) Window both $\{\Re\hat{f}^+\}$ and $\{\Im\hat{f}^+\}$ by $b^D_{I^{\circ}_j}$, producing compactly supported sequences

   `BR = WINDOW( Ref̂⁺, Mod_Mother_j,` $(2^{j-1}, 2^j]$, $(\frac{2^j}{3}, \frac{2^{j+1}}{3})$ `)`

   `BI = WINDOW( Imf̂⁺, Mod_Mother_j,` $[2^{j-1}, 2^j)$, $(\frac{2^j}{3}, \frac{2^{j+1}}{3})$ `)`

   (b) Fold ...

   (i) $\{b^D_{I^{\circ}_j}\Re\hat{f}^+\}$: `FBR = FOLD( BR,` $(2^{j-1}, 2^j]$, $(\frac{2^j}{3}, \frac{2^{j+1}}{3})$, `(-,+) )`

(ii) $\{b^D_{I_j^\circ} \Im \hat{f}^+\}$: `FBI = FOLD( BI, [`$2^{j-1}, 2^j$`), (`$\frac{2^j}{3}, \frac{2^{j+1}}{3}$`), (+,-) )`

(c) Compute ...

(i) DST-III of the folded, windowed real part of $\hat{f}^+$:

`CR = DST_III( FBR )`

(ii) DCT-III of the folded, windowed imaginary part of $\hat{f}^+$:

`CI = DCT_III( FBI )`

(d) Combine the DST-III and DCT-III coefficients `CR` and `CI`

Window

The positive part of the data is multiplied by the modulus of the (unscaled) 'mother' Meyer wavelet, in the frequency domain, at resolution level j i.e.,

$$b_{I_j^\circ}(l) = |\hat{\psi}(2^j l)| \qquad l \in \mathbf{Z},$$

which as a function of $x \in I\!R$ is essentially supported on the interval $I_j \equiv [2^{j-1}, 2^j]$. An adjustment is made at the endpoints of I_j, to account for the discreteness of the data in the folding of Step 3(b). Specifically, we use the window

$$b^D_{I_j^\circ}(l) = \begin{cases} b_{I_j^\circ}(l), & \text{if } l \neq 2^{j-1} \quad \text{or} \quad 2^j \\ 1, & \text{if } l = 2^{j-1} \quad \text{or} \quad 2^j \end{cases}$$

Fold

The windowing of Step 3(a) yields two vectors of length 2^j, corresponding to data at indices centered about the set $I_j \cap \mathbf{Z}$. We use folding to reduce these vectors to signals completely on the set $I_j \cap \mathbf{Z}$. Computing specific discrete trigonometric transforms in Step 3(c) of the results of these windowing and folding operations will yield the desired local trigonometric coefficients.

We fold the leftmost $\lfloor \frac{2^j}{3} \rfloor$ points and the rightmost $2^j - \lfloor \frac{2^j}{3} \rfloor - 1$ points of the vectors

$$\{b^D_{I^\circ_j} \Re \hat{f}^+\} \qquad \text{and} \qquad \{b^D_{I^\circ_j} \Im \hat{f}^+\}$$

about the points indexed by the endpoints $l = 2^{j-1}$ and 2^j of I_j. For the windowed real part, the left subvector is multiplied by -1 prior to folding, while the right subvector is unchanged, resulting in a $(-,+)$ polarity. For the windowed imaginary part, the right subvector is multiplied by -1 and the left remains unchanged, resulting this time in a $(+,-)$ polarity. Similar to the coarse level algorithm, these polarities are predetermined by the local sine and cosine basis functions with which one can represent the projections of the real and imaginary parts of $\hat{f}$, respectively, onto I°_j.

Note that we again fold about integer indices or *at* data points, which necessitates the adjustments in the window as described above. Note too that, similar to the coarse level of coefficients, antisymmetry (i.e. a negative polarity) indicates that the folded vectors must be zero at 2^{j-1} in the case of the real part, and 2^j in the case of the imaginary part.

Discrete Trigonometric Transforms

We use discrete sine and cosine transformations to compute local trigonometric coefficients from the windowed, folded data, so that in Step 3(d) we may combine them to obtain the detail level wavelet coefficients.

From the windowed real part, we want to compute local sine coefficients

$$c^\circ_{j,k} = \sum_{l=2^{j-1}-\lfloor \frac{2^j}{3} \rfloor}^{3\cdot 2^{j-1}-\lfloor \frac{2^j}{3} \rfloor -1} \Re \hat{f}(l) \sqrt{\frac{2}{|I^\circ_j|}} b^D_{I^\circ_j}(l) \sin\Big[(k+1/2)\frac{\pi}{|I^\circ_j|}(l-2^{j-1})\Big],$$

for $k = 0, 1, \ldots, 2^j - 1$. Letting $S^r_j(\cdot)$ represent the folded version of the windowed real part, and setting $m = l - 2^{j-1}$ and $N = |I^\circ_j| \equiv 2^{j-1}$, we may instead write this as

$$c^\circ_{j,k} = \sqrt{\frac{2}{N}} \sum_{m=1}^{N} S^r_j(m) \sin\Big[(k+1/2)\frac{\pi m}{N}\Big].$$

Up to normalization by $1/\sqrt{2}$ at $m = N$,this is simply the third discrete sine transform (DST-III) of $\{S_j^r(m)\}_{m=1}^{2^{j-1}}$, and can be computed easily using the FFT.

From the windowed imaginary part, we want to compute local cosine coefficients,

$$d^{\circ}_{j,k} = \sum_{l=2^{j-1}-\lfloor\frac{2^j}{3}\rfloor}^{3\cdot 2^{j-1}-\lfloor\frac{2^j}{3}\rfloor-1} \Im\hat{f}(l)\sqrt{\frac{2}{|I_j^{\circ}|}}b^D_{I_j^{\circ}}(l)\cos\Big[(k+1/2)\frac{\pi}{|I_j^{\circ}|}(l-2^{j-1})\Big],$$

for $k = 0,\ldots,2^j-1$. Defining m and N as above, and letting $S_j^i(\cdot)$ represent the folded version of the windowed imaginary part of the data, we may write this as

$$d^{\circ}_{C,k} = \sqrt{\frac{2}{N}}\sum_{m=0}^{N-1} S_j^i(m)\cos\Big[(k+1/2)\frac{\pi m}{N}\Big],$$

which is the third discrete cosine transform (DCT-III) of $\{S_j^i(m)\}_{m=0}^{2^j-1}$, and can also be computed using the FFT.

Combining the Local Trigonometric Coefficients

The final step in computing the jth detail level wavelet coefficients, $\{\alpha^{\circ}_{j,k}\}_{k=0}^{2^j-1}$, is to combine the local cosine and sine coefficients following the relations derived in Section 3.3. Specifically,

$$\alpha^{\circ}_{j,k} = \begin{cases} (-1)^{k+1}(d^{\circ}_{j,k}+c^{\circ}_{j,k}), & \text{if } k = 0,\ldots,2^{j-1}-1 \\ (-1)^{k+1}(d^{\circ}_{j,2^j-k-1}-c^{\circ}_{j,2^j-k-1}), & \text{if } k = 2^{j-1},\ldots,2^j-1 \end{cases}$$

3.4.3 Fine Detail Level Coefficients

For the finest level of detail coefficients, $j = J-1$, Step 3 of the forward algorithm is identical to that of the other detail levels, except for a change of the window used in Step 3(a). The support of the window $b^D_{I^{\circ}_{J-1}}(\cdot)$ extends beyond the frequency domain limits of the data, so we replace it by

$$b^{D*}_{I^{\circ}_{J-1}}(l) = \begin{cases} b^D_{I^{\circ}_{J-1}}(l), & \text{if } 2^{J-2}-\lfloor\frac{2^{J-1}}{3}\rfloor \le l \le 2^{J-2}+\lfloor\frac{2^{J-1}}{3}\rfloor \\ 1, & \text{if } 2^{J-2}+\lfloor\frac{2^{J-1}}{3}\rfloor < l \le 2^{J-1} \end{cases}$$

This function approximates a truncation in the frequency domain of the (unscaled) 'mother' Meyer wavelet at resolution level $J-1$. Folding about the left endpoint of I_{J-1} is done in the same manner as the other detail levels, while folding about the right endpoint is simply done with a vector of zeros. Steps 3(c) and 3(d) are unchanged.

3.5 Inverse Meyer Wavelet Transform

Detailed algorithms for Step 1 of the IMWT are presented in this section. The algorithms are essentially the reverse of those for the FMWT, with the goal being to recover projections of the signal, $\hat{f}$, onto a set of frequency bands that partition the discrete circle. Summing these projections yields $\hat{f}$ itself. Each frequency band roughly corresponds to the support of the wavelets underlying a given level of wavelet coefficients. As with the FMWT, the difference in specifics among the coefficient levels derives from the different wavelets underlying each level.

3.5.1 Coarse Level Projection

The coarse level wavelet coefficients, $\{\beta_{C,k}\}_{k=0}^{2^C-1}$, are used to calculate the coefficients of local cosine and sine bases from which we recover the projections of $\{\Re\hat{f}\}$ and $\{\Im\hat{f}\}$, respectively, onto $\tilde{I}_C^\circ = [-2^{C-1}, 2^{C-1}] \cap \mathbf{Z}$ (the discrete analogue of the projection interval $\tilde{I}_C = [-2^{C-1}, 2^{C-1}]$ for functions on $I\!R$).

Coarse Level Projection Algorithm

1. For the coarse level of coefficients,

   (a) Separate coefficients into local cosine and sine coefficients, `CR` and `CI`, corresponding to the real and imaginary parts of $\hat{f}$, respectively.

   (b) Compute ...

i. $\{S^r_C\}$ as the DCT-I of the cosine coefficients :

`FBR = DCT_I( CR ) .`

ii. $\{S^i_C\}$ as the DST-I of the sine coefficients :

`FBI = DST_I( CI ) .`

(c) Unfold ...

i. $\{S^r_C\}$: `BR = UNFOLD( FBR,` $[-2^{C-1}, 2^{C-1}]$`,` $(\frac{2^C}{6}, \frac{2^C}{6})$`, (+,+) )`

ii. $\{S^i_C\}$: `BI = UNFOLD( FBI,` $(-2^{C-1}, 2^{C-1})$`,` $(\frac{2^C}{6}, \frac{2^C}{6})$`, (-,-) )`

(d) Window ...

i. Unfolded $\{S^r_C\}$ by $b^D_{\tilde{I}^\circ_C}$:

`RPROJ = WINDOW( BR, Mod_Father_C,` $[-2^{C-1}, 2^{C-1}]$`,` $(\frac{2^C}{6}, \frac{2^C}{6})$ `)`

ii. Unfolded $\{S^i_C\}$ by $b^D_{\tilde{I}^\circ_C}$:

`IPROJ = WINDOW( BI, Mod_Father_C,` $(-2^{C-1}, 2^{C-1})$`,` $(\frac{2^C}{6}, \frac{2^C}{6})$ `)`

Produces compactly supported sequences `RPROJ` and `IPROJ`, containing the projections $\{P_{\tilde{I}^\circ_C} \Re \hat{f}\}$ and $\{P_{\tilde{I}^\circ_C} \Im \hat{f}\}$

(e) Skip. No extension necessary.

Separation into Local Trigonometric Coefficients

The local cosine representation of the projection of $\{\Re \hat{f}\}$ onto $\tilde{I}^\circ_C$ has 2^{C+1} coefficients, $\{\tilde{c}^\circ_{C,k}\}_{k=0}^{2^{C+1}-1}$, which may be calculated from the coarse level wavelet coefficients as

$$\tilde{c}^\circ_{C,k} = \begin{cases} \beta^\circ_{C,0}, & \text{if } k = 0 \\ \frac{(-1)^{k/2}}{\sqrt{2}}(\beta^\circ_{C,\frac{k}{2}} + \beta^\circ_{C,2^C-\frac{k}{2}}), & \text{if } k \text{ is even} \\ 0, & \text{if } k \text{ is odd} \end{cases}$$

The local sine representation of the projection of $\{\Im \hat{f}\}$ onto $\tilde{I}^\circ_C$ has the same number, $\{\tilde{d}^{\circ}_{C,k}\}_{k=0}^{2^{C+1}-1}$, and may be calculated as

$$\tilde{d}^{\circ}_{C,k} = \begin{cases} 0, & \text{if } k = 0 \\ \frac{(-1)^{\frac{k+2}{2}}}{\sqrt{2}}(\beta^{\circ}_{C,\frac{k}{2}} - \beta_{C,2^C-\frac{k}{2}}), & \text{if } k \text{ is even} \\ 0, & \text{if } k \text{ is odd} \end{cases}$$

See Section 3.3 for details concerning the derivation of these coefficient relations.

Inversion of Local Trigonometric Coefficients

Following the derivations in Section 3.3, we may express the projection of $\{\Re\hat{f}\}$ onto $\tilde{I}^{\circ}_C$, as a function of $l \in \mathbf{Z}$, by the equations

$$\begin{aligned} P_{\tilde{I}^{\circ}_C}(\Re\hat{f})(l) &= \frac{\tilde{c}^{\circ}_{C,0}}{\sqrt{|\tilde{I}^{\circ}_C|}} b^D_{\tilde{I}^{\circ}_C}(l) + \sqrt{\frac{2}{|\tilde{I}^{\circ}_C|}} b^D_{\tilde{I}^{\circ}_C}(l) \sum_{k=1}^{2^{C+1}-1} \tilde{c}^{\circ}_{C,k} \cos\Big[k\frac{\pi}{|\tilde{I}^{\circ}_C|}(l + 2^{C-1})\Big] \\ &= b^D_{\tilde{I}^{\circ}_C}(l)\Big\{ \sqrt{\frac{2}{|\tilde{I}^{\circ}_C|}} \sum_{k=0}^{2^{C+1}-1} \tilde{c}^{\circ\prime}_{C,k} \cos\Big[k\frac{\pi}{|\tilde{I}^{\circ}_C|}(l + 2^{C-1})\Big] \Big\} \\ &\equiv b^D_{\tilde{I}^{\circ}_C}(l) \cdot S^r_C(l) \ . \end{aligned}$$

Recall from Section 3.2 that if we can recover $S^r_C(\cdot)$, for $l \in [-2^{C-1}, 2^{C-1}] \cap \mathbf{Z}$, from the local cosine coefficients $\{\tilde{c}^{\circ}_{C,k}\}$, then unfolding and windowing the result will yield the desired projection $P_{\tilde{I}^{\circ}_C}\Re\hat{f}$. Letting $N = 2^{C+1}$ and defining $m = l + 2^{C-1}$, we get

$$S^r_C(m) = \sqrt{\frac{4}{N}} \sum_{k=0}^{N-1} \tilde{c}^{\circ}_{C,k} \cos\Big[\frac{2\pi km}{N}\Big].$$

Writing this as a sum up to N (defining $\tilde{c}^{\circ\prime}_{C,N} = 0$) and substituting $m' = 2m$, we see that $\{S^r_C(m')\}$ can be calculated using a DCT-I and sampling appropriately from the output.

In a similar fashion, the projection of $\{\Im\hat{f}\}$ onto $\tilde{I}^{\circ}_C$ may be expressed as

$$\begin{aligned} P_{\tilde{I}^{\circ}_C}(\Im\hat{f})(l) &= b_{\tilde{I}^{\circ}_C}(l)\Big\{ \sqrt{\frac{2}{|\tilde{I}^{\circ}_C|}} \sum_{k=1}^{2^{C+1}-1} \tilde{d}^{\circ}_{C,k} \sin\Big[k\frac{\pi}{|\tilde{I}^{\circ}_C|}(l + 2^{C-1})\Big] \Big\} \\ &\equiv b_{\tilde{I}^{\circ}_C}(l) \cdot S^i_C(l). \end{aligned}$$

Defining m and N as above, now for $l \in (-2^{C-1}, 2^{C-1}) \cap \mathbf{Z}$, we get

$$S_C^i(m) = \sqrt{\frac{4}{N}} \sum_{k=1}^{N-1} \tilde{d}^{\circ}_{C,k} \sin\Big[\frac{2\pi km}{N}\Big],$$

which is recognized to be a DST-I. From this, via the **unfolding** in Step 1(c) and the **windowing** in Step 1(d), we will recover $P_{\tilde{I}^{\circ}_C}(\Im \hat{f})$ at all values of l in the support of $b_{\tilde{I}_C}$.

Unfold

From Step 1(b) we have vectors of length 2^C+1 and 2^C-1, representing **folded** versions of $\{\Re \hat{f}\}$ and $\{\Im \hat{f}\}$, respectively, with respect to $b^D_{\tilde{I}^{\circ}_C}$. To obtain the corresponding **projections**, we first **unfold** the two vectors, after which we will **window** the results in Step 1(d).

The **projection** of the real part of $\hat{f}$ onto $\tilde{I}^{\circ}_C$ has a $(+,+)$ **polarity**. This dictates that the first $\lfloor \frac{2^C}{6} \rfloor$ values following the first point of $\{S^r_C\}$ be reflected to the left about that point, and the last $\lfloor \frac{2^C}{6} \rfloor$ values preceding the last point of $\{S^r_C\}$ be reflected to the right of that point. This yields an **unfolded** vector of length $2^C + 2\lfloor \frac{2^C}{6} \rfloor + 1$.

The **projection** of the imaginary part of $\hat{f}$ onto $\tilde{I}^{\circ}_C$ has a $(-,-)$ **polarity**, dictating that the first $\lfloor \frac{2^C}{6} \rfloor$ and last $\lfloor \frac{2^C}{6} \rfloor$ values of $\{S^i_C\}$ be reflected about endpoints set to zero and then multiplied by -1. This yields a vector of the same length as above.

Window

The **unfolded** versions of $\{S^r_C\}$ and $\{S^i_C\}$ are multiplied by the same **window** $b^D_{\tilde{I}^{\circ}_C}$ as was used in the FMWT at the coarse level. This yields the **projections** of $\{\Re \hat{f}\}$ and $\{\Im \hat{f}\}$, respectively, onto $\tilde{I}^{\circ}_C$.

Extension

The support of the window $b_{\tilde{I}_C}$ is such that we recover our projections at all (relevant) positive and negative frequencies at once. No extension is necessary.

3.5.2 Detail Level Projection

The jth detail level wavelet coefficients, $\{\alpha_{j,k}\}_{k=0}^{2^j-1}$, are used to calculate the coefficients of local sine and cosine bases from which we recover the projections of $\{\Re\hat{f}\}$ and $\{\Im\hat{f}\}$, respectively, onto $I_j^\circ = [2^{j-1}, 2^j] \cap \mathbf{Z}$ (the discrete analogue of the projection interval $I_j = [2^{j-1}, 2^j]$ for functions on $I\!R$). These projections are then extended to $-I_j^\circ$, the symmetric reflection of I_j° about zero, by symmetry and antisymmetry, respectively.

Detail Level Projection Algorithm

1. For detail level $j = C, \ldots, J-2$,

   (a) Separate coefficients into local sine and cosine coefficients, `CR` and `CI`, corresponding to the real and imaginary parts of $\hat{f}$, respectively.

