SEQUENTIAL AND LOCALIZED IMPLICIT WAVELET-BASED SOLVERS FOR STIFF PARTIAL DIFFERENTIAL EQUATIONS

By
Donald Alexander McLaren
April 2012

A Thesis
submitted to the Faculty of Graduate and Postdoctoral Studies
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy in Mathematics

Abstract

This thesis explains and tests a wavelet based implicit numerical method for the solving of partial differential equations. Intended for problems with localized small-scale interactions, the method exploits the form of the wavelet decomposition to divide the implicit system created by the time discretization into multiple, smaller, systems that can be solved sequentially. Included are tests of this method on linear and non-linear problems, with both its results and the time required to calculate them compared to basic models. It was found that the method requires less computational effort than the high resolution control results. Furthermore, the method showed convergence towards high resolution control results.

Acknowledgements

I first need to thank Dr. Rémi Vaillancourt, from the University of Ottawa, and Dr. Lucy Campbell, from Carleton University, my faculty advisors. It was Dr. Vaillancourt's suggestion that started me on wavelets and Dr. Campbell's suggested test problem that gave me the idea for the new method. I would especially like to thank them for their patience while convincing me to explain myself, and my ideas, clearly.

Little is possible without funding. I need to thank Dr. Vaillancourt, NSERC, and the University of Ottawa for keeping me fed during this process.

Dedication

Thank you Rhea for putting up with my periods of 'this doesn't work' anguish and 'this should work' anger during the seemingly endless programming phases. I have never stopped being less than ecstatic that we are married.

Thank you Dr. Mara Lee McLaren, a.k.a. 'mom,' for constantly reminding me to 'finish your damn thesis.'

Contents

List of Figures

Glossary

Symbols

- $A^{\perp}$ is the orthogonal complement of $A \subseteq V$, so the set $\{x \in V | \langle x, y \rangle = 0 \, \forall \, y \in A\}$. We will frequently using spaces $B \bigcap A^{\perp}$, the orthogonal complement of A in B.
- $f * g$ is the convolution of f with g, Definition A.1.2. We will need one particular property of the convolution, given right after the definition.
- ψ, a wavelet. Definition 2.1.1 and the main subject of Chapter 2.
- ψ_H, the Haar wavelet. Discussed first in Example 2.1.4.
- ϕ, a scaling function. Always paired with a wavelet. The subject of Section 2.2 and Definition 2.2.1.
- ϕ_H, the Haar scaling function, Example 2.3.6.
- $\mathbf{1}_A$, the indicator function of the set A. We get $\mathbf{1}_A(x) = 0$ if $x \notin A$ and $\mathbf{1}_A(x) = 1$ if $x \in A$.
- $\mathcal{S} = \left\{ f \in C^{\infty}(\mathbb{R}) | \text{ for all } m, n \geqslant 0 \in \mathbb{Z}, \, |x|^n \left| \frac{d^m}{dx^m} f(x) \right| \text{ is bounded} \right\}$, referred to as the Schwartz class functions or Schwartz space. The space C^{∞} is that of infinitely differentiable functions. The definition of $\mathcal{S}$ takes that one step further, requiring that any derivative of the function converge to zero at infinity faster than any power of x (note that this includes negative powers). The Schwartz space is most useful when defining the Fourier transform.

- l^2 a space composed of infinite sequences, specifically those of the form $\{a_n\}_{n\in\mathbb{N}}$ such that $\sum_k |a_k|^2$ is finite. The inner product is $\langle a, b\rangle = \sum_{n\in\mathbb{N}} a_n\overline{b_n}$.

- $L^2(\Omega)$, a standard space, composed of (measurable) functions f on Ω such that $\int_\Omega |f|^2\, dx < \infty$. Typically, the domain will be $\Omega = \mathbb{R}$, though we will, occasionally, use $[0, 2\pi)$. The inner product is $\langle f, g\rangle = \int_\Omega f(x)\,\overline{g(x)}\, dx$, and the norm is $\|f\| = \sqrt{\int_\Omega |f(x)|^2\, dx}$. Most functions we look at will be in this space. The concept of density for L^2 comes up fairly often. If the subset $H \subset L^2$ is dense in L^2 then for all $f \in L^2$ there exists a sequence $\{g_k\}_{k\in\mathbb{N}}$ in H such that $\lim_{k\to\infty} \|f - g_k\| = 0$. This means that the closure of H will be equal to L^2.