   (b) Compute ...

      i. $\{S_j^r\}$ as the DST-II of the sine coefficients :

         `FBR = DST_II( CR ) .`

      ii. $\{S_j^i\}$ as the DCT-II of the cosine coefficients :

         `FBI = DCT_II( CI ) .`

   (c) Unfold ...

      i. $\{S_j^r\}$: `BR = UNFOLD( FBR,` $(2^{j-1}, 2^j]$`,` $(\frac{2^j}{3}, \frac{2^{j+1}}{3})$`, (-,+) )`

      ii. $\{S_j^i\}$: `BI = UNFOLD( FBI,` $[2^{j-1}, 2^j)$`,` $(\frac{2^j}{3}, \frac{2^{j+1}}{3})$`, (+,-) )`

   (d) Window ...

i. Unfolded $\{S_j^r\}$ by $b^D_{I_j^\circ}$:

`RPROJ = WINDOW( BR, Mod_Mother_j,` $(2^{j-1}, 2^j]$, $(\frac{2^j}{3}, \frac{2^{j+1}}{3})$ `)`

ii. Unfolded $\{S_j^i\}$ by $b^D_{I_j^\circ}$:

`IPROJ = WINDOW( BI, Mod_Mother_j,` $[2^{j-1}, 2^j)$, $(\frac{2^j}{3}, \frac{2^{j+1}}{3})$ `)`

Produces compactly supported sequences `RPROJ` and `IRPOJ`, containing the projections $\{P_{I_j^\circ}\Re\hat{f}\}$ and $\{P_{I_j^\circ}\Im\hat{f}\}$

(e) Extend ...

i. $\{P_{I_j^\circ}\Re\hat{f}\}$ to $-I_j^\circ$:

`EXTRPROJ = EXTEND( RPROJ,` $(2^{j-1}, 2^j]$, $(\frac{2^j}{3}, \frac{2^{j+1}}{3})$, `'even' )`

ii. $\{P_{I_j^\circ}\Im\hat{f}\}$ to $-I_j^\circ$:

`EXTIPROJ = EXTEND( IPROJ,` $[2^{j-1}, 2^j)$, $(\frac{2^j}{3}, \frac{2^{j+1}}{3})$, `'odd' )`

Separation into Local Trigonometric Coefficients

The local sine representation of the projection of $\{\Re\hat{f}\}$ onto I_j° has 2^{j-1} coefficients, $\{c^\circ_{j,k}\}_{k=0}^{2^{j-1}-1}$, which may be calculated from the level j wavelet coefficients as

$$c^\circ_{j,k} = \frac{(-1)^{k+1}}{2}(\alpha^\circ_{j,k} + \alpha^\circ_{j,2^j-k-1}).$$

The local cosine representation of the projection of $\{\Im\hat{f}\}$ also has 2^{j-1} coefficients, $\{d^\circ_{j,k}\}_{k=0}^{2^{j-1}-1}$, which may be calculated as

$$d^\circ_{j,k} = \frac{(-1)^{k+1}}{2}(\alpha^\circ_{j,k} - \alpha^\circ_{j,2^j-k-1}).$$

Inversion of Local Trigonometric Coefficients

We may express the projection of $\{\Re\hat{f}\}$ onto I_j°, as a function of $l \in \mathbf{Z}^+$, by the equation

$$\begin{aligned} P_{I_j^\circ}(\Re\hat{f})(l) &= b_{I_j^\circ}(l)\Big\{ \sqrt{\frac{2}{|I_j^\circ|}} \sum_{k=0}^{2^{j-1}-1} c^\circ_{j,k} \sin\Big[(k+1/2)\frac{\pi}{|I_j^\circ|}(l-2^{j-1})\Big] \Big\} \\ &= b_{I_j^\circ}(l)\cdot S_j^r(l). \end{aligned}$$

Similar to the coarse level **projection** algorithm, if we can recover $S_j^r(\cdot)$, for $l \in (2^{j-1}, 2^j] \cap \mathbf{Z}$, then **unfolding** and **windowing** the result will yield the desired **projection** $P_{I_j^\circ}\Re\hat{f}$. Letting $N = 2^j$ and $m = l - 2^{j-1}$, we get

$$S_j^r(m) = \sqrt{\frac{2}{N}} \sum_{k=0}^{N-1} c_{j,k}^\circ \sin\Big[\frac{(k+1/2)\pi m}{N}\Big].$$

This implies that $\{S_j^r(m)\}$ may be calculated using a DST-II.

Similarly, the **projection** of $\{\Im\hat{f}\}$ onto I_j° may be expressed as

$$\begin{aligned} P_{I_j^\circ}(\Im\hat{f})(l) &= b_{I_j^\circ}(l)\Big\{ \sqrt{\frac{2}{|I_j^\circ|}} \sum_{k=0}^{2^{j-1}-1} d_{j,k}^\circ \cos\Big[(k+1/2)\frac{\pi}{|I_j^\circ|}(l-2^{j-1})\Big] \Big\} \\ &= b_{I_j^\circ}(l)\cdot S_j^i(l). \end{aligned}$$

Defining N and m as above, now for $l \in [2^{j-1}, 2^j) \cap \mathbf{Z}$, we get

$$S_j^i(m) = \sqrt{\frac{2}{N}} \sum_{k=0}^{N-1} d_{j,k}^\circ \cos\Big[\frac{(k+1/2)\pi m}{N}\Big].$$

Hence $\{S_j^i(m)\}$ may be calculated using a DCT-II. **Unfolding** and **windowing** the result in Steps 1(c) and 1(d) will recover $P_{I_j^\circ}\Im\hat{f}$ at all positive values l in the support of $b_{I_j^\circ}$.

Unfold

From Step 1(b) we have 2 vectors of length 2^{j-1}, representing **folded** versions of $\{\Re\hat{f}^+\}$ and $\{\Im\hat{f}^+\}$, respectively, with respect to $b_{I_j^\circ}^D$. To obtain the corresponding **projections**, we first **unfold** the two vectors, after which we will **window** the results in Step 1(d).

The **projection** of the real part of $\hat{f}$ onto I_j° has a $(-,+)$ **polarity**. This dictates that the first $\lfloor\frac{2^j}{3}\rfloor$ values of $\{S_j^r\}$ be reflected about an endpoint value of zero at $l = 2^{j-1}$ and multiplied by -1. The last $2^{j-1} - \lfloor\frac{2^j}{3}\rfloor - 1$ values preceding the last point of $\{S_j^r\}$ are reflected to the right about that point. This process yields an **unfolded** vector of length 2^j.

The projection of the imaginary part of $\hat{f}$ onto I_j° has a $(+,-)$ polarity, dictating that the opposite of the above steps be done. Specifically, the first $\lfloor \frac{2^j}{3} \rfloor$ values following the first point of $\{S_j^i\}$ are reflected to the left of that point, while the last $2^{j-1} - \lfloor \frac{2^j}{3} \rfloor - 1$ values are reflected to the right of an endpoint value of zero at $l = 2^j$ and multiplied by -1. This also yields an unfolded vector of length 2^j.

Window

The unfolded versions of $\{S_j^r\}$ and $\{S_j^i\}$ are multiplied by the same window $b^D_{I_j^\circ}$ as was used in the FMWT at the jth detail level. This yields the projections of $\{\Re\hat{f}\}$ and $\{\Im\hat{f}\}$, respectively, onto I_j°.

Extension

Since the function $b_{I_j^\circ}$ is bimodal and symmetric about zero, we use extension to obtain the projections of $\{\Re\hat{f}\}$ and $\{\Im\hat{f}\}$ onto $-I_j^\circ \cup I_j^\circ$ with respect to the pair of windows defined by $b_{I_j^\circ}$. As $\{\Re\hat{f}\}$ is even, its projection onto $-I_j^\circ$ is the symmetric reflection of its projection onto I_j°. Similarly, as $\{\Im\hat{f}\}$ is odd, its projection onto $-I_j^\circ$ is the antisymmetric reflection of its projection onto I_j°.

3.5.3 Fine Detail Level Projection

The finest $(j = J-1)$ level of detail wavelet coefficients, $\{\alpha_{J-1,k}\}_{k=0}^{2^{J-1}-1}$, is used to calculate the coefficients of local sine and cosine bases from which we recover the projections of $\{\Re\hat{f}\}$ and $\{\Im\hat{f}\}$, respectively, onto $I_{J-1}^\circ = [2^{J-2}, 2^{J-1}] \cap \mathbb{Z}$ (the discrete analogue of the projection interval $I_{J-1} = [2^{J-2}, 2^{J-1}]$ for functions on $\mathbb{R}$). These projections are then extended to $-I_{J-1}^\circ$, the symmetric reflection of I_{J-1}° about zero, by symmetry and antisymmetry, respectively.

Since the support of the associated window, $b^D_{I_{J-1}^\circ}$, extends beyond the frequency domain limits of the data, we again substitute the window $b^{D*}_{I^\circ_{J-1}}$, as in the FMWT. With this change, the rest of Step 1 in the IMWT is identical to that of the other detail levels, excepts that no unfolding is done to the right of $l = 2^{J-1}$.

3.6 Two-Dimensional Transforms

For two dimensions, we create algorithms for the forward and inverse transforms that are based on the same type of two-dimensional wavelet basis as the Daubechies algorithms. In $L^2(\mathbb{R}^2)$ this is an orthonormal basis composed of all dyadic dilations and translations of the basis functions $\phi(x)\psi(y)$, $\psi(x)\phi(y)$, and $\psi(x)\psi(y)$, where ϕ and ψ are the 'father' and 'mother' Meyer wavelet functions. Due to the nature of this basis and the form of our algorithms for one dimension, it is straightforward to create algorithms for two dimensions using the tools already available.

For the two-dimensional forward Meyer wavelet transform, the algorithm works by applying the appropriate level of the forward algorithm to the rows, as dictated by the wavelet underlying the x-direction, and then applying the appropriate level of the forward algorithm to the columns of the result, as dictated by the wavelet underlying the y-direction. For example, the jth level detail coefficients, with vertical directionality, are based on the basis functions $\{\psi_{j,k}(x)\phi_{j,k'}(y)\}_{k,k'}$. In the context of a matrix of discrete data, these coefficients are computed in two steps. First the one dimensional FMWT is applied to the rows, using the window $b^D_{I^\circ_j}$ as discussed in Section 3.4.2. To the columns of the result we then apply the one-dimensional FMWT using the window $b_{\tilde{I}^\circ_j}$ as discussed in Section 3.4.1.

The two-dimensional inverse Meyer wavelet transform simply reverses this process. For example, to invert the jth level coefficients of the above paragraph, the one-dimensional IMWT is applied first to the columns, using $b_{\tilde{I}^\circ_j}$, and then to the rows of the result, using $b^D_{I^\circ_j}$. After this has been done at all levels, for all directions, the resulting matrices are summed, yielding the original matrix of data.

3.7 Implementation in Matlab

The FMWT and IMWT algorithms were implemented for the computer as a series of Matlab functions. The functions were organized and written so as to mirror the descriptions above, with the hope of facilitating the use and understanding of them.

The forward algorithm consists of a top level function called `FwdMeyerTrans` that takes a signal vector, and specifications of the coarse level and smoothness of the wavelets in frequency space. The output is a vector of wavelet coefficients. This function computes the coefficients on a level by level basis, using the algorithms in Section 3.4 by calling the functions `CoarseMeyerCoefs`, `DetailMeyerCoefs`, and `FDetailMeyerCoefs`. Each of these three coefficient functions windows and folds the real and imaginary parts of the signal using the function `MeyerFold`, computes forward discrete trigonometric transforms using `DCTI`, `DSTI`, `DCTIII`, or `DSTIII`, and combines the trigonometric coefficients using `CoefCombine`. The discrete trigonometric transforms constitute the major transformations involved in terms of computing time. We use the $O(n \log_2(n))$ Fast Fourier Transform to do these calculations at each of the $\log_2(n)$ resolution levels, yielding a forward transform algorithm of $O(n \log_2^2(n))$ operations.

The inverse algorithm takes a vector of wavelet coefficients and the same specifications of level and smoothness, and outputs the original signal. The top level function is called `InvMeyerTrans`, and calls the functions `CoarseMeyerProj`, `DetailMeyerProj`, and `FDetailMeyerProj` to perform the calculations in the algorithms of Section 3.5 on a level by level basis. Each of the latter three functions calls the function `CoefSeparate` to create discrete local sine and cosine coefficients from the wavelet coefficients of a given level. Following this separation is a call to `DCTI`, `DSTI`, `DCTII`, or `DSTII` to invert the coefficients. The final step is to `unfold` and `window` the inversions using `MeyerUnfold` and then `extend` the result, when necessary, and fill in zero values using `ExtendProj`. Again, the Fast Fourier Transform is used to compute the discrete trigonometric transforms, yielding an $O(n \log_2^2(n))$ algorithm.

Note that the definitions of the 'father' and 'mother' Meyer wavelets given in Section 3.3 depend on an auxiliary function, denoted $\nu(\cdot)$, with the properties

$$\nu(x) = \begin{cases} 0, & \text{if } x \leq 0 \\ 1, & \text{if } x > 1 \end{cases}$$

and $\nu(x) + \nu(1-x) = 1$. Following Gao [22], we use polynomial functions $\nu_p(\cdot) \in C^p(\mathbb{R})$, of degree $2p+1$, on the interval $[0,1]$. These can be derived as

$$\nu_p(x) = x^{p+1} \sum_{k=0}^{p} b_k x^k$$

for unique coefficients $\{b_0, \ldots, b_k\}$. In our code, `MeyerNu`, we allow the option of $p = 0, 1, 2$, or 3, corresponding to the functions

$$\begin{aligned} \nu_0(x) &= x \\ \nu_1(x) &= x^2(3-2x) \\ \nu_2(x) &= x^3(10-15x+6x^2) \\ \nu_3(x) &= x^4(35-84x+70x^2-20x^3) \ . \end{aligned}$$

Chapter 4

Applications of Meyer Transform Algorithms

We will present a number of applications of our algorithms for the fast forward and inverse, discrete periodic Meyer wavelet transforms. We begin in Section 4.1 by looking at the decay of the Meyer wavelets in the time domain. We continue in Section 4.2 with a study to compare the sensitivity of Meyer and Daubechies wavelets to singularities. In Section 4.3 we give the results of a timing study comparing computation time for these two wavelets. Sections 4.4 and 4.5 address modifications of the algorithms in Chapter 3 that allow us to compute operators and biorthogonal decompositions of operators through the addition of a single step.

4.1 Decay of the Meyer Wavelets

Although the Meyer wavelet $\psi \in C^\infty(\mathbb{R})$ is infinitely differentiable, its decay depends on the auxiliary function ν. If we let $\nu = \nu_p$, as defined in the previous chapter, then $\hat{\psi} \in C^p(\mathbb{R})$ and the rate of decay of ψ should increase with p. In Figure 4.1 we plot $\log(|\psi(x)|)$, $x > 1/2$, for $p = 0, 1, 2$, and 3.

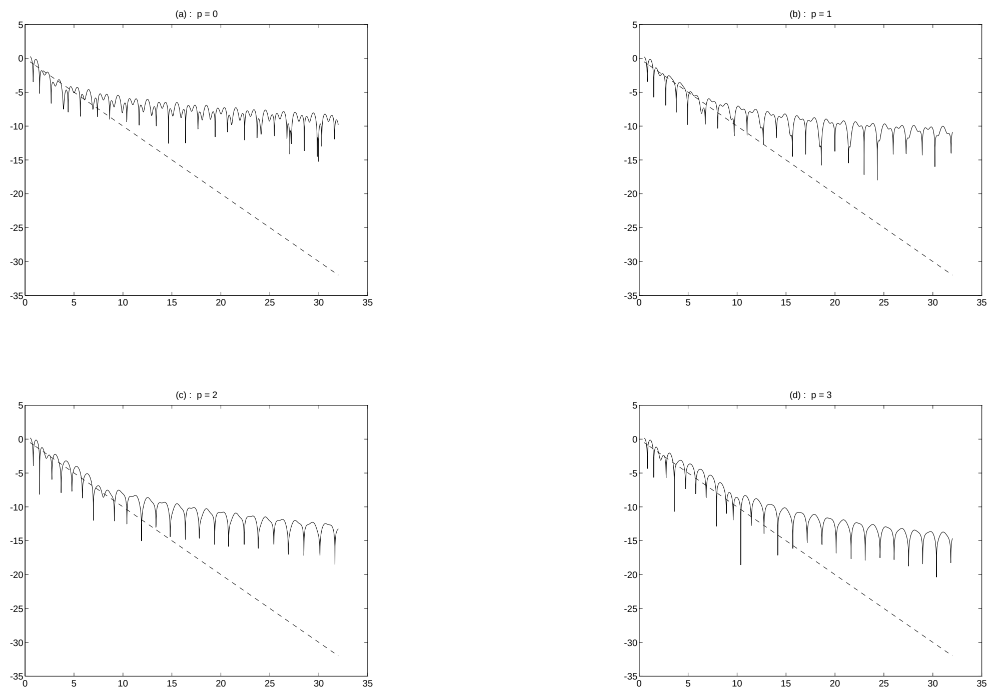


Figure 4.1: Plots of $\log(|\psi(x)|)$ for $x > 1/2$, using $\nu \in C^p(\mathbb{R})$. Figures (a), (b), (c), and (d) correspond to $p = 0, 1, 2,$ and 3.

The line $y = -x = \log(e^{-x})$ is plotted for comparison. Note that in all four figures $\log(|\psi|)$ decays exponentially fast at first, but then slows as x increases. The "elbow-point," at which the apparent exponential decay ends, increases with increasing p.

4.2 Sensitivity Study

Wavelets with infinite support in the time domain are often cited as being less localized in their sensitivity to singularities than those of compact support, such as the Daubechies wavelets. Since the Meyer wavelets are a member of the former group, we set out to evaluate this criticism. We compare our Meyer wavelet transforms with that of a periodic $D6$ Daubechies wavelet transform.

The evaluation was done using the boxcar function

$$f(x) = \begin{cases} 0, & \text{if } x \in [0, \frac{1}{4}] \\ 1, & \text{if } x \in (\frac{1}{4}, \frac{3}{4}) \\ 0, & \text{if } x \in [\frac{3}{4}, 1] \end{cases}$$

since this function is periodic on $[0, 1]$ and is constant except for two jump discontinuities. Only wavelets whose support contains the points $\frac{1}{4}$ or $\frac{3}{4}$ should have significantly nonzero coefficients.

A simple graphical comparison involves plotting the coefficients from our forward Meyer wavelet transform with those of the forward $D6$ Daubechies wavelet transform. The signal was of length $n = 2^{11}$. The result appears in Figure 4.2, where resolution levels $j = 3, \ldots, 10$ have been plotted.

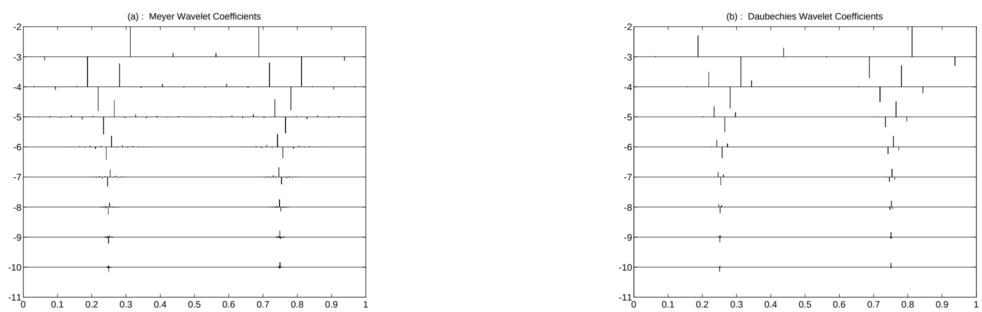


Figure 4.2: (a) Meyer Wavelet Coefficients of Boxcar. (b) $D6$ Daubechies Wavelet Coefficients of Boxcar.

The difference in spread at the finest 3 levels is almost indistinguishable. At resolution levels $j \leq 7$, the Meyer wavelet transform begins to exhibit a noticeably larger spread than the $D6$ transform. However, the size of the Meyer coefficients away from the immediate vicinity of the singularities appears quite small as compared to that of the central 2 or 3. To quantify the degree to which this is true, we computed the ratio of the energy (sum of squares) of the central three coefficients about both singularities to that of the entire level of coefficients, for each level. The central

3 are used because examination of Figure 4.2(b) shows that only the central 3 $D6$ coefficients are visible at most resolution levels. Table 4.1 gives the results.

j	3	4	5	6	7	8	9	10
$p=0$	99.3	96.6	97.1	97.1	97.3	97.6	98.3	98.9
$p=1$	100	97.9	97.4	97.5	97.6	97.8	98.0	98.1
$p=2$	99.7	98.4	97.4	97.4	97.4	97.5	97.5	97.3
$p=3$	99.2	98.6	97.2	97.2	97.2	97.2	97.0	96.7

Table 4.1: Fraction of energy contained in the 'three central' wavelet coefficients of the Meyer wavelet transform of a boxcar function. Columns correspond to resolution levels $j = 3, \ldots, 10$. Rows correspond to increasing decay of ψ.

The same energy ratio for the $D6$ transform was approximately 100%. The numbers in Table 4.1 show that all levels of coefficients from the Meyer wavelet transform, independent of the four values of p, contain approximately 97% of the total energy in their central 3 coefficients. This result makes a strong argument in favor of the localization of the discrete, Meyer wavelet transforms. Another approach would be to plot the sorted, absolute values of the coefficients for both the Meyer and $D6$ wavelet transforms. Such plots showed almost identical curves, suggesting that both transforms compress the boxcar signal equally.

4.3 Timing Study

In addition to the issue of localization addressed in the previous section, there is also the issue of computation time required for the Meyer and Daubechies wavelet transforms. Recall that while the Daubechies transforms may be computed using order $O(n)$ algorithms, our algorithms for the Meyer transforms are order $O(n \log_2^2(n))$. This would suggest longer computation time for the Meyer transforms. However, the $n \log_2(n)$ factor, at each of the $\log_2(n)$ resolution levels, in the Meyer transform comes from application of the Fast Fourier Transform (FFT). Since the FFT is often implemented in hardware, it is conceivable that the Meyer transforms could be as fast, if not faster, than the Daubechies transforms, especially for large n. Such a phenomenon has been observed in comparing Daubechies wavelet transforms and discrete cosine packet transforms (Donoho [13]).

A timing study will be conducted to address this issue. Vectors containing discretiztions of the curve $y = sin(2\pi x)$, $x \in [0, 1]$, will be computed, with lengths $n = 2^a, \ldots, 2^b$. The $D6$ Daubechies and Meyer forward transforms will be computed 10 times each, for each value of n. A table such as Table 4.2 will show the mean CPU times required for the calculations.

n	2^a	4	5	6	7	8	9	2^b
Meyer	( )	( )	( )	( )	( )	( )	( )	( )
$D6$	( )	( )	( )	( )	( )	( )	( )	( )

Table 4.2: Mean CPU times for forward $D6$ Daubechies and Meyer wavelet transforms of the discretized function $y = sin(2\pi x)$, of length n. Means are based on 10 trials each. Standard deviations are given in parentheses.

A similar comparison is planned for the inverse transforms. In this case, the inverse transforms of the appropriate forward transforms of $y = sin(2\pi x)$ will be computed. Again, there will be 10 trials for each length n. The reresults will be given in a table such as Table 4.3.

Implementation of these plans requires adapting our Matlab code for the FMWT and IMWT to the C programming language, work still in progress at this writing.

n	2^a	4	5	6	7	8	9	2^b
Meyer	( )	( )	( )	( )	( )	( )	( )	( )
$D6$	( )	( )	( )	( )	( )	( )	( )	( )

Table 4.3: Mean CPU times for inverse $D6$ Daubechies and Meyer wavelet transforms of the appropriate forward transforms of $y = sin(2\pi x)$ (of length n). Means are based on 10 trials each. Standard deviations are given in parentheses.

4.4 Computation of Operators

In the IMWT, for each level j, recall that

$$\{\alpha^\circ_{j,k}\}_{k=0}^{2^j-1} \quad \Rightarrow \quad Proj_{I_j^\circ}(\hat{f}).$$

Now suppose we have a linear operator, L, which for a general function, g, has the property

$$(\widehat{Lg})(\xi) = \ell(\xi)\hat{g}(\xi). \tag{4.1}$$

Then it is easy to show that

$$Proj_{I_j^\circ}(\widehat{Lf})(\xi) = \ell(\xi)[Proj_{I_j^\circ}(\hat{f})(\xi)]. \tag{4.2}$$

A similar relation holds for the coarse level coefficients, $\{\beta^\circ_{C,k}\}$, and their projection onto $\tilde{I}^\circ_C$.

The implication of Equation (4.2) is the following. Given samples of a function f, we may compute Lf using the two stage process

Operator Computation Algorithm

1. Compute forward Meyer transform of `{f_i}`: `w = FMWT( {f_i} )`

2. Compute modified inverse Meyer transform of `w` by inserting

   Step 1(f): `proj_L = OPERATOR_MULTIPLY( proj, L )`

where

---

define `lX` = `OPERATOR_MULTIPLY( X, L)`

  { *Multiply by Operator in Frequency Domain* }

Variables of function `OPERATOR_MULTIPLY` ≡

  `X` := *Frequency domaain input vector of length* n

  `L` := *(string) operator to be computed*

  `lX` := n *length vector of* `L` *operating on* `X`

So with a simple multiplication, we may reconstruct Lf from the wavelet coefficients of f.

Equation (4.1) holds for self-adjoint linear operators such as integration, differentiation, and their fractional equivalents, as well as linear, time-invariant convolution operators. As an example, we consider the fractional derivative operator

$$(L^{\alpha}g)(x) = \frac{d}{dx}\{\ \frac{1}{\Gamma(\alpha)}\int_{x}^{\infty}\frac{g(t)}{(t-x)^{1-\alpha}}dt\}$$

with index $\alpha \in (0,1)$. In the frequency domain this is written

$$\widehat{L^{\alpha}g}(\xi) = l^{\alpha}(\xi)\hat{g}(\xi)$$

where

$$l^{\alpha}(\xi) = sgn(\xi)|2\pi\xi|^{1-\alpha}\mathcal{C}_{\alpha}$$

and $\mathcal{C}_{\alpha}$ is a complex constant depending on α. Figure 4.3 shows the periodic, 'mother' Meyer wavelet at resolution 3 and position 3, $\psi^{\circ}_{3,3}$, and its fractional derivatives of index $\alpha = 0.25, 0.50$, and 0.75. The figures were computed using a variant of

`InvMeyerTrans.m`, called `OpInvMeyerTrans.m` that performs the equivalent of Step 1(f). See Appendix A for more details.

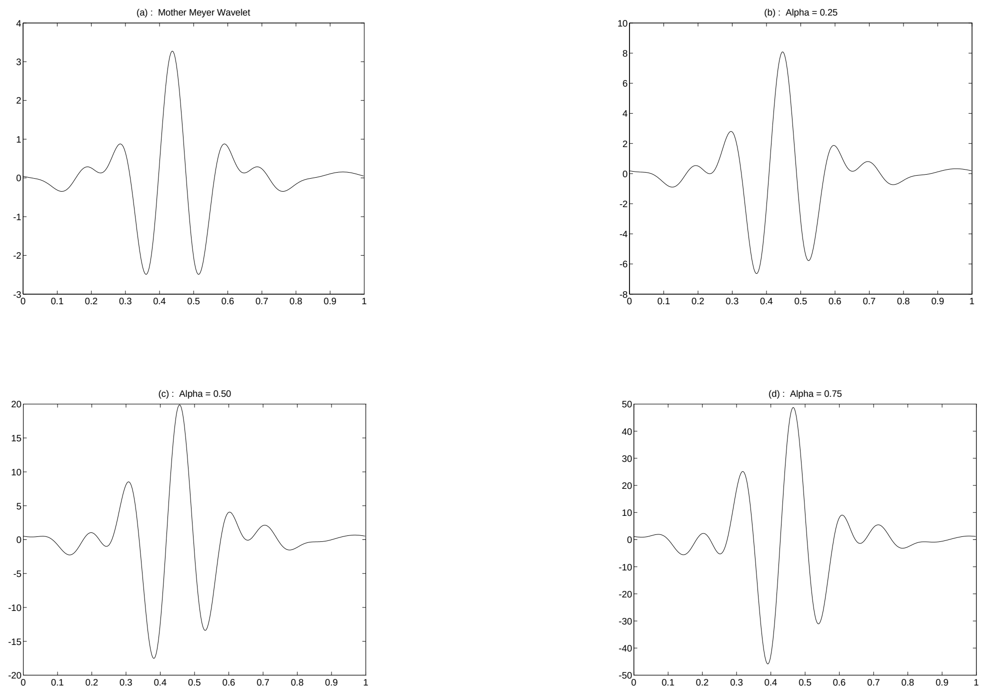


Figure 4.3: (a) Plot of $\psi^{\circ}_{3,3}$, the periodic, 'mother' Meyer wavelet at resolution 3 and position 3. (b), (c), and (d) Fractional derivatives of $\psi^{\circ}_{3,3}$ with indices $\alpha = 0.25, 0.50,$ and 0.75.

4.5 Biorthogonal Decompositions

Consider now the context of linear inverse problems and the wavelet-vaguelette decomposition (WVD), as introduced in Chapter 2.4. Recall that the WVD enables one to create a biorthogonal decomposition of the (homogeneous) linear operator K, through the use of wavelet and vaguelette bases, $\{\psi_{j,k}\}$ and $\{\gamma_{j,k}\}$, and the identity

$$[\gamma_{j,k}, Kf] = \langle \psi_{j,k}, f \rangle.$$

Suppose the vaguelette functions are defined in terms of the wavelet functions by the relation

$$\gamma_{j,k}(x) = (L^*\psi_{j,k})(x),$$

for some linear operator, L, where L has the property in Equation (4.1). Then

$$[\gamma_{j,k}, Kf] = [L^*\psi_{j,k}, Kf] = \langle \psi_{j,k}, L(Kf)\rangle$$

and we can calculate the biorthogonal decomposition of K with Meyer wavelets by using $L(Kf)$ as input for the forward transform algorithm. Specifically, we do the following.

Biorthogonal Decomposition Algorithm

1. Compute FFT of `x` (Step 1 of FMWT): `X = FFT( x )`
2. Multiply by operator in frequency domain:

   `lX = OPERATOR_MULTIPLY( X, L )`
3. Complete FMWT (Steps 2 and 3)

To illustrate our ideas, consider the example of integration, where

$$(Kf)(x) = \int_0^x f(t)dt.$$

The vaguelette in this situation is just

$$\gamma_{j,k} = -\frac{d}{dx}\psi_{j,k},$$

so the operator $L = L^* = -\frac{d}{dx}$. In the frequency domain, the action of L on a function g can be expressed as

$$\widehat{Lg}(\xi) = -\imath 2\pi\xi\hat{g}(\xi).$$

Hence, using our above algorithm, we can compute the discrete vaguelette coefficients of a sampling of Kf, and by their equivalence with the wavelet coefficients of

f, we can recover f itself using the IMWT. Figure 4.4 shows the integral of the function 'Bumps' from Chapter 2, as well as its reconstruction using the procedure just described. Calculations were done using a variant of the `FwdMeyerTrans.m`, called `OpFwdMeyerTrans.m`. Details are given in Appendix A.

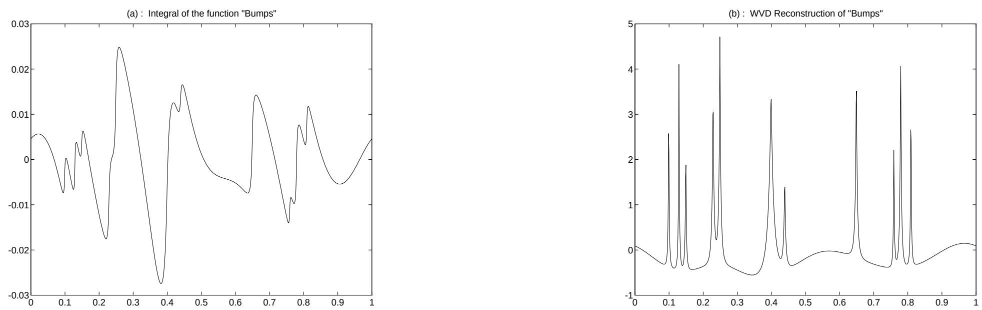


Figure 4.4: (a) Integral of the function 'Bumps'. (b) WVD reconstruction of 'Bumps' from its integral.

Chapter 5

Implementation of the WVD in Tomography

In this chapter, we present a method for implementing the wavelet-vaguelette decomposition in tomographic problems. Our approach hinges on a biorthogonal decomposition of the Radon transform operator, which we present in Section 1. An efficient algorithm for computing this decomposition is given in Section 2. Section 3 is an examiniation of some of the effects of the discretizations involved in moving from theory to applications, in the context of test objects from noiseless data. Finally, in Section 4 we combine our algorithms for computing the decomposition coefficients with wavelet shrinkage methods, using resolution-dependent thresholds, to yield 2D wavelet based reconstructions from noisy data. The two scenarios we consider are parallel beam X-ray tomography and positron emission tomography. Moderate deviation probability calculations are given to justify the thresholds.

5.1 Biorthogonal Decomposition of the Radon Transform Operator

The operator underlying many image reconstruction problems in tomography is the 2D Radon transform operator. In order to apply the WVD formalism to such problems, we first must develop convenient expressions for the biorthogonal decomposition of this operator. Therefore, we begin this chapter by deriving concise, closed-form expressions for the (unnormalized) vaguelettes and the corresponding coefficient functionals.

For $(u, \theta) \in \mathbb{R} \times [0, \pi)$, the 2D Radon transform operator is defined as

$$(Kf)(u, \theta) = \int_{L_{u,\theta}} f(x, y) dx dy,$$

which is the line integral of f over

$$L_{u,\theta} = \{(x, y) | x \cos(\theta) + y \sin(\theta) = u\}.$$

Let $\mathcal{F}_d$ denote the d-dimensional Fourier transform

$$(\mathcal{F}_d g)(\xi) = \int_{\mathbb{R}^d} g(x) e^{-i2\pi\langle x,\xi\rangle} dx.$$

Then the Projection-Slice Theorem (Deans [11]) states that

$$\mathcal{F}_2 f = \mathcal{F}_1 K f,$$

where the right hand side transform is taken with respect to the parameter u. Using this theorem, we can show that

$$[Kf, \gamma_\lambda] = [Kf, K\sigma_\lambda] = \int_{\mathbb{R}^2} (\mathcal{F}_2 f)(\omega)(\mathcal{F}_2 \sigma_\lambda)^*(\omega) \cdot |\omega|^{-1} d\omega.$$

However, since by definition we also have

$$[Kf, \gamma_\lambda] = \langle f, \psi_\lambda \rangle = \int_{\mathbb{R}^2} (\mathcal{F}_2 f)(\omega)(\mathcal{F}_2 \psi_\lambda)^*(\omega) d\omega,$$

we find that

$$(\mathcal{F}_2\sigma_\lambda)(\omega) = |\omega| \cdot (\mathcal{F}_2\psi_\lambda)(\omega) \qquad \text{a.e.}$$

This implies, again using the Projection-Slice Theorem, that the vaguelettes corresponding to the Radon transform can be expressed as

$$\gamma_\lambda(u,\theta) = (\mathcal{F}_1^{-1}\mathcal{F}_2\sigma_\lambda)(u,\theta) = \int_{I\!R} |v| \cdot (\mathcal{F}_2\psi_\lambda)(v\cos(\theta), v\sin(\theta)) \cdot e^{i2\pi uv} dv. \qquad (5.1)$$

Figures 5.1(a-c) show plots of the vaguelette functions $\gamma_{(3,0,\epsilon)}$, where $\epsilon = 1, 2$, and 3 indicates horizontal, vertical, and diagonal directions. Figure 5.1(d) shows a plot of a diagonal vaguelette at resolution level 4, position $(3, 9)$.

By definition, the vaguelette coefficients are the inner product of (5.1) with Kf over $I\!R \times [0, \pi)$. In an effort to obtain a more concise expression for these coefficients, we introduce the backprojection operator,

$$\begin{aligned}(\mathcal{B}(Kf))(x,y) &= \int_0^\pi (Kf)(u_{(x,y,\theta)}, \theta) d\theta \\ &= f(x,y) ** \frac{1}{\sqrt{x^2+y^2}}.\end{aligned}$$

This operator is commonly used in expressing the function f in terms of Kf as

$$f(x,y) = \mathcal{F}_2^{-1}\{|\omega|\mathcal{F}_2\mathcal{B}(Kf)\}(x,y). \qquad (5.2)$$

If we view the vaguelette coefficients as *functions of their indices*, $\lambda = (j, (k_x, k_y), \epsilon)$, then a similar expression may be derived in our context, that is,

$$[\gamma_\lambda, Kf] = \mathcal{F}_2^{-1}\{|\omega| h_j^\epsilon(\omega)\mathcal{F}_2\mathcal{B}(Kf)\}(\frac{k_x}{2^j}, \frac{k_y}{2^j}), \qquad (5.3)$$

where

$$h_j^\epsilon(\omega) = \frac{1}{2^j}(\mathcal{F}_2\psi^\epsilon)(\frac{-\omega}{2^j}).$$

The form of these vaguelette-coefficient functions differs from that of Equation (5.2) only by the replacement of the frequency modulus, $|\omega|$, by the windowed modulus $|\omega|h_j^\epsilon(\omega)$, and the fact that they are evaluated on $2^{-j}\mathbf{Z}^2$ instead of $I\!R^2$. The window function $h_j^\epsilon(\cdot)$ is the 2D Fourier transform of the underlying wavelet, with directional indicator ϵ, scaled to resolution level j. Since $\mathcal{F}_2\psi^\epsilon$ is local in frequency space, Equation (5.3) tells us that each level of coefficients will contain the contribution of Kf from a specific frequency band, as is to be expected from the equivalence of $[Kf, \gamma_\lambda]$ and $\langle f, \psi_\lambda \rangle$.

5.2 An Efficient Algorithm

In practice, the full continuum of line integrals is not available, as is implicitly assumed in the previous section. It is from this incompleteness of the data that the ill-posedness of tomography problems arises. For example, reconstructions that apply Equation (5.2) directly to discrete data are highly unstable, due to the inversion of $|\omega|$ at high frequencies. In order to regularize their solutions in this approach, researchers typically approximate (5.2) by the expression

$$\tilde{f}(x, y) \approx \mathcal{F}_2^{-1}\{|\omega| h(\omega)\mathcal{F}_2\mathcal{B}(Kf)\}(x, y), \tag{5.4}$$

where $h(\cdot)$ is a window function used to damp out the high frequency components of the signal. By discretizing the backprojection operator and replacing $\mathcal{F}_2$ and $\mathcal{F}_2^{-1}$ by their corresponding 2D FFTs, the approximation in (5.4) is implemented as the popular *filtering of backprojected projections* (FBP) method of reconstruction.

Comparing Equations (5.4) and (5.3), one sees that they are identical in format, which motivates us to create a multiresolution version of the FBP algorithm to compute the vaguelette coefficients from discrete data. Specifically, let the data, $\{y\}$, consist of N_{bin} line integrals, or approximations thereof, at each of N_{ang} angles. Also, let $n = 2^J$ be the nearest power of 2 greater than or equal to N_{bin}, and fix $0 \leq C < J$ as the coarsest level of coefficients. Then an $n \times n$ matrix of vaguelette coefficients may be computed from the $N_{bin} \times N_{ang}$ pieces of data using the following algorithm.

Multiresolution FBP Algorithm

1. Backproject data $\{y\}$ to an $n \times n$ matrix.
2. Take the 2D FFT.
3. For each level $j = C, \ldots, J-1$,
   (a) Multiply by the wavelet window function, $h_j^\epsilon(\cdot)$.
   (b) Take the 2D IFFT.
   (c) Decimate the resulting matrix and store.

Many of the software tools necessary for implementing this algorithm are available or can be coded with some effort. Various methods for computing discrete backprojections can be found throughout the tomographic imaging community (e.g. Huesman *et al.* [23]). Our simple versions use linear interpolation and are described in Appendix A. Algorithms for computing the necessary 2D FFTs are implemented in numerous software packages, including both 'S' and Matlab. Finally, the wavelet window functions of Step 3(a) can be computed from their mathematical definitions, either approximately in the case of Daubechies wavelets, or exactly using the Meyer wavelets of Chapter 2.

The decimation process of Step 3(c) requires some explanation. Steps 1 through 3(b) yield a $2^J \times 2^J$ matrix, but at each level $j < J$ and direction ϵ only $2^j \times 2^j$ coefficients are required. Some sampling of the final matrix is clearly necessary. Examination of Equation (5.3) indicates that only matrix entries corresponding to positions $(\frac{k_x}{2^j}, \frac{k_y}{2^j})$ should be sampled, where in the discrete setting we now have $k_x, k_y = 0, 1, \ldots, 2^j - 1$. Using the usual convention of envisioning the output matrix as a discretization of the unit square $[0,1]^2$, Figure 5.2 shows schematically how the values should be sampled in the cases of $j = 1$ and 2.

Note that our multiresolution analogue of the FBP algorithm really is just that. Instead of using a single window function to recover an approximation of f, we use $3(J-C)+1$ different windows, to recover each level and direction of the vaguelette coefficients. Due to the nature of the underlying wavelet basis, these windows effectively partition the 2D frequency space. The increase in computational cost of our method over the FBP algorithm is minimal. The filtering step is done $3\log(n)$ times, but the costly $O(N_{ang}N_{bin}^2)$ backprojection step is done only once. In deriving the identity in Equation (5.3), we were also able to show that

$$[\gamma_\lambda, Kf] = \mathcal{B}\mathcal{F}_1^{-1}\{\bar{h}_j^\epsilon(\xi)\mathcal{F}_1(Kf)\}(\frac{k_x}{2^j}, \frac{k_y}{2^j}), \tag{5.5}$$

where

$$\bar{h}_j^\epsilon(\xi) = |\frac{\xi}{2^j}|(\mathcal{F}_2\psi^\epsilon)(\frac{-\xi\cos(\theta)}{2^j}, \frac{-\xi\sin(\theta)}{2^j}).$$

This identity could be used to create a multiresolution analogue of the current *backprojection of filtered projections* (BFP) method. However, such an approach would require the computation of $3\log(n)$ backprojections, rather than the one backprojection in our MRFBP. Unless we had some special reason to use the approach suggested by (5.5), we would prefer to avoid the additional cost of $3\log(n)$ backprojections.