- L_0^2 is the space of 2π periodic functions. The inner product is $\langle f, g\rangle = \int_0^{2\pi} f(\xi)\,\overline{g(\xi)}\, d\xi$. Usually, the functions are complex valued and written using Fourier series.

- $P_V f$, the projection of the function f onto the closed subspace V. We will use $P_V f$ as equal to the element in V that is closest to f. As a result, we can define it $P_V f = \{g \in V |\, \|f - g\| \leqslant \|f - h\|\, \forall\, h \in V\}$. The projection onto a subspace V with orthonormal basis vectors $\{g_k\}_{k\in K}$ is a special case, being merely $P_V f = \sum_k \langle f, g_k\rangle\, g_k$. We will make frequent use of this property of orthonormal bases.

- $\|f\|$, the norm of the element f in a given space. We will mostly use the L^2 norm, $\|f\|_2 = \sqrt{\langle f, f\rangle}$. In a space with an inner product, $\|f\|^2 = \langle f, f\rangle$.

- $\widehat{f}$ and $\check{f}$ are the continuous Fourier transforms, for functions $f \in L^1 \bigcap L^2$. The two transforms are defined in Section A.1.

- $\widetilde{f}$, the dual of f, usually applied using sets of functions. The basic setup has $\{f_k\}_{k\in K}$ and $\left\{\widetilde{f}_k\right\}_{k\in K}$, such that $\left\langle f_l, \widetilde{f}_k\right\rangle = \delta_{lk}$.

- δ_{jk}, the Kronecker delta function. It is defined on $\mathbb{Z}\times\mathbb{Z}$ and is equal to zero when $j \neq k$ and to one when $j = k$.

- $\langle f, g\rangle$, the inner product of f and g. The inner products we will use the most are that from L_2 and l_2, applied to functions and infinite sequences, respectively.

- $\langle f(\cdot - k), g\rangle$, the inner product of the function $f(x-k)$ with the function $g(x)$. In the case of L^2, this will be $\langle f(\cdot - k), g\rangle = \int f(x-k)g(x)\,dx$. This is done to clarify cases where one of the functions has its variable modified in some way (from simply x to $2^j x - k$, for instance). The '$\cdot$' merely indicates the variable used when calculating the inner product.

Acronyms, Single Letters

- FFT, the Fast Fourier Transform, a highly efficient algorithm that converts a set of equally spaced values of a function into a set of that function's Fourier coefficients.

- FWT, the Fast Wavelet Transform. This converts a set of high resolution ϕ based coefficients into an equivalent ψ and ϕ coefficients (with a lower resolution ϕ, of course).

- H or $H(\xi)$, a complex function which is equivalent to the scaling step for a scaling function's Fourier transform. The definition is $H(\xi) = \frac{1}{\sqrt{2}} \sum_k h_k e^{-ik\xi}$. The scaling step is usually $\phi(x) = \sqrt{2} \sum_k h_k \phi(2x-k)$. Using $\widehat{\phi}$ it becomes

$$\widehat{\phi}(\xi) = \frac{1}{\sqrt{2}} \sum_k h_k\, e^{-ik\,\xi/2}\, \widehat{\phi}\left(\frac{\xi}{2}\right) = H\left(\frac{\xi}{2}\right)\, \widehat{\phi}\left(\frac{\xi}{2}\right).$$

- IFFT, the inverse of the Fast Fourier Transform (FFT). This converts a set of Fourier coefficients into the exact same number of evenly spaced function values.

- IFWT, the Inverse Fast Wavelet Transform, converting a set of ϕ and ψ coefficients into the equivalent, high resolution, ϕ coefficients.

- L^2, l^2, and so forth are discussed in the section on symbols.

Phrases and Names

- Almost Everywhere: a phrase from measure theory that is frequently necessary when working with spaces of functions. The phrase 'almost everywhere on Ω'

means 'on a subset $\Lambda \subset \Omega$ whose complement $\Omega \backslash \Lambda$ has measure zero.' Typically, the phrase will be used all alone, with 'almost everywhere' meaning 'almost everywhere on $\mathbb{R}$,' and we will always be using the Lebesgue measure. Measure theory is not easily explained, but for our purposes it suffices to be aware that an integration over a set of measure zero is always zero. The Lebesgue measure of a countably infinite number of individual points is zero, for instance. Basically, 'almost everywhere' means 'everywhere except on an inconsequential set.'