5.3 Evaluation Using Noiseless Data

Our ultimate goal is to combine the MRFBP algorithm with wavelet shrinkage methods and apply the result to discrete, noisy data. This will be done in the cases of parallel beam X-ray tomography and positron emission tomography in the next section. In this section, we consider discrete but noiseless data in an effort to evaluate the effects of discretization in moving from theory to practice.

5.3.1 Backprojection

We begin by considering the backprojection operator. In the continuum, recall that this may be expressed as

$$\begin{aligned}\mathcal{B}(Kf)(x,y) &= \int_0^\theta (Kf)(x\cos(\theta)+y\sin(\theta),\theta)d\theta \\ &= f(x,y)**\frac{1}{\sqrt{x^2+y^2}},\end{aligned}$$

where Kf is the Radon transform of f, observed on $I\!R \times [0,\pi)$.

In practice, we typically observe N_{bin} *bins* or *strips* at each of N_{ang} equally spaced angles. The operator $\mathcal{B}$ is discretized by replacing the integral over $[0,\pi)$ in its definition by a sum over the N_{ang} angles. Evaluation is done on a grid of equispaced points (x_i, y_j). By convention, the object of interest is taken to be centered within the unit circle. The grid $\{(x_i, y_j)\}_{i,j}$ is therefore taken to be a sampling of the square $[-1,1]^2$, which circumscribes the unit circle. Since the values

$$u = x_i\cos(\theta) + y_j\sin(\theta)$$

are often not directly observed for a given angle θ, we use simple linear interpolation to obtain them.

Backprojection algorithms were coded in Matlab for the two medical imaging applications mentioned above. We defer discussion of the specific geometry of these two situations until the next section. In order to test the backprojection algorithms, we calculated projection data corresponding to a point source, which was approximated by a $N(0,\sigma^2 I_2)$ density, and then backprojected. The standard deviation, σ, of the density, f, was chosen to be small enough so as to make the support of f essentially at the origin. The projection data were calculated on a $N_{bin} = 129 \times N_{ang} = 201$ grid according to the formula

$$(Kf)(u,\theta) = \frac{1}{\sigma\sqrt{2\pi}}\exp\{-\frac{u^2}{2\sigma^2}\}$$

(see Deans [11]).

Since the function f approaches a point source as $\sigma \to 0^+$, the backprojection of Kf should approximate $\frac{1}{\sqrt{x^2+y^2}}$. Multiplying the backprojected data by $\sqrt{x^2+y^2}$ should produce a matrix of approximately uniform values over $[-1,1]^2$. Figure 5.3 shows this normalized backprojection, both from the perspective of a 3D mesh plot and an image. The mesh plot shows that while the result has a uniform trend overall (except at the corners), the surface is quite rough. Looking at the image plot, we see that the largest deviations from the overall surface level are in the corners, where the values rely heavily on extrapolation. Inside the unit circle, the overall surface level is fairly uniform, but displays a number of curious symmetric patterns of artifacts within. The median value of the normalized backprojection was 0.99, with a (centered) l_2 norm of 0.3. These numbers confirm what our eyes have told us.

The observations of the above paragraph clearly suggest that the results of our current approach to backprojection are somewhat simple. More sophisticated algorithms and/or preprocessing of the data will be used for future studies. However, it is hoped that this algorithm will suffice for the present, while we are concerned mainly with developing a methodology and demonstrating its feasibility.

5.3.2 Wavelet Coefficients versus Vaguelette Coefficients

A major benefit of using the WVD with inverse problems is that the vaguelette coefficients of the observed object are actually equal to wavelet coefficients of the unobserved object. In the case of tomography, since we have only a discrete set of observations, we can expect this relationship to be approximate only. Figure 5.4 shows the matrix of vaguelette coefficients for the simulated point source in the previous subsection, obtained by carrying out the rest of the MRFBP algorithm from the backprojected data. Also shown is the direct 2D wavelet transform of the point source. The figures are nearly identical in nature, though the vaguelette coefficients have a slightly rougher background where the wavelet coefficients are zero.

5.3.3 Reconstruction of Gaussian Bumps

Gaussian bumps are good candidates for examining the quality of reconstructions, as both the bumps and their Radon transforms have closed forms that are easily computed. We generated the Radon transforms of Gaussian bumps with standard deviation $\sigma = \frac{1}{10}$, located at $(0,0)$ and $(0.25, 0.25)$. The vaguelette coefficients were calculated as in the previous subsections, and an inverse wavelet transform was applied to yield the reconstructions. These reconstructions, as well as the actual objects, are displayed in Figure 5.5.

The reconstructions are rather crude, but adequate enough to demonstrate the feasibility of our methods. The artifacts introduced by our discretization of the backprojection operator are carried over to the reconstruction, as is typical in such methods. Better reconstructions could be obtained by using a finer grid (we use 128×128). More significant improvement could be effected by the use of more sophisticated methods in the backprojection step, as was mentioned above. However, again, as our purpose in this immediate work is simply to demonstrate the feasibility of our methodology, we leave such refinements to future work. On the other hand, we note that our expectations for our method should remain reasonably modest for the time being, until such improvements may be made.

5.3.4 Reconstruction of Images

In the medical applications with which we will be concerned, the reconstructions are typically of 2D slices of an object of interest, and are therefore displayed as an image instead of with the mesh plots used above. This is important in that the eye is much kinder to the types of artifacts noted above when viewed within the context of a gray-scale image, instead of a 3D plot.

In our medical imaging examples of the next section, we use a 2D computer phantom as our object of interest. Specifically, we use the phantom of Shepp & Logan [49], variations of which have been a standard for nearly 20 years now. Figure 5.6 shows this phantom on a 256×256 grid, as well as noiseless projection data obtained by calculating a discrete set of line integrals with respect to the various circles and ellipses. Black is taken to represent areas of low intensity, while white represents high intensity.

Figure 5.7 shows the direct 2D wavelet transform of the phantom. The decomposition was done with 3 detail levels and the coarse level. One can see quite clearly the manner in which the coarse level coefficients mirror the image itself in their pattern. Also shown is an enhanced version of the same wavelet transform, stressing the smaller detail level coefficients. For the purpose of exposition, the black/white color scheme has been reversed. Note the extreme sparseness of the transform, indicating a high rate of compression and thus suggesting promise for reconstruction methods based on wavelet shrinkage. Note too that the detail levels are almost entirely devoted to representing the edges of the phantom, which primarily characterize the object.

In Figure 5.8 we show the WVD reconstruction of the Shepp/Logan phantom from noiseless projection data. The edges have all been resolved sharply, and the separation of the three small objects near the base of the figure is clear. Some of the rippling artifacts are still apparent, which in turn is responsible for the fact that the shading appears dimmer in the reconstruction than in the original.

It should be noted that such reconstructions are equivalent, both theoretically and numerically, to those obtained by FBP without a filter function. See Chapter 6 for more comments on this subject.

5.4 Reconstruction From Noisy Data

In this final section, we consider two specific examples of tomographic data, those of parallel beam X-ray tomography and positron emission tomography (PET). We use the algorithms derived in Sections 4.1 and 4.2 to compute the empirical wavelet coefficients, and then apply level-dependent thresholds prior to inversion to denoise the underlying image in the wavelet domain. The exact form of the thresholds will depend on the stochastic behavior of the coefficients, which is different in each problem.

5.4.1 Parallel Beam X-ray Tomography

Background

In parallel beam X-ray tomography, one observes N_{bin} equally spaced X-ray readings, at N_{ang} equally spaced angles, within a plane intersecting some object of interest. Each reading measures the cumulative attenuation of the media along the path of the beam, so inversion of the total set of readings should yield a reconstruction of the attenuation coefficient distribution in the chosen plane. Studies of this type are often done for some body part, such as the brain. Figure 5.9 shows a schematic version of the data acquisition.

The Model

We will assume that the plane of interest is centered at $(0,0)$, and its intersection with the object is contained within the unit circle. The observations are modeled as

$$Y(u_b, \theta_a) = \int_{L_{u_b,\theta_a}} f(x,y)dxdy + z_{b,a}$$

where

- f is the attenuation coefficient distribution

- $\theta_a = a\pi / N_{ang} \quad a = 0, \ldots, N_{ang} - 1$
- u_b ranges from -1 to 1, by steps of $1/2N_{bin}$
- $z_{b,a} \sim N(0, \sigma^2) \quad$ (iid)

The methods of Sections 4.1 and 4.2 can be used to compute the empirical vaguelette coefficients from the data $\{Y(u_b, \theta_a)\}_{a,b}$. By the nature of the WVD, we know that these coefficients should be approximately equal to noisy versions of wavelet coefficients of f. As the attenuation coefficient distributions can often be described as sparse objects, wavelet shrinkage is a promising candidate for denoising the data.

Derivation of Thresholds

We begin such an approach by examining the stochastic behavior of the empirical vaguelette coefficients. These coefficients are computed from discrete data as

$$[\gamma_\lambda, Y] = \frac{2\pi}{N_{ang} N_{bin}} \sum_{a=0}^{N_{ang}-1} \sum_{b=1}^{N_{bin}} \gamma_\lambda(u_b, \theta_a) Y(u_b, \theta_a) \ ,$$

which implies that they are normally distributed, with approximate mean and variance

$$E(\ [\gamma_\lambda, Y] \ ) \approx [\gamma_\lambda, Kf] = \langle f, \psi_\lambda \rangle$$

and

$$Var(\ [\gamma_\lambda, Y] \ ) \ \approx \ \frac{2\pi}{N_{ang} N_{bin}} \ \sigma^2 \ ||\gamma_\lambda||_2^2$$

The approximations to the mean and variance are a result of the discreteness of the data, becoming more exact as N_{ang} and N_{bin} increase. The near-orthogonality of the vaguelette functions implies that the vaguelette coefficients are approximately independent.

Note that the variance of the coefficients increases by resolution level like 2^j, a direct result of the ill-posedness of the problem. This suggests that in using wavelet shrinkage methods, a single threshold for all $n \times n$ coefficients will be inadequate. Instead, we should apply different thresholds at each level, where the coefficients are independent, normal, and of equal variance. More specifically, for each level of detail coefficients $j = C, \ldots, J-1$, and each direction ϵ, we shrink the corresponding 2^{2j} coefficients towards zero using the threshold

$$t_{j,\epsilon} = \sqrt{2\log(2^{2j})}\ \ 2^{j/2}\ \ \sigma||\gamma_{(0,0,\epsilon)}||_2\ \ \mathcal{C}_{N_{ang},N_{bin}},$$

where $\mathcal{C}_{N_{ang},N_{bin}} = \frac{2\pi}{N_{ang}N_{bin}}$. The form of these thresholds is analogous to that used by Donoho *et al.*, with adjustments for the increasing noise level and the discreteness of our data. The constants $||\gamma_{(0,0,\epsilon)}||_2$ may be computed using numerical integration, and is especially easy when using the Meyer wavelet as we have done.

Example Using the Shepp/Logan Phantom

We applied our wavelet shrinkage method to data from the Shepp/Logan phantom. Specifically, we calculated the line integrals of this phantom for $N_{bin} = 257$ and $N_{ang} = 402$. Independent and identically distributed Gaussian noise was then added to the noiseless data. The noise level was chosen to be $\sigma^2 = 0.1, 0.03$, and 0.01 so as to yield data with signal-to-noise ratios (SNR) of 10, 15, and 20 decibels (dB), where we define

$$SNR = 10\log_{10}\frac{\sum_{i=1}^{N_{ang}}||(Kf)(\cdot,\theta_i)||^2}{\sigma^2}.$$

The noisy vaguelette coefficients were computed using the MRFBP algorithm, and a soft threshold operator was applied to the two finest levels.

As a comparison, we also computed reconstructions from each set of data using the traditional FBP method. A Parzen window was used, with 0.5 cutoff. See Huesman *et al.* [23] for a discussion and implementation of the FBP and other traditional methods. The reconstructions using both methods are displayed together in Figure 5.10.

Note that the WVD reconstructions appear to be characterized by a noticeably greater reduction in the overall noise level. This is confirmed in Figure 5.11, where we plot a slice of the Fourier transform of the error surface, for both the WVD and FBP reconstructions (20dB). Our wavelet shrinkage approach appears to have effectively removed components of the noise at frequencies greater than or equal to 35 in absolute value, whereas the FBP approach has only done so for frequencies above 90. Since we applied thresholds at levels $j = 6$ and 7, which roughly correspond to frequencies greater than or equal to 32, Figure 5.11(a) demonstrates the effectiveness of wavelet shrinkage in this context.

On the other hand, although wavelet shrinkage looks to do better in removing high frequency noise, this advantage appears to come at the expense of a slight loss of detail in the edges. Such a phenomenon would be similar to the experience of Donoho *et al.* in the case of direct, one-dimensional data. In Figure 5.12 we show close-up views of the WVD and FBP reconstructions (20dB) in the lower region of the phantom containing the three “hot spots”. Although the WVD close-up appears more blurred than that of the FBP, it seems to do a better job of representing the existence of three separate objects. Taken together, the comments of this and the above paragraph seem to indicate that the wavelets do a better job of simultaneously removing noise and preserving the existence of objects than traditional FBP methods, in this context. They could be improved, however, through the use of thresholds more sensitive to edges. See Chapter 6 for more comments in this direction.

5.4.2 Positron Emission Tomography

Background

Positron emission tomography differs from parallel beam X-ray tomography in that the data take the form of total counts of detected radioactive emissions, instead of a attenuation readings. This means that while the Radon transform operator still plays a key role, the stochastic properties of the data are somewhat different.

In PET, researchers are typically interested in obtaining a picture of the metabolic process in an organ. The brain is an organ of particular interest, especially to psychologists of late, who study images of the brain reconstructed from data taken while the patient was performing a particular task [43] [50]. Areas in the reconstruction that indicate a high metabolic rate are interpreted as having been used more than others during performance of the task.

PET data are obtained in the following manner. A radioactively tagged metabolite, say glucose, is injected into the organ of interest, which we will take to be the brain. Concentration of radioactive isotopes in an area will be proportional to the uptake of glucose in that area. Each isotope emits positrons (subatomic particles with the same weight and opposite charge as an electron). After emission, a positron will travel no more than a few millimeters before coming to rest, where almost immediately it is impacted by a nearby electron. The result of this collision is two gamma particles that fly off approximately 180° from each other. While this is happening, the patient sits with his/her head within a ring of crystal detectors, positioned to capture the plane of interest within the brain. If two opposite detectors register the arrival of photons simultaneously, an emission will be recorded as having occurred somewhere within the tube defined by the two detectors. The relationship between tube counts and the distribution of the radioactive isotopes can be described by the Radon transform. By inverting the matrix of tube counts, one should be able to reconstruct an approximation to this distribution, and hence the intensity of glucose uptake.

The Model

We can represent the PET setup with the following model. Let f be the underlying distribution of the radioactive isotopes, within the plane of interest, centered at $(0,0)$. The patient port will be the unit circle, and the detector ring will have radius $\sqrt{2}$. This implies that for $N_{det} = 2^{J+1}$ detectors there will be $N_{ang} = 2^J$ angles and $N_{bin} = 2^{J-1}$ tubes (or bins) per angle with possibly nonzero counts. Tubes will be indexed by polar coordinate pairs (u_b, θ_a) corresponding to the central chords of the tubes. Suppose now that a study is conducted for a duration of T time units. As is discussed in Vardi, Shepp, and Kaufman [56], at the end of the study, the number of counts in tube (u_b, θ_a) can be modeled as $N_{b,a} \sim \text{Poisson}(TI_{b,a})$, where $I_{b,a}$ is the expected number of counts per unit time. Ignoring the slight tendency for counts near the edge of the ring to be detected more frequently, that is to say, ignoring attenuation, we use the approximation

$$I_{b,a} \;\approx\; \frac{1}{\pi^2}\int_{L_{u_b,\theta_a}} f(x,y)dxdy \;=\; \frac{1}{\pi^2}(Kf)(u_b,\theta_a) \; .$$

Representing the data by the random process $Y_T(u_b,\theta_a) = \frac{N_{b,a}}{T}$, the empirical vaguelette coefficients are computed as

$$\begin{aligned}
[\gamma_\lambda, Y_T] &= \frac{2\pi}{N_{ang}N_{bin}} \sum_{a=0}^{N_{ang}-1} \sum_{b=1}^{N_{bin}} \gamma_\lambda(u_b,\theta_a) Y_T(u_b,\theta_a) \\
&= \frac{2\pi}{N_{ang}N_{bin}T} \sum_{a,b} \gamma_\lambda(u_b,\theta_a) N_{b,a}
\end{aligned}$$

Similar to the parallel beam case, these coefficients are unbiased for the underlying wavelet coefficients,

$$E(\;[\gamma_\lambda, Y_T]\;) \approx \frac{1}{\pi^2}[\gamma_\lambda, Kf] = \frac{1}{\pi^2}\langle \psi_\lambda, f\rangle \;\; ,$$

(up to a constant of proportionality), and nearly-independent as a result of the near-orthogonality of the vaguelette functions. However, their distribution is no longer normal. This suggests that we make a more careful study of the stochastic properties of the coefficients in deriving the thresholds to use with wavelet shrinkage methods.

Derivation of Thresholds

Recall that Donoho *et al.'s* $\sqrt{2\log(n)}\frac{\sigma}{\sqrt{n}}$ threshold, in the case of direct data, is basically a probabilistic upper bound on the Gaussian random errors in the space of wavelet coefficients. Specifically, for $Z_1, \ldots, Z_n$ independent and identically distributed standard normals,

$$\begin{aligned}
\Pr(\ \max_i |Z_i| > \sqrt{2\log n}\ ) &\leq 2(1 - \Pr(\ \max_i Z_i < \sqrt{2\log n}\ )) \\
&= 2(1 - [\Pr(\ Z_i < \sqrt{2\log n}\ )]^n) \\
&\approx 2(1 - e^{-1/\sqrt{\log(n)}})\ ,
\end{aligned}$$

which tends to 0 as $n \to \infty$ (see Leadbetter *et al.* [26]). Defining the errors in our present situation to be

$$z_\lambda = [\gamma_\lambda, Y_T] - E(\ [\gamma_\lambda, Y_T]\ )\ ,$$

we would like to derive thresholds $\{t_{j,\epsilon}\}_{j=C}^{J-1}$ such that

$$\Pr(\ \max_{\lambda \in \Lambda_{j,\epsilon}} |z_\lambda| > t_{j,\epsilon}\ ) \tag{5.6}$$

has a similar rate for large j.

Since the exact distribution of the z_λ's is unknown, we proceed by finding threshold values $t_{j,\epsilon}$ such that $\Pr(z_\lambda > t_{j,\epsilon}) \leq \frac{1}{B_j}$. The value of $B_j = 2^{2j}\log(2^{2j})$ is chosen so that the probability in (5.6) tends to zero roughly at the rate for 2^{2j} standard normals. To bound the tail probability of z_λ, we will make use of the following result [2].

Lemma 5.1 (Bennett's Inequality (1962)) *Suppose $X_1, \ldots, X_n$ are independent random variables such that $E[X_i] = 0$ and $E|X_i|^r = v_i^r \leq M^{r-2}\sigma_i^2$, where $r \geq 2$ and $Var(X_i) = \sigma_i^2$. Define $n\sigma^2 = \sum_{i=1}^n \sigma_i^2$. Then*

$$P(\overline{X} \geq t) \leq exp\{\frac{-nt}{M}[(1 + \frac{\sigma^2}{Mt}) \log(1 + \frac{Mt}{\sigma^2}) - 1]\}$$

Defining N_c to be the total number of counts in our PET study, we can re-express the vaguelette coefficients as

$$[\gamma_\lambda, Y_T] = \frac{1}{N_c}\sum_{i=1}^{N_c} \frac{2\pi N_c}{N_{bin}N_{ang}T} \sum_{a,b} \gamma_\lambda(u_b, \theta_a)\delta_{a,b}(\mathcal{E}_i) \ ,$$

where $\mathcal{E}_i$ is the ith emission counted and $\delta_{a,b}(\mathcal{E}_i)$ is an indicator for whether or not it was detected by tube (u_a, θ_b). It follows that $z_\lambda = \overline{W_\lambda}$, where

$$W_{\lambda,i} = \frac{2\pi N_c}{N_{bin}N_{ang}T} \sum_{a,b} \gamma_\lambda(u_b, \theta_a)\{\delta_{a,b}(\mathcal{E}_i) - E[\delta_{a,b}(\mathcal{E}_i)]\}.$$

Since the $W_{\lambda,i}$ are independent, have expectation 0, and are bounded by

$$M = \frac{2\pi N_c}{N_{bin}N_{ang}T} \ 2^{j/2} \ ||\gamma_{(0,0,\epsilon)}||_\infty \ ,$$

we may use Bennett's Inequality to write

$$\begin{aligned} \Pr(z_\lambda > t_{j,\epsilon}) &\leq exp\{\frac{-N_c t_{j,\epsilon}}{M}[(1 + \frac{\sigma^2}{Mt_{j,\epsilon}}) \log(1 + \frac{Mt_{j,\epsilon}}{\sigma^2}) - 1]\} \\ &\approx exp\{\frac{-N_c t_{j,\epsilon}}{M}[(1 + \frac{\sigma^2}{Mt_{j,\epsilon}})(\frac{Mt_{j,\epsilon}}{\sigma^2}) - 1]\} \\ &= exp\{\frac{-N_c t_{j,\epsilon}^2}{\sigma^2}\} \ , \end{aligned}$$

when $Mt_{j,\epsilon}/\sigma^2 < 1$. Hence we find that

$$\Pr(z_\lambda > t_{j,\lambda}) \leq \frac{1}{B_j} \ \Leftrightarrow \ t_{j,\epsilon} \geq \sigma\sqrt{\frac{\log(B_j)}{N_c}}.$$

Note that conditioning on N_c implies that $\{\delta_{a,b}(\mathcal{E}_i)\}_{a,b} \sim Multinomial(1, \{p_{a,b}\})$, for each i, where $p_{a,b} = I_{a,b}/\sum I_{a,b}$. Using this fact, we can show that σ^2 is bounded above by

$$(\frac{2\pi N_c}{N_{bin}N_{ang}T})^2 \ 2^j \quad \| \gamma_{(0,0,\epsilon)} \|_2^2 \ \| \frac{Kf}{\int Kf} \|_\infty .$$

This yields the threshold

$$t_{j,\epsilon} = \sqrt{\log(B_j)} \ \frac{\sqrt{N_c}}{T} \ 2^{j/2} \quad \| \gamma_{(0,0,\epsilon)} \|_2 \ \| \frac{Kf}{\int Kf} \|_\infty \ \ \mathcal{C}_{N_{bin},N_{ang}}, \tag{5.7}$$

where $\mathcal{C}_{N_{bin},N_{ang}}$ is the same as in Section 4.4.1.

Example Using the Shepp/Logan Phantom

We simulated PET data from the Shepp/Logan phantom in a manner consistent with our model. Specifically, we chose $N_{det} = 512$, and calculated the line integrals of the phantom along the central chords between all detector pairs. Poisson counts were then generated for each tube with mean equal $T(Kf)(u,\theta)$, where T was chosen so as to generate an expected 1 million, 2.5 million, and 5 million counts. Similar to the parallel beam case in the previous section, we reconstructed the image using our wavelet shrinkage methods as well as the traditional FBP methods, for each of the three data sets. Figure 5.13 shows the results.

Notice that in this setting the wavelet shrinkage appears to *under-smooth*, in comparison to the FBP. Figures 5.14 and 5.15 add strong evidence towards this conclusion. The former shows a slice of the Fourier transform of the error surface, for both reconstructions, while the latter shows close-up views of the reconstructions in the lower region containing the three "hot spots". These results suggest that we may wish to choose our thresholds in such a way that $\Pr(z_\lambda > t_{j,\epsilon})$ decays faster than the case of normal errors. Also, as was the case with parallel beam data, greater sensitivity to edges is desired.

Chapter 6

Conclusion

Our aim in this dissertation was to explore the use of wavelets to perform signal and image reconstruction in certain linear inverse problems. We chose to work within the framework of the wavelet-vaguelette decomposition, because such a framework, as a wavelet analogue of the singular value decomposition, is both general and familiar. In using the WVD, however, two points should be noted.

<u>Points to Remember in Applying the WVD</u>

$\Rightarrow$ *The operator K must be homogeneous with respect to dilations.*

$\Rightarrow$ *The form of the vaguelette functions will be problem specific.*

A homogeneous operator K leads to a vaguelette basis whose functions may be expressed with respect to a single 'mother' vaguelette function, thus allowing for the possibility of efficient coefficient algorithms in the spirit of the traditional wavelet transforms. The problem-specific functional form of the vaguelettes suggests that problem-specific algorithms are needed. Furthermore, as demonstrated by the examples in parallel beam and positron emission tomography, even when the vaguelettes are the same in two problems, the specifics of the underlying noise processes should be considered when using the WVD in conjunction with wavelet shrinkage methods.

Our choice to work with the Meyer wavelets was motivated in part by a desire to make the calculation of vaguelette coefficients less problem-specific. As the Meyer wavelets have a simple, closed-form expression in the frequency domain, we envisioned them being particularly appropriate for use with operators K which are most easily managed in the frequency domain. Our fast order $O(n \log_2^2(n))$ algorithms for the forward and inverse, discrete Meyer wavelet transforms were a first step towards using these wavelets with the WVD in such problems. With regard to such usage, we point out the following advantages.

<u>Advantages of Using Meyer Wavelets with the WVD</u>

$\Rightarrow$ *Meyer wavelets allow for simple calculation of the level-dependent windows used in our MRFBP algorithm.*

$\Rightarrow$ *A simple adjustment to the forward Meyer wavelet transform enables the computation of vaguelette coefficients for self-adjoint operators.*

The specific problem of tomographic reconstruction, and in particular PET reconstruction, was the primary motivation for our exploring the usage of wavelets in inverse problems. This problem is especially difficult, due to the indirectness and incompleteness of the data. The multitude of new reconstruction methods being developed, after 20 years of experience, suggests that the area is still open for major improvements. We believe that wavelet methods have the potential to offer such improvements.

The approach we took in applying the WVD to tomography was to develop a methodology and demonstrate its feasibility. The three major steps taken to accomplish this task were as follows.

<u>Steps Taken to Implement the WVD in Tomography</u>

$\Rightarrow$ *Developed concise, closed-form expressions for the vaguelette coefficients* *of the Radon transform operator.*

⇒ *Used these expressions to create an efficient algorithm for computing the vaguelette coefficients (MRFBP).*

⇒ *Derived theoretical, level-dependent thresholds for applying wavelet shrinkage methods in reconstructing images.*

These algorithms and thresholds were coded, primarily in the Matlab language, and tested on both noiseless and (simulated) noisy data. The following points are of particular interest or note.

Conclusions Regarding the WVD in Tomography

⇒ *The increase in computational cost of MRFBP over FBP is minimal.*

⇒ *WVD reconstruction done without altering the empirical coefficients is equivalent, numerically and theoretically, to FBP with no filter.*

⇒ *As compared to FBP methods, our WVD-based methods with level-dependent soft thresholding . . .*

(a) *Yielded reconstructions with less high frequency noise in X-ray tomography, but at the expense of some detail.*

(b) *Yielded reconstructions with more high frequency noise in positron emission tomography.*

The thresholding used in our WVD-based reconstructions may be viewed as a form of regularization. Whereas the FBP approach uses a single filter function to damp out high-frequency components, we use a set of level-dependent thresholds adjusted for particular frequency bands.

Although preliminary in nature, we find the above results quite encouraging, as the potential for denoising with respect to a sparse wavelet representation of the underlying (unobserved) object remains largely untapped. For example, the thresholds used in Chapter 5 were theoretically derived; improvement might be had by estimating thresholds from the data. Also, these thresholds are only *level*-dependent, while the WVD localizes information by both level *and* location. These points suggest that we direct future efforts in some or all of the following directions.

<u>Future Efforts in Denoising in Conjunction with the WVD</u>

⇒ *Adaptive thresholds, perhaps using*

  (a) *SURE (Donoho and Johnstone [17])*

  (b) *Cross-validation (Nason [38])*

⇒ *Level- and location-dependent thresholds.*

⇒ *Edge-enhancing filters (Richardson [46]) such as*

  (a) *Mean curvature partial differential equation filters*

  (b) *Weighted Majority-Minimum Range filters*

Despite the overall encouraging tone of our work, we were still disappointed with the general quality of reconstruction, both from the WVD and the FBP. We expect that this quality will be improved in the future by making two important changes.

<u>Improving Overall Reconstruction Quality</u>

⇒ *Increase the sophistication of our treatment of sampling and discretization issues.*

⇒ *Take advantage of more accurate discrete backprojection algorithms currently in use (e.g. Huesman et al. [23]).*

It should be noted that, despite our disappointment, because the same (simple) backprojection algorithms were used for both the WVD and the FBP reconstructions in Chapter 5, the comparisons therein are made on equal grounds.

To date, within the wavelet community the method closest in spirit to our WVD-based approach is probably that of Peyrin *et al.* [44]. Their method is based on computing a one-dimensional wavelet transform of the projection data, $Kf(\cdot, \theta)$, at each angle θ, for a fixed resolution. They then show that passing the result through the traditional BFP algorithm yields the equivalent of a two-dimensional wavelet transform of the underlying object, f, with respect to an isotropic wavelet defined in terms of the one-dimensional wavelet (at the same resolution). Hence, although the authors do not comment explicitly on this point, denoising appears to be accomplished in the same manner as the traditional BFP approach, but in the wavelet domain. A decidedly different approach is taken by Bhatia *et al.* [6] using wavelet-based multiscale techniques. These authors also work within the BFP framework, but choose to filter the data prior to computing a one-dimensional discrete orthonormal wavelet transform at each angle. This induces a corresponding change of basis in the final backprojection step, enabling the reconstruction of the entire object or only at a given scale. A key step in their approach is the insertion of a second wavelet transform into the process which essentially diagonalizes the filtering operator in the manner of Beylkin *et al.* [5]. In the presence of noisy data, they regularize their solution by calculating maximum aposteriori probability (MAP) estimates of the object to be backprojected.

As a final note, we move away from the discussion of tomographic reconstruction to mention that much groundwork has been laid in this dissertation for the use of the WVD in other linear inverse problems. Such problems include integration, fractional integration, and certain convolution problems. The one-dimensional forward Meyer wavelet transform algorithm may be adjusted by the addition of a single line to compute the vaguelette coefficients in these problems. We demonstrated the feasibility of such an approach in the context of integration, with noiseless data. Preliminary calculations show that level-dependent thresholds similar to those in Chapter 5 may be derived for the case when the data are observed in the presence of Gaussian noise. A Poisson analogue is also being explored. Finally, future research will include the exploration of WVD-based methods for density estimation in stereology, where the underlying operator may be related to the Abel integral transform, a fractional integral with index $\frac{1}{2}$.

Appendix A

In this appendix we describe the software that has been developed in conjunction with the dissertation. The body of software can be broken down into three categories:

1. Meyer wavelet transforms and applications.

2. Simulation from and generation of phantoms.

3. Wavelet reconstruction from tomographic data.

The majority of the software was written for Matlab. Such Matlab language files have the suffix *.m. S-language files have the suffix *.s, and C programs have filenames ending in *.c .

Each of the three software categories will be described below in separate sections. A section will begin with a brief discussion of the data structures involved, continue with a summary listing of software for that section, and end with a detailed listing of each piece and its utility.

A.1 Meyer Wavelet Transforms and Applications

A.1.1 Data Structures

The software in this section pertains to the calculation of discrete, periodized Meyer wavelet transforms. These include forward and inverse transforms, in both 1 and 2 dimensions. Also included is software for the efficient calculation of certain homogeneous, self-adjoint operators and their biorthogonal-decompositions using the Meyer wavelet. Available operators are those for integration, differentiation, and fractional integration and differentiation (degree between 0 and 1). The efficiency of the operator transform algorithms results from the fact that the action of these operators is most easily characterized in the frequency domain as a simple multiplication, and the fact that the Meyer wavelet transform algorithms are also done in the frequency domain. Similar transforms involving convolution kernels could be added easily.

The one dimensional transforms require data vectors to be of dyadic length i.e., $n = 2^J$ for integer $J > 3$. In two dimensions, the input array must have dyadic (equal) row and column lengths.

The forward Meyer wavelet transform in dimension one takes a data vector, `y`, and returns a vector, `w`, of empirical wavelet coefficients. The user must specify a coarsest resolution level, C, an integer, ≥ 3. The first 2^C entries of `w` correspond to the (coarse level) coefficients of the 'father' Meyer wavelets (or scale functions) at level C and positions $k-1$, for entries $k = 1, \ldots, 2^C$. The (detail level) coefficients of the 'mother' Meyer wavelets at levels $j = C, \ldots, J-1$ follow in a telescoping fashion. Specifically, entry $2^C + 2^{j-C} + k$ of `w` corresponds to the empirical wavelet coefficient of the 'mother' Meyer wavelet at scale j and position $k-1$, for $k = 1, \ldots, 2^j$.

Also specified by the user is the degree of the auxiliary window function for the underlying Meyer wavelets. The window function is defined on $[0, 1]$ to be a polynomial of degree `deg`. The larger `deg`, the smoother the wavelets in the frequency domain.

In two dimensions, the input array, `ymat`, is transformed to an empirical wavelet coefficient array, `wmat`, of equal size. Coefficients are arranged so that the upper left hand corner of the array contains coarse-scale coefficients, and the lower right hand corner contains fine scale coefficients. The coarse coefficients are found in the submatrix $[1:2^C] \times [1:2^C]$. Each detail level has three submatrices of coefficients, corresponding to horizontal, vertical, and diagonal characteristics. The horizontal coefficients of level j are found in the submatrix $[1:2^j] \times [2^j+1:2^{j+1}]$, the vertical in $[2^j+1:2^{j+1}] \times [1:2^j]$, and the diagonal in $[2^j+1:2^{j+1}] \times [2^j+1:2^{j+1}]$.

A.1.2 Summary Listing of Software

Wavelet Transforms	
FwdMeyerTrans.m	1D Forward Meyer transform (periodized)
InvMeyerTrans.m	1D Inverse Meyer transform (periodized)
FwdMeyerTrans2.m	2D Forward Meyer transform (periodized)
InvMeyerTrans2.m	2D Inverse Meyer transform (periodized)
OpFwdMeyerTrans.m	Adapt forward Meyer transform to compute biorthogonal-decompositions of certain operators
OpInvMeyerTrans.m	Adapt inverse Meyer transform to compute operators from wavelet coefficients

Wavelet Transform Utilites	
CoarseMeyerCoefs.m	Computes course coefficients for level C
DetailMeyerCoefs.m	Computes detail coefficients for levels $C \leq j \leq J-2$
FDetailMeyerCoefs.m	Computes detail coefficients for finest level $j = J-1$
CoarseMeyerProj.m	Computes projection of data vector onto $[-2^{C-1}, 2^{C-1}]$ from course coefficients
DetailMeyerProj.m	Computes projection of data vector onto $(-2^j, -2^{j-1}] \cup [2^{j-1}, 2^j)$ from detail coefficients at levels $C \leq j \leq J-2$
FDetailMeyerProj.m	Computes projection of data vector onto $[-2^{J-1}, -2^{J-2}] \cup [2^{J-2}, 2^{J-1}]$ from detail coefficients at level $j = J-1$
MeyerFold.m	Windows and folds data onto a given interval, according to specified polarities.
MeyerUnfold.m	Unfolds and windows data about a given interval, according to specified polarities.
CoefCombine.m	Combines local trigonometric coefficients into wavelet coefficients.
CoefSeparate.m	Separates wavelet coefficients into local sine and cosine coefficients.
ExtendProj.m	Extends projections on positive frequencies to negative frequencies.

DCTI.m	Computes the Type-I Discrete Cosine Transform.
DSTI.m	Computes the Type-I Discrete Sine Transform.
DCTII.m	Computes the Type-II Discrete Cosine Transform.
DSTII.m	Computes the Type-II Discrete Sine Transform.
DCTIII.m	Computes the Type-III Discrete Cosine Transform.
DSTIII.m	Computes the Type-III Discrete Sine Transform.
FwdOpMult.m	Creates the multiplying vector in the frequency domain for OpFwdMeyerTrans.m
InvOpMult.m	Creates the multiplying vector in the frequency domain for OpInvMeyerTrans.m
MeyerNu.m	Meyer auxiliary window function

Wavelet Generation Tools	
FatherMeyerFreq.m	Father Meyer wavelet in frequency domain
MotherMeyerFreq.m	Mother Meyer wavelet in frequency domain
TMotherMeyerFreq.m	Mother Meyer wavelet in frequency domain, truncated at ± 1
MotherGen.m	Mother Meyer wavelet in the time domain
PFatherMeyerWav.m	Periodic father Meyer wavelet in time domain
PMotherMeyerWav.m	Periodic mother Meyer wavelet in time domain
PTMotherMeyerWav.m	Periodic mother Meyer wavelet in time domain, truncated in frequency domain at ± 1

A.1.3 Software Descriptions

FwdMeyerTrans.m

```
Usage:  Computes the ForWarD MEYER wavelet TRANSform of a signal.
        w = FwdMeyerTrans(y,deg,C)
```

```
Inputs:  y   := signal vector of length n = 2^J.
                J an integer > 3.
         deg := degree of nu function in def. of Meyer wavelet
                (integer) Currently 0, 1, 2, and 3 available.
         C   := course level of coefficients.  Must be integer >= 3.
Output:  w   := wavelet coefficient vector of length n = 2^J.
Calls:  CoarseMeyerCoefs.m, DetailMeyerCoefs.m, and
        FDetailMeyerCoefs.m
```

InvMeyerTrans.m

```
Usage:  Computes the INVerse MEYER wavelet TRANSform of coefficients.
        y = InvMeyerTrans(w,deg,C)
Inputs:  w   := vector of wavelet coefficients of length n = 2^J.
                J an integer > 3.
         deg := degree of nu function in def. of Meyer wavelet
                (integer) Currently 0, 1, 2, and 3 available.
         C   := course level of coefficients.  Must be integer >= 3.
Output:  y   := reconstructed signal vector of length n = 2^J.
Calls:  CoarseMeyerProj.m, DetailMeyerProj.m, and
        FDetailMeyerProj.m
```

FwdMeyerTrans2D.m

```
Usage:  Computes the Two Dimensional ForWard MEYER wavelet
        TRANSform of a given image array.
        wmat = FwdMeyerTrans2D(ymat,deg,C)
Inputs: ymat := square image array ( 2^J x 2^J )
                J an integer > 3.
        deg  := degree of nu function in def. of Meyer wavelet
```

```
                (integer) Currently 0, 1, 2, and 3 available.
        C    := course level of coefficients. Must be integer >= 3.
Output: wmat := square matrix of 2D wavelet coefficients (2^J x 2^J)
Calls:  CoarseMeyerCoefs.m, DetailMeyerCoefs.m, and
        FDetailMeyerCoefs.m
```

InvMeyerTrans2D.m

```
Usage:  Computes the two dimensional INVerse MEYER wavelet
        TRANSform of a given coefficient array.
        ymat = InvMeyerTrans2D(wmat,deg,C)
Inputs: wmat := square matrix of 2D wavelet coefficients (2^J x 2^J)
                J an integer > 3.
        deg  := degree of nu function in def. of Meyer wavelet
                (integer) Currently 0, 1, 2, and 3 available.
        C    := course level of coefficients.  Must be integer >= 3.
Output: ymat := reconstructed image array ( 2^J x 2^J )
Calls:  CoarseMeyerProj.m, DetailMeyerProj.m, and
        FDetailMeyerProj.m
```