- Collocation based methods are a means to approximate the solution to partial or ordinary differential equations. They create function based approximations of the form $f(x) = \sum_k a_k g_k(x)$. The coefficients a_k are chosen so that $f(x)$ satisfies the given problem at a set of discrete points. We will use collocation to model spatial derivatives and the multiplication of non-constant functions (not for time discretization). For example, to approximate $\frac{\partial}{\partial x} f(x) = \sum_k a_k g_k(x)$ via collocation we will use

$$f'(x) \approx f_1(x) = \sum_k b_k g_k(x), \quad f_1(2^j m) = f'(2^j m), \quad \forall\, m \in \text{ index set } M.$$

  Notice that this means the $f_1(x)$ approximation of the first derivative will match the true first derivative at any M multiple of 2^j. If j is small (negative) and f is smooth, f_1 should be a fair approximation.

- Convolution, written $f * g$, defined in Section A.1.2.

- Fourier Series, a common way to express functions, usually those with periodic boundary conditions. Discussed in Section A.2.

- Infinity Norm: written $\|f\|_\infty$, this is a norm that usually relates to the supremum of f. If f is a sequence $\{f_n\}_{m \in \mathbb{N}}$ then $\|f\|_\infty = \sup_n |f_n|$. If f is a function then $\|f\|_\infty = \inf \{A \geqslant 0 | \, |f| \leqslant A \text{ almost everywhere } \}$, basically the supremum over all sets of non-zero measure. If f is continuous, then $\|f\|_\infty = \sup |f|$.

- Order of a spline function, a measure of the smoothness of a spline function. The first, B_0, is not continuous. The nth derivative of B_n will not be continuous, but the $(n-1)$th derivative will be continuous.

- Order of a wavelet, a concept measuring the effectiveness of a wavelet at modeling functions. The full explanation can be found in Section B.1.

- Orthogonal: in a space with an inner product $\langle \cdot, \cdot \rangle$, f is orthogonal to g, written $f \perp g$, if $\langle f, g \rangle = 0$.

- Orthonormal. A set $\{f_k\}_{k \in K}$ is orthonormal if $\langle f_k, f_l \rangle = \delta_{kl}$, meaning every pair of different f functions is orthogonal and the norm of any f function is one.

- Periodic: a property of functions where $f(x + y) = f(x)$ for some fixed value y. Common examples are functions with $f(x + 2\pi) = f(x)$, so 2π periodic functions. Usually these will be complex trigonometric polynomials composed of e^{ikx} terms ($k \in \mathbb{Z}$).

- Pseudo-Fourier: the standard means for approximating the multiplication of a Fourier decomposed function with some other, non-constant, function. The basic idea is to convert the function back into its physical values via the Inverse Fast Fourier Transform, then multiply the two functions, then convert the new values back using the Fast Fourier Transform. The nature of the IFFT and FFT are such that this means the calculation will be an approximation, but an approximation that matches perfectly with the true value at a set of equally spaced points.

- Pseudo-Wavelet: like Pseudo-Fourier, with wavelets (and scaling functions). The basic procedure is the same: convert the decomposition into physical values, calculate the multiplication, then convert back.

- Resolution, a concept applicable both to our function (wavelet and scaling function) and finite difference based decompositions. There are two ways of defining resolution for finite difference schemes. One could use Δx, the distance between the points, with a smaller distance meaning finer resolution, meaning better accuracy. The alternative, what we will use most commonly, is the reciprocal of Δx: the number of function values in any unit interval $(x, x + 1]$, $x \in \mathbb{R}$. The format will be '2^6 elements per unit.'

- Spline function: splines are symmetric, have finite support, and automatically include a scaling step. They are constructed piecewise from polynomials. We are only using them as biorthogonal scaling functions (Chapter 4). Section A.3 covers then in some detail

- Stiff problems are differential equations that are difficult to solve using explicit methods. A linear differential equation $\mathbf{u}_t = M\mathbf{u}$ is stiff if the matrix M has very large and very small eigenvalues, or, rather, a large maximum ratio between its eigenvalues. Partial differential equations can be described as stiff if they become stiff differential equations when discretized in their physical dimensions.

- Streamfunctions are a type of function Ψ mapping some physical domain Ω and time to $\mathbb{R}$. They are used to represent the flow of a fluid in the domain Ω. If $\Omega \subset \mathbb{R}^2$ then we get a function $\Psi(x, y)$ such that the flow in the x direction is equal to $-\Psi_y(x, y)$ and the flow in the y direction is equal to $\Psi_x(x, y)$. Notice that this makes the flow orthogonal to the (physical) gradient of Ψ, so the actual flow is represented by curves where Ψ is constant.