OpFwdMeyerTrans.m

```
Usage:  Computes the ForWarD MEYER wavelet TRANSform of a signal in
        the case where the Meyer wavelet may have been acted on by
        an OPerator.  Useful for biorthogonal decompositions.
        Ow = OpFwdMeyerTrans(y,deg,C,'type')
Inputs: y     := signal vector of length 2^J.
                 J an integer > 3.
        deg   := degree of nu function in def. of Meyer wavelet
                 (integer) Currently 0, 1, 2, and 3 available.
```

```
         C      := course level of coefficients.  Must be integer >= 3.
        type    := type of operator (string)
                   Currently can try 'iden' (identity), 'deri'
                   (derivative), and 'fder' (fractional derivative).
        expfac := Index of fractional integration or differentiation.
                   Must be a fraction between 0 and 1.
                   Ignored if fractional operator not used.  Enter 0.
Output: Ow    := operator wavelet coefficient vector of length 2^J.
Calls: CoarseMeyerCoefs.m, DetailMeyerCoefs.m, FDetailMeyerCoefs.m,
       and FwdOpMult.m
```

OpInvMeyerTrans.m

```
Usage:  Computes the OPerator INVerse MEYER wavelet TRANSform of
        coefficients.  Calculate Lf from coefficients of f.
        y = OpInvMeyerTrans(w,deg,C,'type',expfac)
Inputs: w    := vector of wavelet coefficients of length 2^J
                 J an integer > 3.
        deg  := degree of nu function in def. of Meyer wavelet
                 (integer) Currently 0, 1, 2, and 3 available.
         C   := course level of coefficients.  Must be integer >= 3.
      type   := type of operator (string)
                 Currently can try 'iden' (identity), 'deri'
                 (derivative), and 'fder' (fractional derivative).
      expfac:= Index of Fractional Integration or Differentiation.
                 Must be fraction between 0 and 1.
                 Ignored if fractional operator not used.  Enter 0.
Output: y    := reconstructed signal vector of length 2^J
Calls: CoarseMeyerProj.m, DetailMeyerProj.m, FDetailMeyerProj.m,
```

```
       and InvOpMult.m
```

CoarseMeyerCoefs.m

```
Usage:  Compute course level coefficients based on the Meyer wavelet,
        for a given level C >= 3.
        beta = CoarseMeyerCoefs(fhat,C,n,deg)
Inputs:  fhat := FFT of signal vector.
         C    := resolution level
         n    := length of fhat (must be of the form 2^J, J an
                 integer > 3)
         deg  := degree of nu function in def. of Meyer wavelet
                 (integer) Currently 0, 1, 2, and 3 available.
Output:  beta := vector of course coefficients for level C
                 ( length(beta) = 2^C )
Calls: MeyerFold.m, DCTI.m, DSTI.m, and CoefCombine.m.
```

DetailMeyerCoefs.m

```
Usage:  Compute detail level coefficients based on the Meyer wavelet,
        for a given level j, 3 <= j <= J-2.   J an integer > 3.
        alpha = DetailMeyerCoefs(fhat,j,n,deg)
Inputs:  fhat := FFT of signal vector.
         j    := resolution level
         n    := length of fhat (must be of the form 2^J)
         deg  := degree of nu function in def. of Meyer wavelet
                 (integer) Currently 0, 1, 2, and 3 available.
Output: alpha := vector of coefficients for level j
                 ( length(alpha) = 2^j )
Calls: MeyerFold.m, DSTIII.m, DCTIII.m, and CoefCombine.m.
```

FDetailMeyerCoefs.m

```
Usage:  Compute detail coefficients based on the Meyer wavelet, for
        finest level j = J-1.   J an integer > 3.
        falpha = FDetailMeyerCoefs(fhat,n,deg)
Inputs:    fhat := FFT of signal vector
           n    := length of fhat (must be of the form 2^J)
           deg  := degree of nu function in def. of Meyer wavelet
                   (integer) Currently 0, 1, 2, and 3 available.
Output:  falpha := vector of coefficients for level J-1
                  ( length(falpha) = 2^(J-1) )
Calls: MeyerFold.m, DSTIII.m, DCTIII.m, and CoefCombine.m.
```

CoarseMeyerProj.m

```
Usage:  Invert the Meyer wavelet transform at the course level C
        for the father Meyer wavelet.  Must have C >= 3.
        cpjf = CoarseMeyerProj(beta,C,n,deg)
Inputs:  beta := vector of coarse wavelet coefficients for
                 level C.   ( length(beta) = 2^C )
         C    := resolution level
         n    := length of full signal (must be of the form 2^J,
                 J an integer > 3)
         deg  := degree of nu function in def. of Meyer wavelet
                 (integer) Currently 0, 1, 2, and 3 available.
Output:  cpjf  := vector of length n containing contribution to
                  the signal (in the frequency domain) from the
                  coarse level of coefficients.
Calls: CoefSeparate.m, DCTI.m, DSTI.m, and MeyerUnfold.m.
```

DetailMeyerProj.m

```
Usage:  Invert the Meyer wavelet transform at a single level
        j of the detail coeffcients, for 3 <= j <= J-2.
        J an integer > 3.
        dpjf = DetailMeyerProj(alpha,j,n,deg)
Inputs:   alpha := vector of detail coefficients for level j
                   ( length(alpha) = 2^j )
          j     := resolution level
          n     := length of full signal (must be of the form 2^J)
          deg   := degree of nu function in def. of Meyer wavelet
                   (integer) Currently 0, 1, 2, and 3 available.
Output:   dpjf   := vector of length n containing contribution to
                    the signal (in the frequency domain) from the
                    jth detail level of coefficients.
Calls: CoefSeparate, DSTII.m, DCTII.m, and MeyerUnfold.m.
```

FDetailMeyerProj.m

```
Usage:  Invert the Meyer wavelet transform at the finest level
        J-1 of the detail coeffcients.   J an integer > 3.
        fdpjf = FDetailMeyerProj(alpha,j,n,deg)
Inputs: alpha := vector of fine detail coefficients
                 ( length(alpha) = 2^(J-1) )
        j     := resolution level (i.e. J-1)
        n     := length of full signal (must be of the form 2^J)
        deg   := degree of nu function in def. of Meyer wavelet
                 (integer) Currently 0, 1, 2, and 3 available.
Output: fdpjf  := vector of length n containing contribution to
```

```
                  the signal (in the frequency domain) from the
                  finest level of detail coefficients.
Calls: CoefSeparate.m, DSTII.m, DCTII.m, and MeyerUnfold.m.
```

MeyerFold.m

```
Usage: Fold a vector onto itself using a specified window.
       fldx = MeyerFold(x, sympts, polarity, window,deg)
Inputs:        x  := signal vector in frequency domain
                     (typically of length n=2^J)
        sympts   := symmetry points, of the form [a,b]
        polarity := polarity with respect to which to do the
                    folding  (string)
                    ' mp  =>  (-,+) '
                    ' pm  =>  (+,-) '
                    ' mm  =>  (-,-) '
                    ' pp  =>  (+,+) '
        window   := (string) indicator for window
                    'm' => Mother Meyer Wavelet Window
                    't' => Truncated Mother Meyer Wavelet Window
                    'f' => Father Meyer Wavelet Window
Output: fldx := vector containing folded version of x onto
                interval between sympts(1) and sympts(2), with
                'polarity' polarities.
                Typically of length 2^j < 2^J .
Calls: MeyerNu.m
```

MeyerUnfold.m

```
Usage: Unfold a vector using a specified window.
```

```
       unfldx = MeyerUnfold(x, sympts, polarity, window,deg)
Inputs:       x  := signal vector in frequency domain
                    (typically of length n=2^j for some j)
        sympts   := symmetry points, of the form [a,b]
        polarity := polarity with respect to which to do the
                    unfolding  (string)
                    ' mp  =>  (-,+) '
                    ' pm  =>  (+,-) '
                    ' mm  =>  (-,-) '
                    ' pp  =>  (+,+) '
        window   := (string) indicator for window
                    'm' => Mother Meyer Wavelet Window
                    't' => Truncated Mother Meyer Wavelet Window
                    'f' => Father Meyer Wavelet Window
Output: ufldx := vector containing unfolded version of x, onto a
                 subset of the integers, about sympts(1) and
                 sympts(2), with 'polarity' polarities.
Calls: MeyerNu.m
```

<u>CoefCombine.m</u>

```
Usage:  Combine local trigonometric coefficients to yield wavelet
        coefficients.
        wcoefs = CoefCombine(rtrigcoefs,itrigcoefs,window,n)
Inputs:  rtrigcoefs := vector of trigonometric coefficients
                       corresponding to projection of real part of
                       \hat f.  Typically of length 2^j, for
                       integer j >= 3.
         itrigcoefs := vector of trigonometric coefficients
```

```
                       corresponding to projection of imag part of
                       \hat f.  Same length as rtrigcoefs.
             window := window with respect to which the projection is
                       done.  'm' -> mother, 'f' -> father, and
                       't' -> truncated mother
                  n := length of original signal f
Output  wcoefs := wavelet coefficients for a given level.
                  Typically of length 2^j or 2^(j+1).
Calls:  None.
```

<u>CoefSeparate.m</u>

```
Usage: Separate wavelet coefficients into local trig coefs.
       [rtrigcoefs, itrigcoefs] = CoefSeparate(wcoefs,window)
Inputs:  wcoefs  := wavelet coefficients, of a given level,
                    to be separated.  Of length 2^j, j an
                    integer >= 3.
         window  := window with respect to which the separating
                    is done.  'm' -> mother, 'f' -> father, and
                    't' -> truncated mother
Output: rtrigcoefs := trigonometric coefficients corresponding to
                      the projection of the real part of \hat f.
                      Typically of length 2^j or 2^(j-1).
        itrigcoefs := trigonometric coefficients corresponding to
                      the projection of the imag part of \hat f.
                      Typically of length 2^j or 2^(j-1).
Calls:  None
```

<u>ExtendProj.m</u>

```
Usage: Extend a projection to all of the integers -n/2+1 -> n/2.
       extproj = ExtendProj(proj,n,window,sympts,sym)
Inputs: proj   := windowed projection vector
           n   := length of full signal
        window := window used in windowing projection
                  'm' -> mother, 'f' -> father
                  't' -> truncated mother
        sympts := points of symmetry and antisymmetry of projection
           sym := symmetry of extension.
                  'e' -> even      'o' -> odd
Output: extproj := extended projection of length n.
Calls:  None.
```

DCTI.m

```
Usage:  Compute the Discrete Cosine Transform, Type I.
        For use in either the forward or inverse Meyer transform.
        dcti = DCTI(x,dir)
Inputs:   x := input signal vector
        dir := directional indicator (string)
               'f' -> Used in forward Meyer wavelet transform
               'i' -> Used in inverse Meyer wavelet transform
Output: dcti := discrete cosine transform, type I, of x
                length(dcti) =  length(x)/2 or length(x)+1,
                depending on direction of transform.
Calls:  None
```

DSTI.m

```
Usage:  Compute the Discrete Sine Transform, Type I.
```

```
        For use in either the forward or inverse Meyer transform.
        dsti = DSTI(x,dir)
Inputs:   x := input signal
        dir := directional indicator (string)
               'f' -> Used in forward Meyer wavelet transform
               'i' -> Used in inverse Meyer wavelet transform
Output: dsti := discrete sine transform, type I, of x
                length(dsti) =  length(x)/2 or length(x),
                depending on direction of transform.
Calls:  None
```

DCTII.m

```
Usage:  Compute the Discrete Cosine Transform, Type II.
        For use in inverting detail levels in Meyer transform
        dctii = DCTII(x)
Inputs:  x := input signal
Output: dctii := discrete cosine transform, type II, of x
                 length(dctii) = length(x).
Calls: None
```

DSTII.m

```
Usage:  Compute the Discrete Sine Transform, Type II.
        For use in inverting detail levels with Meyer transform.
        dstii = DSTII(x)
Inputs:   x := input signal
Output: dstii := discrete sine transform, type II, of x
                 length(dstii) = length(x).
Calls: None
```

DCTIII.m

```
Usage:  Compute the Discrete Cosine Transform, Type III.
        For use in computing detail coefficients in forward
        Meyer transform.
        dctiii = DCTIII(x)
Inputs:   x := input signal
Output: dctiii := discrete cosine transform, type III, of x.
                    length(dctiii) = length(x).
Calls:  None
```

DSTIII.m

```
Usage:  Compute the Discrete Sine Transform, Type III.
        For use in computing detail coefficients in forward
        Meyer transform.
        dstiii = DSTIII(x)
Inputs:   x := input signal
Output: dstiii := discrete sine transform, type III, of x.
                    length(dstiii) = length(x).
Calls:  None
```

FwdOpMult.m

```
Usage:    Creates the multiplying vector of the chosen operator
          in the frequency domain.
          lvec = FwdOpMult('type',expfac,n)
Inputs:   type   := type of operator  (string)
                      Currently can try 'iden' (identity), 'deri'
                      (derivative), and 'fder' (fractional derivative).
```

```
          expfac := exponential factor for frac. int. and diff.
                    Must be a fraction between 0 and 1.
                    Ignored if fractional operator not used.  Enter 0.
          n      := length of signal (must be of the form 2^J).
Outputs:  lvec   := vector of Fourier transform of operator.
                    length( lvec ) = n
Calls:    None
```

InvOpMult.m

```
Usage:    Creates the multiplying vector of the chosen operator
          in the frequency domain.
          lvec = InvOpMult('type',expfac,n)
Inputs:   type   := type of operator (string)
                    Currently can try 'iden' (identity), 'deri'
                    (derivative), and 'fder' (fractional derivative).
          expfac := exponential factor for frac. int. and diff.
                    Must be fraction between 0 and 1.
                    Ignored if fractional operator not used.  Enter 0.
          n      := length of signal (must be of the form 2^J).
Outputs:  lvec   := vector of Fourier transform of operator.
                    length( lvec ) = n
Calls:    None
```

MeyerNu.m

```
Usage: Computes the auxiliary window function or 'Nu' function
       for Meyer wavelets.
       nu = MeyerNu(xi,deg)
Inputs:  xi  := abscissa values for evaluation
```

```
         deg := degree of the polynomial defining Nu on [0,1].
                (integer) Currently 0, 1, 2, and 3 available.
Outputs: nu  := evaluations of Nu() at the points of xi
                length( nu ) = length( xi )
Calls: None
```

FatherMeyerFreq.m

```
Usage:  Matlab function to create the FT of the father Meyer wavelet
        phihat = FatherMeyerFreq(chsi,deg)
Inputs:  chsi  := vector of abscissa values
         deg   := degree of nu function in def. of Meyer wavelet
                  (integer) Currently 0, 1, 2, and 3 available.
Output: phihat := FT of father Meyer wavelet wrt chsi
                  length( phihat ) = length( chsi )
Calls:  MeyerNu.m
```

MotherMeyerFreq.m

```
Usage:  Matlab function to create the FT of the Mother meyer wavelet
        psihat = MotherMeyerFreq(chsi,deg)
Inputs:    chsi := vector of abscissa values
          deg   := degree of nu function in def. of Meyer wavelet
                   (integer) Currently 0, 1, 2, and 3 available.
Output:  psihat := FT of mother Meyer wavelet wrt chsi
                   length( psihat ) = length( chsi )
Calls: MeyerNu.m
```

TMotherMeyerFreq.m

```
Usage:  Matlab function to create the FT of the Mother meyer wavelet,
```

```
        truncated in the frequency domain.
        tpsihat = TMotherMeyerFreq(chsi,deg)
Inputs:     chsi := vector of abscissa values
           deg   := degree of nu function in def. of Meyer wavelet
                    (integer) Currently 0, 1, 2, and 3 available.
Output:  tpsihat := FT of the mother Meyer wavelet, truncated in the
                    frequency domain.
                    length( tpsihat ) = length( chsi )
Calls: MeyerNu.m
```

<u>MotherGen.m</u>

```
Usage: Matlab function to create the 'mother' Meyer wavelet.
       psi = MotherGen(nn,l,deg)
Inputs:  nn := signal length (n = 2^J, for integer J>0)
          l := integer parameter controling range of plot.
               Must have 2^l < nn.
        deg := degree of nu function in def. of Meyer wavelet
               (integer) Currently 0, 1, 2, and 3 available.
Output: psi := 'mother' Meyer wavelet in the time domain
               (length nn)
Calls:  MotherMeyerFreq.m
```

<u>PFatherMeyerWav.m</u>

```
Usage:  Plot the periodized Father Meyer wavelet on [0,1].
        phi = PFatherMeyerWav(n,j,k,deg)
Inputs:  n := length of final wavelet.  (Must have n = 2^J, J>3).
         j := resolution level.  Must have j <= J-2.
         k := location
```

```
       deg := degree of nu function in def. of Meyer wavelet
              (integer) Currently 0, 1, 2, and 3 available.
Output: phi := Periodic father Meyer wavelet on the interval [0,1]
Calls:  FatherMeyerFreq.m
```

PMotherMeyerWav.m

```
Usage:  Plot the periodized Mother Meyer wavelet on [0,1].
        psi = PMotherMeyerWav(n,j,k,deg)
Inputs:   n := length of final wavelet.  (Must have n=2^J, J>3).
          j := resolution level.  Must have j <= J-2.
          k := location
        deg := degree of nu function in def. of Meyer wavelet
                  (integer) Currently 0, 1, 2, and 3 available.
Output: psi := Periodic mother Meyer wavelet on the interval [0,1]
Calls:  MotherMeyerFreq.m
```

PTMotherMeyerWav.m

```
Usage:  Plot the periodized Mother Meyer wavelet on [0,1].
        tpsi = PTMotherMeyerWav(n,j,k,deg)
Inputs:    n := length of final wavelet.  (Must have n=2^J, J>3).
           j := resolution level.  Must have j <= J-2.
           k := location
         deg := degree of nu function in def. of Meyer wavelet
                  (integer) Currently 0, 1, 2, and 3 available.
Output: tpsi := Periodic mother Meyer wavelet on the interval [0,1]
Calls:  TMotherMeyerFreq.m
```

A.2 Simulation From and Generation of Phantoms

A.2.1 Data Structures

A number of programs were developed for the simulation of tomographic data from computer (i.e. non-physical) phantoms. The object to be reconstructed was considered to be a 2D slice from a 3D object. For simplicity, the former was taken to be centered at the origin and contained within the unit circle. Reconstructions took the form of pixel arrays of size $2^J \times 2^J$, for integer $J > 3$, covering the area $[-1,1]^2$ with equisized squares. Each pixel value is interpreted as the value of the image at the center of the corresponding square.

Two types of tomographic data collection were simulated, both from the field of medical imaging. The first corresponds to parallel beam X-ray tomography in which a sequence of equispaced X-ray readings are taken over a series of angles between 0 and π. The second corresponds to Positron Emission Tomography (PET). In PET, the object of interest (typically an organ such as the brain) is injected with a radioactively tagged metabolyte. The object sits within a ring of detectors which register pairs of photon emissions. In our setting, the detector ring is taken to be of radius $\sqrt{2}$ with the number of detectors $N_{det} = 2^{J+1}$, for some integer $J > 3$. This geometry yields a sinogram (histogram of readings) of 2^J equispaced angles and 2^{J-1} non-equispaced bins or tubes which intersect the unit circle.

In order to test and calibrate our methods, we developed software to generate both noisy and noiseless data. The noiseless data in both of the above geometries simply correspond to line integrals of the underlying plane object. In the parallel beam setting, noisy data was generated by adding Gaussian noise to the matrix of line integrals. In the PET setting, noisy data was generated in either of two fashions.

The simpler approach is to generate independent Poisson counts for each tube, with mean the line integral over the central chord of that tube. This corresponds exactly to the distributional assumptions underlying our development of thresholds for denoising PET data with wavelet shrinkage methods. A slightly more realistic method follows Shepp and Logan [49] in which emissions are simulated from the underlying intensity function via rejection sampling. Each emission is then assigned to a particular tube by choosing a random orientation for the line of flight of its photons.