- Support. The support of a function f is equal to $\overline{\{x|f(x) \neq 0\}}$, with the line above indicating the closure.

- 'Switch-on' Function: a time based multiplier applied to the boundary conditions of a numerical calculation. We use a basic function of the form $f(t) = \min\left\{\frac{t}{t_1}, 1\right\}$, $t_1 > 0$, in Chapter 8. The functions ensures that the boundary conditions are applied gradually, avoiding instability in cases where the boundary conditions are very different from the initial conditions.

- Trigonometric Polynomial, a polynomial created by (positive) powers of cos and sin. We will frequently use complex trigonometric polynomials, of the form $\sum_{m=-M}^{M} a_m e^{imx}$, $a_m \in \mathbb{C}$. These are a more general form, since $\cos x = \frac{e^{ix}+e^{-ix}}{2}$ and $\sin x = \frac{e^{ix}-e^{ix}}{2i}$.

- Vorticity, a value in fluid dynamics expressing the extent of circular movement of the fluid around a point. In $\mathbb{R}^2$ it is simply $Z = \Delta\Psi = \frac{\partial^2}{\partial x^2}\Psi + \frac{\partial^2}{\partial y^2}\Psi$, with

Ψ the streamfunction. Due to the definition of the streamfunction, this means that the vorticity is

$$\begin{aligned} Z &= \frac{\partial}{\partial x}\left(\frac{\partial}{\partial x}\,\Psi\right) - \frac{\partial}{\partial y}\left(-\frac{\partial}{\partial y}\,\Psi\right) \\ &= \frac{\partial}{\partial x}\,(\text{Flow in the } y \text{ direction}) - \frac{\partial}{\partial y}\,(\text{Flow in the } x \text{ direction})\,. \end{aligned}$$

Assume we have a flow of zero at the origin, but a counterclockwise flow around it. This means that the flow in the y direction will be positive for $x > 0$ and negative for $x < 0$. Also, the flow in the x direction will be negative for $y > 0$ and positive for $y < 0$. So,

$$\frac{\partial}{\partial x}\,(\text{Flow in the } y \text{ direction}) > 0, \qquad \frac{\partial}{\partial y}\,(\text{Flow in the } x \text{ direction}) < 0.$$

So, if the flow is counterclockwise around a point then the vorticity at that point will be positive. Clockwise flow will result in a negative vorticity.

- Wavelet Decompositions are simply equivalent means of expressing a function. A wavelet decomposition of the function f is of the form

$$f(x) = \sum_k a_k\,\phi_{n,k}(x) + \sum_{j=0}^{m}\sum_k b_{n-j,k}\,\psi_{n-j,k}(x),$$

though we could also use only scaling functions, as in $f(x) = \sum_k a_k\,\phi_{n-j-1,k}(x)$.

Chapter 1

Introduction

1.1 Motivation

A major impediment to working with physically derived partial differential equations is that they are, invariably, stiff (see glossary). Those derived from fluid mechanics are no exception. Stiff problems are not well suited to explicit methods. Results become highly unstable when not using very small time steps, and computational round-off error becomes an issue when the time step is very small. Implicit methods are frequently brought into play, allowing large time steps to be used while preserving stability. However, implicit methods require the solving of a system of equations at every step. The system is likely to be both quite large and non-linear. Using wavelets to improve the efficiency of these methods is the purpose of this thesis.

Wavelets are, fundamentally, a means with which to express functions. The continuous wavelet transform of a function $f(x)$ is $W_\psi f(a, b)$, with a as resolution (basically frequency) and b as the location. The pattern continues with the coefficients from the discrete wavelet transform, which have the form $b_{j,k}$, j is resolution and k is location. This combination of resolution and location based information makes wavelets ideally suited to detecting areas of high irregularity (shocks, spikes, etc). If a function is fairly smooth at a location, the wavelet coefficients corresponding to that location will converge very quickly to zero as the resolution becomes finer. If the function is highly irregular at a point (if there is a shock nearby) then the coefficients will