A.2.2 Summary Listing of Software

Phantom Generation	
psphantom_gen.c	Generates point source phantom
slphantom_gen.c	Generates Shepp/Logan phantom

Noiseless Data Calculation	
sltrueproj_pb.c	True projections from Shepp/Logan phantom for parallel beam geometry
sltrueproj_apb.c	True average projections from Shepp/Logan phantom for parallel beam geometry
sltrueproj_ring.c	True projections from Shepp/Logan phantom for ring geometry

Noisy Data Simulation	
SinoAddNoise.m	Simulates additive noise for parallel beam
psprojection.c	Simulates count data from point source
slprojection.c	Simulates count data from Shepp/Logan phantom
poiscount_sim.s	Simulates poisson count data based on line integral means

A.2.3 Software Descriptions

psphantom_gen.c

```
Usage:  Generate a point source phantom of uniform intensity.

Inputs: The following parameters are used.  Set internally.

         nrow    := number of rows

         ncol    := number of columns

         circrad := radius of circular point source

         ints    := intensity of point source

         cntr    := center of point source

Output: nrow x ncol pixel array version of phantom will be

        written to the file 'phantom.out'

Calls:  None
```

slphantom_gen.c

```
Usage: Generate a version of the Shepp/Logan '74 phantom

Inputs: The following parameters are used to generate a phantom

        composed of 10 circles or ellipses.  Set internally.

        maj1-maj10    := major axes lengths

        min1-min10    := minor axes lengths

        ints1-ints10  := ellipse intensities

        ints##'       := intensity at the intersection of ellipses

                         number # and #'

        foci1#,foci2# := first and second foci of ellipse number #

        centr#        := centers of ellipse number # with equal

                         major and minor axes (i.e. circles)

Output: nrow x ncol pixel array version of phantom will be

        written to the file 'phantom.out'
```

```
Calls: None
```

<u>sltrueproj_pb.c</u>

```
Usage: Computes true projection data, or line integrals, of the
       Shepp/Logan phantom for the parallel beam geometry.
       The algorithm interprets ellipses as affine
       transformations of circles.  For each ellipse and line,
       it calculates the points of intersection of the
       corresponding circle and the line, and computes the
       distance between these points in ellipse space.
Inputs: The following input parameters are used.  Set internally.
        nangle        := number of angles at which readings
                           are taken
        nbin          := number of bins per angle at which
                           readings are taken
        ints1-ints10  := ellipse intensities
        ints??        := intensities of any ellipse intersections
        xaxlen,yaxlen := vectors of x and y axes lengths
        xcntr, ycntr  := vectors of x and y coordinates of centers
        cphi, sphi    := vectors of cosine and sine of angles of
                           rotation
Output: nbin x nangle pixel array of true projections will be
        written to the file 'tprojection.out'
Calls:  None
```

<u>sltrueproj_apb.c</u>

```
Usage:  Same as 'sltrueproj_pb.c' above, except that an average
        of 5 line integrals per bin is returned for each
```

```
        projection.
Inputs: Same as 'sltrueproj_pb.c' above.
Output: Same as 'sltrueproj_pb.c' above.
Calls:  None
```

sltrueproj_ring.c

```
Usage:  Computes true projection data, or line integrals, of the
        Shepp/Logan phantom for the ring geometry.  Algorithms
        work the same as in 'sltrueproj_pb.c'.
Inputs: The following input parameters are used.  Set internally.
        ndetect       := number of detectors ( 2^{J+1} , int J > 3)
        nangle        := number of angles  ( = ndetect / 2 )
        nbin          := number of bins per angle which intersect
                         the unit circle   ( = ndetect / 4 )
        ints1-ints10  := ellipse intensities
        ints??        := intensities of any ellipse intersections
        xaxlen,yaxlen := vectors of x and y axes lengths
        xcntr, ycntr  := vectors of x and y coordinates of centers
        cphi, sphi    := vectors of cosine and sine of angles of
                         rotation
Output: nbin x nangle pixel array of true projections will be
        written to the file 'tprojection.out'
```

SinoAddNoise.m

```
Usage: Generates additive white noise for a sinogram
       of true projections.
       noisemat = SinoAddNoise(tprojection,sigma)
Inputs:  tprojection := matrix (nbin x nangle) of true projections
```

```
         sigma       := standard deviation of noise
Output:  noisemat    := matrix (nbin x nangle) of white noise
Calls:  None
```

<u>psprojection.c</u>

```
Usage: Simulates counts data from a point source phantom,
       for ring geometry, following the method of Shepp/Logan '74.
Inputs: The following input parameters are used.  Set internally.
        ncount  := total number of counts to be collected
        ndetect := number of detectors ( 2^{J+1} , int J > 3)
        nangle  := number of angles  ( = ndetect / 2 )
        nbin    := number of bins per angle which intersect
                   the unit circle   ( = ndetect / 4 )
        circrad := radius of circular point source
        ints    := intensity of point source
        cntr    := center of point source
Output: Two output files are written.  An nangle x nangle matrix
        of the histogram of actual emissions is written to
        `histogram.out'.  An nbin x nangle matrix of the sinogram
        of count data is written to `sinogram.out'.
Calls:  ran0.c or any other similar uniform random number generator
```

<u>slprojection.c</u>

```
Usage: Simulates count data from the Shepp/Logan phantom,
       for ring geometry, following the method of Shepp/Logan '74.
Inputs: The following input parameters are used.  Set internally.
        ncount  := total number of counts to be collected
        ndetect := number of detectors ( 2^{J+1} , int J > 3)
```

```
        nangle  := number of angles  ( = ndetect / 2 )
        nbin    := number of bins per angle which intersect
                   the unit circle   ( = ndetect / 4 )
        maj1-maj10    := major axes lengths
        min1-min10    := minor axes lengths
        ints1-ints10  := ellipse intensities
        ints??        := intensities of any ellipse intersections
        foci1?,foci2? := first and second foci of ellipses
        centr?        := centers of ellipses with equal major
                         and minor axes (i.e. circles)
Output: Two output files are written.  An nangle x nangle matrix
        of the histogram of actual emissions is written to
        'histogram.out'.  An nbin x nangle matrix of the sinogram
        of count data is written to 'sinogram.out'.
Calls:  ran0.c or any other similar uniform random number generator
```

<u>poiscount_sim.s</u>

```
Usage:  Simulates count data, for the ring geometry, based on
        Poisson variates with means proportional to the line
        integrals of the central chords of each tube. Commented
        lines allow simulation from a uniform disc phantom.
        Default is to input a matrix of line integrals.
Inputs: The following input parameters are used.   Set internally.
        time      := length of study
        nangle    := number of angles
        nbin      := number of bins per angle which intersect
                     the unit circle
Output: simcounts := nbin x nangle matrix of the sinogram of counts
```

Calls: None

A.3 Wavelet Reconstruction From Tomographic Data

A.3.1 Data Structures

The software in this final section was used to implement our 2D wavelet methods for reconstruction from tomographic data. The primary challenge facing us was the calculation of the empirical wavelet coefficient matrix from indirect data. Working within the framework of Donoho's wavelet-vaguelette decomposition (WVD), a wavelet analogue of the singular value decomposition (SVD), we developed a multiresolution version of the commonly used filtering of backprojections method for this task. The initial step is to backproject the tomographic data, resulting in a blurred version of the image to be recovered. The second step filters the backprojected data across a sequence of frequency bands, yielding the corresponding level of empirical wavelet coefficients at each band or resolution.

When the data are noisy (as opposed to noiseless line integrals), denoising is accomplished by applying level-dependent thresholds in the wavelet domain. These thresholds can be calculated either theoretically or empirically. Theoretical thresholds involve constants which depend on the underlying wavelet basis. Our usage of the Meyer wavelets makes the calculation of these constants fairly straightforward with a standard numerical integration routine.

The final step in our reconstruction is to invert the 2D wavelet coefficient matrix using `InvMeyerTrans2D.m`. We include a simple software tool for viewing the result with Matlab's `image` command, after first scaling in an appropriate fashion.

A.3.2 Summary Listing of Software

Coefficient Calculation	
Backproject_pb2q.m	Backproject projection data obtained from parallel beam geometry
Backproject_ringN.m	Backproject projection data obtained from ring geometry
FBP_FStep.m	Filter backprojected data in the manner of the traditional filtering of backprojected projections approach.
MRFBP_FStep.m	Filter backprojected data across frequency bands, yielding wavelet coefficient matrix

Thresholding	
RadonThrEmp_pb.m	Threshold coefficient matrix based on tomographic data, from parallel beam geometry, using empirical estimates of the noise level.
RadonThrEmp_ring.m	Threshold coefficient matrix based on tomographic data, from ring geometry, using empirical estimates of the noise level.
RadonThrTh_pb.m	Threshold coefficient matrix based on tomographic data, from parallel beam geometry, using theoretical estimates of the noise level.
RadonThrTh_ring.m	Threshold coefficient matrix based on tomographic data, from ring geometry, using theoretical estimates of noise level.
gammal2.c	Calculates wavelet dependent constants for theoretical thresholds.

Utilities	
WavSel.m	Selects appropriate wavelet in 'RadonCoefs.m'
AlterContrast.m	Scales an image to be within the necessary boundaries for Matlab's `image` command.
Trim.m	Trims negative values from a matrix by setting them equal to zero.
Hann_Filt.m	Multiplies a vector by a Hann window.
Hamming_Filt.m	Multiplies a vector by a Hamming window.
Parzen_Filt.m	Multiplies a vector by a Parzen window.
Butter_Filt.m	Multiplies a vector by a Butterworth window.
Rect_Filt.m	Multiplies a vector by a rectangle window.
OrigGauss.m	Generates a Gaussian bump in the square $[-1, 1]^2$
GaussRT.m	Generates the true Radon transform of a Gaussian bump
RadonVagCalc.m	Generates a vaguelette corresponding to the Radon transform operator

A.3.3 Software Descriptions

<u>Backproject_pb2.m</u>

```
Usage:  Computes matrix of backprojections from a matrix of
        projection data.  Simple linear interpolation is used.
        Intended for use with parallel beam geometry, or ring geometry
        after interpolation of the sinogram.
        bp=Backproject_pb2q(proj)
Inputs: proj := matrix of projection data ( 2q+1 x nangle )
                q is an integer power of 2
Output: bp   := matrix of backprojections ( 2q  x 2q  )
Calls:  None
```

Backproject_ringN.m

```
Backproject_ringN.m:  Created 94 - 03 - 30.
Usage:  Computes matrix of backprojections from a matrix of
        projection data.  Simple linear interpolation is used.
        Intended for use with ring geometry only.
        Implements correction terms based on Natterer , '93
        SIAM Applied Mathematics.  Values centered on midpixel.
        bp=Backproject_ringN(proj)
Inputs: proj := matrix of projection data ( 2^(J-1) x 2^J )
                J > 3 an integer
Output: bp   := matrix of backprojections ( 2^J x 2^J )
Calls:  None
```

FBP_FStep.m

```
Usage:  Takes backprojected projection data and calculates the
        filtering step of the traditional filtering of
        backprojected projections reconstruction.
        ymat = FBP_FStep(bp,filter,ffm)
Inputs:      bp := matrix of backprojections  ( 2^J x 2^J )
                   J > 3 an integer.
         filter := (string) filter to be used.  Choose from
                   'hann' (Hann), 'hamm' (Hamming), 'parz'
                   (Parzen), 'butt' (Butterworth), and 'rect'
                   (rectangle, i.e. pure ramp function).
Output:      ymat := matrix of reconstructed image  ( 2^J x 2^J )
Calls:  Hann_Filt.m, Hamming_Filt.m, Parzen_Filt.m,
        Butter_Filt.m, and Rect_Filt.m
```

MRFBP_FStep.m

```
Usage:  Takes backprojected projection data and calculates 2D wavelet
        coefficients.  Performs the "filtering" part of a multi-
        resolution analogue of the filtered backprojections
        reconstruction algorithm for 2D Radon transform projection
        data.
        wmat = MRFBP_FStep(bp,C,deg)
Inputs:   bp := matrix of backprojections  ( 2^J x 2^J )
                J > 3 an integer
          C  := course level of coefficients
         deg := degree of nu function in def. of Meyer wavelet
                (integer) Currently 0, 1, 2, and 3 available.
Output: wmat := matrix of wavelet coefficients  ( 2^J x 2^ J )
Calls: WavSel.m.
```

RadonThrEmp_pb.m

```
Usage: Program to threshold Radon WVD coefs using empirical
       thresholds.  Intended for use with parallel beam geometry.
       twmat = RadonThrEmp_pb(wmat,C)
Inputs:  wmat := matrix of empirical wavelet coefficients
                 (2^J x 2^J), J > 3 an integer
            C := course level of coefficients
                 C >= 3 an integer
Output: twmat := matrix of thresheld coefficients (2^J x 2^J)
Calls: SoftThresh.m or HardThresh.m (Internal to TeachWave)
```

RadonThrEmp_ring.m

```
Usage: Program to threshold Radon WVD coefs using empirical
       thresholds.  Intended for use with ring geometry.
       twmat = RadonThrEmp_ring(wmat,C,ncnts)
Inputs:  wmat := matrix of empirical wavelet coefficients
                 (2^J x 2^J), J > 3 an integer
            C := course level of coefficients
                 C >= 3 an integer
        ncnts := number of observed counts
Output: twmat := matrix of thresheld coefficients (2^J x 2^J)
Calls: SoftThresh.m or HardThresh.m (Internal to TeachWave)
```

RadonThrTh_pb.m

```
Usage: Program to threshold Radon WVD coefs using theoretical
       thresholds.  Intended for parallel beam setup, where noise
       is considered to be additive Gaussian.
       twmat = RadonThrTh_pb(wmat,C,nbin,nangle,sigma)
Inputs:  wmat := matrix of empirical wavelet coefficients
                 size (2^J x 2^J), for integer J > 3
            C := course level of coefficients. Must be integer >= 3.
         nbin := number of bins per angle
       nangle := number of angles
       sigma  := standard deviation of additive noise
Output: twmat := matrix of thresheld coefficients (2^J x 2^J)
Calls: SoftThresh.m or HardThresh.m (Internal to TeachWave)
```

RadonThrTh_ring.m

```
Usage:  Program to threshold Radon WVD coefs using theoretical
        thresholds.  Intended for use with ring geometry.
```

```
        twmat = RadonThrTh_ring(wmat,C,sinogram)
Inputs:  wmat := matrix of empirical wavelet coefficients
                 size (2^J x 2^J), for integer J > 3
            C := course level of coefficients. Must be integer >= 3.
     sinogram := sinogram used to compute empirical
                    wavelet coefficients
Output: twmat := matrix of thresheld coefficients (2^J x 2^J)
Calls: SoftThresh.m or HardThresh.m (Internal to TeachWave)
```

gammacalc.c

```
Usage:  Uses numerical integration to calculate the square of
        the L2 norm for the non-normalized vaguelettes connected
        with the Radon transform.  This wavelet dependent constant
        is used in the theoretical thresholds implemented by
        `RadonThrTh_pb.m' and `RadonThrTh_ring.m'.
Inputs: The following parameters are used.  Set internally.
        pairind := indicator of which pair of wavelets occur in
                   the integrand.
                   pairind = 1    =>   father*mother
                   pairind = 2    =>   father*truncmother
                   pairind = 3    =>   mother*mother
                   pairind = 4    =>   truncmother*truncmother
              p := degree of nu function in def. of Meyer wavelet
                   (integer) Currently 0, 1, 2, and 3 available.
Output: The value of the integral is written to the screen.
Calls: The program `quad3d.c' from Numerical Recipes was used to
       compute the numerical integrals.  This calls `nr.h' and
       `nrutil.c'.  Any similar numerical integration routine may
```

```
       be substituted as descibed in the code.
```

<u>WavSel.m</u>

```
Usage:  Selects and computes the necessary Father or Mother
        wavelet (in the frequency domain) to create a 2D
        wavelet in 'RadonCoefs.m'.
        w = WavSel(chsi,epsilon,pos,deg,findic)
Inputs:  chsi    := vector of input frequencies
         epsilon := indicator of 1,2, or 3 for directions h,v, or d.
         pos     := position in the tensor product (i.e 1 or 2)
            deg  := degree of nu function in def. of Meyer wavelet
                    (integer) Currently 0, 1, 2, and 3 available.
         findic  := indicator for fine level of coefficients. (0 or 1)
Output:  w := values of the selected wavelet, calculated at chsi
Calls: FatherMeyerFreq.m, MotherMeyerFreq.m, and
       TMotherMeyerFreq.m
```

<u>AlterContrast.m</u>

```
Usage:  Scales pixel values of a matrix to the range 1 - 64
        for use with Matlab's `image' command.
        AlterContrast(injunk,'trimming')
Inputs: injunk   := matrix of values to be scaled and imaged
        trimming := indicator ('y' / 'n') as whether or not
                    negative values should be set to zero prior
                    to scaling.
Output: image of scaled matrix is displayed on screen
Call:  Trim.m
```

Trim.m

```
Usage: Trims a matrix so that all negative values are set to zero.
       tymat = Trim(ymat)
Inputs: ymat := matrix of values
Output: tymat := trimmed version of 'ymat'
Calls:  None
```

Hann_Filt.m

```
Usage:  Multiplies a vector by a Hann window.
        filty = Hann_Filt(y,ffm)
Inputs:    y  := input vector
          ffm := frequency cutoff
Output: filty := windowed version of input vector.
                 length( filty ) = length( y )
Calls:  None
```

Hamming_Filt.m

```
Usage:  Multiplies a vector by a Hamming window.
        filty = Hamming_Filt(y,ffm)
Inputs:    y  := input vector
          ffm := frequency cutoff
Output: filty := windowed version of input vector.
                 length( filty ) = length( y )
Calls:  None
```

Parzen_Filt.m

```
Usage:  Multiplies a vector by a Parzen window.
```

```
        filty = Parzen_Filt(y,ffm)

Inputs:    y  := input vector
          ffm := frequency cutoff

Output: filty := windowed version of input vector.
                 length( filty ) = length( y )

Calls:  None
```

Butter_Filt.m

```
Usage:  Multiplies a vector by a Butterworth window.
        filty = Butter_Filt(y,ffm)

Inputs:    y  := input vector
          ffm := frequency cutoff

Output: filty := windowed version of input vector.
                 length( filty ) = length( y )

Calls:  None
```

Rect_Filt.m

```
Usage:  Multiplies a vector by a rectangle window.
        filtdat = Rect_Filt(dat)

Inputs:    y  := input vector
          ffm := frequency cutoff

Output: filty := windowed version of input vector.
                 length( filty ) = length( y )

Calls:  None
```

OrigGauss.m

```
Usage:  Generate a Gaussian bump on the square [-1,1]^2.
        ogmat = OrigGauss(q,xshift,yshift,sigma)
```

```
Inputs:      q := parameter for size of grid (2qx2q)
        xshift := mean in the x-direction
        yshift := mean in the y-direction
        sigma  := standard deviation (i.e. symmetric bump)
Output: ogmat := (2qx2q) matrix of Gaussian bump evaluations.
Calls:  None.
```

<u>GaussRT.m</u>

```
Usage:  Generate the Radon transform of a Gaussian bump.
        gaussmat=GaussRT(q,xshift,yshift,sigma)
Inputs:      q := parameter for # of bins.
         xshift:= mean of the Gaussian in the x-direction
         yshift:= mean of the Gaussian in the y-direction
         sigma:= standard deviation.  On the order of 2/3(q) for a
                 good approx. of a point source.
Output: gaussmat:= (2q+1)xfloor(pi*q) sinogram matrix containing
                   the exact Radon transform of a Gaussian with
                   mean (xshift,yshift) and standard deviation sigma.
Calls:  None.
```

<u>RadonVagCalc.m</u>

```
Usage:  Calculate vaguelettes corresponding to the Radon transform,
        with respect to Meyer wavelets.
        yy=RadonVagCalc(nn,xx,jj,kx,ky,epsilon)
Inputs:  nn := length of a strip in the u direction (nn=2^J)
         xx := step size parameter
               xx == nn/2  =>  u in [-1,1]
               xx == nn/4  =>  u in [-2,2]
```

```
            xx == nn/8  =>  u in [-4,4]   etc.
      jj :=  Resolution level  (jj >= 3)
      kx :=  Positional index in the x-direction
      ky :=  Positional index in the y-direction
      epsilon := Directional index
                 0  => father*father  basis function
                 1  => father*mother b.f. (i.e. horizontal)
                 2  => mother*father b.f. (i.e. vertical  )
                 3  => mother*mother b.f. (i.e. diagonal  )
Output:  yy  :=  Vaguelette on an nn x nn (u,theta) grid.
                 Theta in [0,pi)
Calls:  FatherMeyerFreq.m,  MotherMeyerFreq.m
```

Appendix B

In this appendix we list and briefly describe the programs used to generate the figures shown in this thesis. At present they may be found in the directory

`/u0/eric/wavelets/scripts`

In the long run, they will become part of TeachWave0.700 and later versions, accessible via anonymous ftp to `playfair.stanford.edu` in the directory

`pub/software/wavelets,`

or via xmosaic to

`http://playfair.stanford.edu`

where the option `software` should be selected.

After installation in TeachWave, users will be able to recreate the figures shown in this thesis by typing the MATLAB command `EKThesisDemo` and then pointing at and clicking on the relevant buttons in the menu that appears. The scripts used to create this demo will be found in

`TeachWave/scripts/KolaczykThesis.`

Figure 2.1: WaveletEx.m

Uses the TeachWave tool `MakeWavelet` to generate selected father/mother wavelet pairs. Many other possible pairs are available.

Figure 2.2: BumpsWT.m

Creates the signal 'Bumps' using the TeachWave tool `MakeSignal`. 'Bumps' is periodized by setting the coarse level coefficients to zero. Wavelet coefficients computed using `FwdMeyerTrans`.

Figure 2.3: Hand drawn using 'xfig'.

Figure 2.4: DJEx.m

The periodized signal 'Bumps' was computed as in Figure 2.2, and the Daubechies6 wavelet transform was taken using the TeachWave tool `FWT_PO`, with the appropriate filter. White noise was added to the signal using `WhiteNoise`. The noisy signal was then denoised in the wavelet domain using `MultiVisu` to perform wavelet shrinkage. Reconstructed signal results from applying `IWT_PO` to the thresheld coefficients.

Figure 3.1: MeyerWavePlots.m

The auxiliary function, $\nu(\cdot)$, was calculated using `MeyerNu`. The frequency domain representation of the 'mother' Meyer wavelet was obtained by plotting the modulus of the result of `MotherMeyerFreq`. Applying the inverse (fast) Fourier transform yields the time domain representation.

Figure 3.2: PMotherMeyerPlot.m

These periodized Meyer wavelets may be obtained by applying `InvMeyerTrans` to a vector of zeros, with a '1' at the coefficient location corresponding to the desired wavelet.

Figure 3.3: MeyerWavePlots.m

The frequency domain representation of the 'father' Meyer wavelet was obtained by plotting the modulus of the result of `FatherMeyerFreq`. Applying the inverse (fast) Fourier transform yields the time domain representation.

Figure 3.4: PFatherMeyerPlot.m

These periodized Meyer wavelets may be obtained by applying `InvMeyerTrans` to a vector of zeros, with a '1' at the coefficient location corresponding to the desired wavelet.

Figure 4.1: MeyerDecayPlot.m

'Mother' Meyer wavelets were first generated using `MotherGen`, with degrees $0, 1, 2$, and 3 for the auxiliary function. The logarithms of the absolute values of the four wavelets were then plotted for $x \geq \frac{1}{2}$, since they are symmetric and take their maximum at $x = \frac{1}{2}$.

Figure 4.2: BoxcarCompare.m

Boxcar function was created by filling the middle 50% of a vector of zeros with ones. The Meyer wavelet transform was computed using `FwdMeyerTrans`, and the Daubechies6 wavelet transform, using TeachWave's `FWT_PO` with the appropriate filter. Coefficient plots were made using TeachWave's `PlotWaveCoeff`.

Figure 4.3: FracDerivPlot.m

Fractional derivatives of the Meyer wavelet were computed with `OpInvMeyerTrans`, using the `'fder'` option, on a vector of zeros with one '1' at the coefficient corresponding to the wavelet to be differentiated.

Figure 4.4: DerivofIntBumps.m

The discrete, periodized Meyer wavelet transform of the periodized signal 'Bumps' was created as in Figure 2.2. `OpInvMeyerTrans` was used, with the `'intg'` option, to compute the integral of the periodized 'Bumps'. This was then input to the tool `OpFwdmeyerTrans`, with the `'deri'` option, and inverted using `InvMeyerTrans` to yield the derivative of the integral, or the original periodized signal.

Figure 5.1: RadonVagueGen.m

These (and other) vaguelette functions corresponding to the Radon transform may be calculated using the tool `RadonVagueCalc`. Plots created using Matlab's `mesh` command on the appropriate grid.

<u>Figure 5.2</u>: Hand drawn using 'xfig'

<u>Figure 5.3</u>: BPTest.m

Test of the backprojection algorithm `Backproject_pb2q`. The Radon transform of a Gaussian bump was calculated using `GaussRT`, then backprojected and normalized by $\sqrt{x^2+y^2}$.

<u>Figure 5.4</u>: WVCoefCompare.m

The backprojection of the Radon transform of a Gaussian bump was kept from Figure 5.3. This matrix was input to `MRFBP_FStep` to calculate the corresponding vaguelette coefficients. These were then compared with the actual wavelet coefficients of the original Gaussian, which were computed by applying the tool `FwdMeyerTrans2D` to the result of `OrigGauss` (with the appropriate parameters).

<u>Figure 5.5</u>: GaussReconTest.m

The two original Gaussian bumps were computed using `OrigGauss`. The two reconstructions were computed by following the steps of Figures 5.3 and 5.4 (with different parameter settings) and then inverting the resulting coefficient matrices with `InvMeyerTrans2D`.

<u>Figure 5.6</u>: phantom_gen.c and trueproj_apb.c

A matrix of intensities was computed for the Shepp/Logan phantom using the C code `phantom_gen.c`. The associated projection data, for the parallel beam geometry, was computed using the C code `trueproj_apb.c`. See Appendix A for descriptions of these files.

<u>Figure 5.7</u>: WTPhantom.m

The Meyer wavelet transform of the Shepp/Logan phantom was obtained by applying `FwdMeyerTrans2D` to the phantom calculated in Figure 5.6. The enhanced version of the image plot of these coefficients results from inflating the coefficient matrix to the extent that only the smallest values (corresponding to the edges) are visible on a 64bit image map.

Figure 5.8: PBExample.txt

The file `PBExample.txt` outlines a series of six steps required to compute and display the reconstructions in this and Figure 5.10, using a combination of C code and Matlab tools.

Figure 5.9: Hand drawn using 'xfig'

Figure 5.10: PBExample.txt

The file `PBExample.txt` outlines a series of six steps required to compute and display the reconstructions in this and Figure 5.8, using a combination of C code and Matlab tools.

Figure 5.11: PBErrorFreqComp.m

The two-dimensional Fourier transforms of the error surface from the WVD and FBP reconstructions were computed (20dB). A slice of the modulus, corresponding to frequency 0 in the y-direction, was taken, and plotted minus its DC component. Only half the spectrum is necessary.

Figure 5.12: PBReconZoom.m

An approximately 55×70 submatrix is extracted from the WVD and FBP reconstructions, as well as the original phantom, and displayed using Matlab's 'image' command.

Figure 5.13: PETExample.txt

The file `PETExample.txt` outlines a series of five steps required to compute and display the reconstructions, using a combination of C code and Matlab tools.

Figure 5.14: PETErrorFreqComp.m

The two-dimensional Fourier transforms of the error surface from the WVD and FBP reconstructions were computed (5 million counts). A slice of the modulus, corresponding to frequency 0 in the y-direction, was taken, and plotted minus its DC component. Only half the spectrum is necessary.

Figure 5.15: PETReconZoom.m

An approximately 55×70 submatrix is extracted from the WVD and FBP reconstructions, as well as the original phantom, and displayed using Matlab's 'image' command.

Appendix C

We comment briefly on issues related to translating certain software tools defined in this thesis into working code.

In Chapter 3, we created a set of tools which implemented key concepts of Auscher, Weiss, and Wickerhauser [1] for use with our Meyer wavelet transform algorithms. We chose not to write these tools in the format of a standard programming language, but rather in a pseudo-code manner, imitating Knuth's WEB language [25], so as to improve readability. However, implementation of these tools in a standard programming language may be accomplished in a fairly straightforward manner, as we have done using the Matlab language.

For example, the tool `EXTEND` was defined in Chapter 3 as

---

define `extproj` ≡ `EXTEND( proj, [a,b], (e,e'), type )`

{*Extend Projection to Negative Values* }

Variables of function `EXTEND` ≡

`proj` := *projection onto [a,b]. Vector of length card*$([\mathtt{a}-\mathtt{e},\mathtt{b}+\mathtt{e}']\cap \boldsymbol{Z})$.

`[a,b], (e,e')` := *as defined in* `WINDOW`

`type` := *(string) type of extension, 'even' or 'odd'*

`extproj` := *extended projection. Vector of length* $2\times$ *length(*`proj`*)*

This tool takes the projection of a data vector onto the discrete analogue of an interval $I \subset I\!R^+$ and extends it the union $-I \cup I$, where $-I$ is the reflection of I about zero.

The tool `EXTEND` is implemented in Matlab through the function `ExtendProj` (see Appendix A). Of the inputs to `EXTEND`, `proj` and `type` have direct equivalents in `ExtendProj`. However, the inputs `[a,b]` and `(e,e')` are only representative. The first is replaced by a 2×1 vector containing the endpoints `a` and `b` of the projection interval. The second is replaced by a string variable indicating the window associated with the projection. This string variable dictates which part of an 'if-elseif' statement is accessed, and the corresponding window overlap values `e` and `e'` are assigned there. In addition to these differences, the Matlab function `ExtendProj` also takes an extra integer argument, `n`, giving the length of the original signal vector. This is used in zero-padding the vectors of projections to vectors with length `n`, so that when summed, the original signal is recovered. Therefore, although there is no extension necessary for the coarse level projection, this projection is still input to `ExtendProj` to be zero-padded in the appropriate manner.

Finally, the reader should note that the tools `WINDOW` and `FOLD`, as used in the FMWT, are combined in the one Matlab function `MeyerFold`. Similarly, the tools `UNFOLD` and `WINDOW`, as used in the IMWT, are combined in `MeyerUnfold`.

Bibliography

[1] P. Auscher, G. Weiss, and M. Wickerhauser (1992) Local Sine and Cosine Bases of Coifman and Meyer and the Construction of Smooth Wavelets. In *Wavelets: A tutorial in theory and applications*, C.K. Chui (ed.), pp. 237-256. Academic Press, Inc.

[2] G. Bennett (1962) Probability Inequalities for the Sum of Independent Random Variables. *Journal of the American Statistical Association*, **57**, 33-45.

[3] C. Berenstein and D. Walnut (1993) Local Inversion of the Radon Transform in Even Dimensions Using Wavelets. Technical Report CAM-21/93, Center For The Applications of Mathematics, George Mason University.

[4] Bertero, M. (1989) Linear Inverse and Ill-Posed Problems. In *Advances in Electronics and Electron Physics*. Academic Press, New York.

[5] G. Beylkin, R. Coifman, and V. Rokhlin (1991) Fast Wavelet Transforms and Numerical Algorithms I. *Communications on Pure and Applied Mathematics*, Vol. XLIV, pp. 141-183.

[6] M. Bhatia, W.C. Karl, and A.S. Willsky (1993) A Wavelet-Based Method for Multiscale Tomographic Reconstruction. Technical Report LIDS-P-2182, Stochastic Systems Group, MIT.

[7] F. Biétrix, B. Lövström, and G. Salomonsson (1992) Enhancement of Ultrasound Images By Multiscale Analysis. *IEEE-SP Symposium on Time-Frequency and Time-Scale Analysis*, IEEE Press, Piscataway, New Jersey, 223-226.

[8] R.R. Coifman and Y. Meyer (1991) Remarques sur l'analyse de Fourier à fenêtre, série I, *C.R. Acad. Sci. Paris*, **312**, 259-261.

[9] I. Daubechies (1993) Two recent developments: Wavelets on the interval and biorthogonal decompositions of the differentiation operator, in "Recent Advances in Wavelet Analysis" (L. Schumaker and G. Webb, Eds.), Academic Press, New York.

[10] I. Daubechies (1992) *Ten Lectures on Wavelets*, SIAM, Philadelphia, Pennsylvania.

[11] S.R. Deans (1983) *The Radon Transform and Some of Its Applications*, John Wiley & Sons, Inc.

[12] J. DeStefano and T. Olson (1992) Wavelet Localization of the Radon Transform in Even Dimensions. *IEEE-SP Symposium on Time-Frequency and Time-Scale Analysis*, IEEE Press, Piscataway, New Jersey, 137-140.

[13] D.L. Donoho (1994) In conversation with the author.

[14] D.L. Donoho (1992) Nonlinear solution of linear inverse problems by Wavelet-Vaguelette Decomposition. Technical Report No. 403, Dept. of Statistics, Stanford University.

[15] D.L. Donoho, I.M. Johnstone, G. Kerkyacharian, and D. Picard (1995) Wavelet Shrinkage: Asymptopia ?, *Journal of the Royal Statistical Society, Series B*, **57**, 000-000.

[16] D.L. Donoho, I.M. Johnstone, G. Kerkyacharian, and D. Picard (1993) Density estimation by wavelet thresholding. Technical Report 426, Department of Statistics, Stanford University.

[17] D.L. Donoho and I.M. Johnstone (1995) Adapting to unknown smoothness via wavelet shrinkage. *Journal of the American Statistical Association*, **90**, ??-??.

[18] D.L. Donoho and I.M. Johnstone (1994) Ideal spatial adaptation via wavelet shrinkage. *Biometrica*, to appear.

[19] D.L. Donoho and I.M. Johnstone (1994) Neo-classical minimax problems, thresholding, and adaptation. Technical Report 446, Department of Statistics, Stanford University.

[20] D.L. Donoho and I.M. Johnstone (1995) Minimax estimation via wavelet shrinkage. *Annals of Statistics*, to appear.

[21] N.P. Galatsanos, A.K. Kataggelos, R.T. Chin, and A. Hillery (1991) Least squares restoration of multi-channel images. *IEEE Transactions on Signal Processing*, **39**, 2222-2236.

[22] H. Gao (1993) *Spectral Density Estimation by Using Wavelets.* Ph.D. Thesis, University of California, Berkeley.

[23] R.H. Huesman, G.T. Gullberg, W.L. Greenberg, and T.F. Budinger (1977) RECLBL Library Users Manuel - Donner Algorithms for Reconstruction Tomography. Technical Report PUB214, Lawrence Berkeley Laboratory, Berkeley, California.

[24] B. Jawerth and W. Sweldens (1993) Wavelet multiresolution analyses adapted for the fast solution of boundary value ordinary differential equations. In *Sixth*

Copper Mountain Conference on Multigrid Methods, N.D. Melson, T.A. Manteuffel, and S.F. McCormick (eds), pp. 259-273. NASA Conference Publication 3224.

[25] D. E. Knuth (1983) The Web System of Structured Documentation. Stanford University, Dept. of Computer Science, TR No. Stan-CS-83-980.

[26] M.R. Leadbetter, G. Lindgren, and H. Rootzén (1983) *Extremes and Related Properties of Random Sequences and Processes.* Springer-Verlag, New York.

[27] P.G. Lemarié-Rieusset (1992) Analyses multi-resolutions non orthogonales, commutation entre projecteurs et derivation et ondelettes vecteurs à divergence nulle. *Revista Mat. Ibero-Americana* **8**, 221-238.

[28] J.H. Letcher (1992) An Imaging Device that Uses the Wavelet Transform as the Image Reconstruction Algorithm. *International Journal of Imaging Systems and Technology*, **4**, 98-108.

[29] M. Luettgen (1993) *Image Processing with Multiscale Stochastic Models.* Ph.D. Thesis, Laboratory for Information and Decision Systems, MIT.

[30] S. Mallat (1989) A theory for multiresolution signal decomposition: The wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **11**, 674-693.

[31] S. Mallat (1989) Multiresolution approximation signal decomposition. Transactions of the American Mathematical Society, **315**, pp. 69-88.

[32] H. Malvar (1990) Lapped transforms for efficient transform/subband coding. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **38**, 969-978.

[33] Y. Meyer (1985) Principe d'incertitude, bases hilbertiennes, et algèbres d'opérateurs. Séminaire Bourbaki, 1985-86, no. 662.

[34] Y. Meyer (1986) *Ondelettes, fonctions splines et analyses graduées.* Lectures given at the University of Torino, Italy.

[35] Y. Meyer (1992) *Wavelets and Operators.* Cambridge University Press, England. (Translated by D.H. Salinger.)

[36] Y. Meyer (1993) *Wavelets: Algorithms and Applications.* SIAM, Philadelphia, Pennsylvania. (Translated by R. D. Ryan.)

[37] E. Miller and A. Willsky (1992) A Multi-Scale Approach to Solving One Dimensional Inverse Problems. *IEEE-SP Symposium on Time-Frequency and Time-Scale Analysis*, IEEE Press, Piscataway, New Jersey, 129-132.

[38] G.P Nason (1994) Wavelet regression by cross-validation. Technical Report 447, Department of Statistics, Stanford University.

[39] F. Natterer (1993) Sampling in Fan Beam Tomography. *SIAM Journal of Applied Mathematics*, **53**, 358-380.

[40] D. Nychka *et al.* (1984) Cross-Validated Spline Methods for the Estimation of Three-Dimensional Tumor Size Distributions From Observations on Two-Dimensional Cross Sections. *Journal of the American Statistical Association*, **79**, 832-846.

[41] T. Olson (1992) Limited Angle Tomography Via Multiresolution Analysis and Oversampling. *IEEE-SP Symposium on Time-Frequency and Time-Scale Analysis*, IEEE Press, Piscataway, New Jersey, 215-218.

[42] F. O'Sullivan (1986) A Statistical Perspective on Ill-Posed Inverse Problems. *Statistical Science*, **1**, 502-527.

[43] S.E. Petersen, P.T. Fox, A.Z. Snider, and M.E. Raichle (1990) Activation of extrastriate and frontal cortical areas by visual words and word-like stimulii. *Science*, 1041-1044.

[44] F. Peyrin, M. Zaim, and R. Goutte (1992) Multiscale Reconstruction of Tomographic Images. *IEEE-SP Symposium on Time-Frequency and Time-Scale Analysis*, IEEE Press, Piscataway, New Jersey, 219-222.

[45] K.R. Rao and P. Yip (1990) *Discrete Cosine Transform: algorithms, advantages, applications*, Academic Press, Inc., Boston.

[46] W.B. Richardson, Jr (1992) Nonlinear filtering and multiscale texture discrimination for mammograms. *SPIE Vol. 1768 Mathematical Methods in Medical Imaging*, 293-305.

[47] B. Sahiner and A.E. Yagle (1993) Image Reconstruction From Projections Under Wavelet Constraints. *IEEE Transactions on Signal Processing*, **41**, 3579-3584.

[48] B. Sahiner and A.E. Yagle (1993) Limited Angle Tomography Using the Wavelet Transform. Preprint.

[49] L.A. Shepp and B.F. Logan (1974) The Fourier reconstruction of a head section. *IEEE Transactions on Nuclear Science*, **NS21**(3), 21-43.

[50] L.R. Squire, J.G. Ojemann, F.M. Miezin, S.E. Petersen, T.O. Videen, and M.E. Raichle (1992) Activation of the hippocampus in normal humans: A functional anatomical study of memory. *Proceedings of the National Academy of Sciences*, **89**, 1837-1841.

[51] J. Starck and A. Bijaoui (1994) Filtering and Deconvolution by the Wavelet Transform. *IEEE Transactions on Signal Processing*, **35**, 3-??.

[52] J. Starck and A. Bijaoui (1993) Multiresolution Deconvolution. To appear in *JOSA A* .

[53] J. Starck, A. Bijaoui, B. Lopez, and C. Perrier (199?) Image Reconstruction by the Wavelet Transform Applied to Aperture Synthesis. To appear in *Astronomy and Astrophysics.*

[54] J. Starck and F. Murtagh (199?) Image Restoration with Noise Suppression Using the Wavelet Transform. Submitted to *Astronomy and Astrophysics.*

[55] W. Sweldens (1994) Construction and Applications of Wavelets in Numerical Analysis. Ph.D. Thesis. Katholieke Universiteit Leuven, Belgium.

[56] Y. Vardi, L.A. Shepp, and L. Kaufman (1985) A Statistical Model for Positron Emission Tomography. *Journal of the American Statistical Association*, **80**, 8-20.

[57] G. Wahba (1990) *Spline Models for Observational Data.* NSF-CBMS Regional Conference Series in Mathematics, SIAM, Philadelphia, Pennsylvania.

[58] Y. Wang (1994) Function Estimation via Wavelets for Data with Long-Range Dependence. Technical Report, Department of Statistics, University of Missouri-Columbia.

[59] M.V. Wickerhauser (1993) Smooth localized orthonormal bases. *Comptes Rendus de l'Académie des Sciences de Paris*, 316:423-427.

[60] M.V. Wickerhauser (1994) *Adapted Wavelet Analysis from theory to software.* Draft manuscript.