converge to zero slowly. As long as you possess some means of calculating (or approximating) the wavelet coefficients (or the values of the continuous wavelet transform), it is possible to determine where and when the resolution is insufficient. Wavelets are rather versatile, so there are many approaches. One of the cleverer examples is found in [19], used to chose optimal cell sizes in a finite volume based turbulence model. This method exploits the fact that the coefficients $b_{j,k}$ converge to zero as the resolution becomes finer (as $j \to -\infty$, in our format). This convergence will be present in the linear correlation between $\log|b_{j,k}|$ and j. If the current resolution can detect that convergence, then the coefficients $b_{j,k}$ at finer resolution than those currently used are probably negligible. As a result, a sufficiently large slope in a best-fit linear modeling between $\log|b_{j,k}|$ and j will determine if the current resolution is sufficient. A simpler approach is to take a value ϵ, and if $|b_{j,k}| > \epsilon$ then the grid point, cells, etc, corresponding to $b_{j,k}$ are included. There is usually some additional use of the wavelet decomposition. For instance, in [17], the wavelet coefficients are analyzed to determine the order used for the finite difference approximation of the spatial derivatives, and not just the Δx distance between points. Further examples of wavelet decompositions used to determine resolution can be found in [15] and [11].

More closely related to our own method are those that use the wavelet decomposition directly. This means decomposing the function being modeled into a span of wavelets (and associated scaling functions). As with the previous applications, these methods involve using wavelet decomposition to make sure we are keeping track of the minimal number of terms, in this case the wavelet coefficients. Mostly this is done fairly directly. For instance, [13] discards coefficients if they are not large enough. At every time step, newly calculated coefficients are set to zero if their absolute value is smaller than a given ϵ. The coefficients included in the calculation are those that were found significant in the previous time step, as well as any coefficients near them. For instance, if $b^n_{j,k}$ (resolution j, location k and time step n) is significant then $b^{n+1}_{j,k-1}$ and $b^{n+1}_{j,k+1}$ would be calculated. Also, if $b^n_{j-1,K}$ is classified as relating to the same physical location as $b^n_{j,k}$ then it would also be calculated. This way, the basis functions used in the model will change based upon the numerical solution being calculated, adaptively. Other adaptive schemes based upon direct wavelet decomposition can be found in

[28], [3], [6] and [1] (where time, and not just the physical dimensions, is decomposed adaptively). There are, of course, other potential advantages to storing the data in wavelet form. In [4], for example, the main objective of the wavelet decomposition is to simplify the more important matrices used in numerical calculations.

Our contribution to wavelet methods for partial differential equations is quite different from those listed above. The idea of the method is to solve time steps at different levels of resolution, sequentially. The idea to exploit a multi-scale decomposition in this way is actually quite obvious, and so there are many examples to be found. For instance, in [21] a wavelet based adaptive scheme is used to solve steady states, with the problem solved repeatedly at multiple resolutions. In [27], an adaptive method is introduced for the solving of elliptic problems, making heavy use of multiple resolutions. The big difference between these methods and ours is that our method is for time based problems, whereas all similar methods, that we have found, are intended for elliptic problems (steady states, and the like). Most of what makes our method unique spring from this different starting point. We will discuss this in greater detail once the method is properly introduced.

1.2 Thesis Organization

The arrangement of the chapters is as follows. The table of contents is followed first by the list of figures then by the glossary. Chapter 2 deals with the basic and theoretical properties of wavelet decompositions, as well as such things as the symbols used to express them. Chapter 3 extends that more into numerical implementation, how to calculate the wavelets and related functions in a manner that is useful for computation. Chapters 2 and 3 primarily contain material adapted from [5], a very concise and readable introduction to wavelets, though primarily from a signal processing perspective. Chapter 4 extends results from Chapters 2 and 3 to incorporate biorthogonal wavelets. Chapter 5 covers some simple, but necessary, details for solving actual problems. Examples of numerical approximation of partial differential equations can be found in Chapter 6, starting with the heat equation. Nothing new is introduced until after Chapter 6. That being said, an understanding of our

own particular approaches to the subjects involved, especially over Chapters 5–6, will prove helpful to understanding the new material. This begins in Chapter 7, with the introduction of what we will call the 'multi-scale' method. It will be explained and applied to a simple problem in Chapter 7, then applied to a much more complicated one in Chapter 8. Some basic, but necessary, concepts are found in Appendix A. Supplementary material on wavelets can be found in Appendix B. Finally, Appendix C contains example MATLAB programs.

1.3 Thesis Contribution

As was said above, the original material begins in Chapter 7. What is introduced is a method that breaks apart the physical domain of a partial differential equation into different scales (resolutions, etc). First is the large-scale system, which has coarse resolution but covers the entire domain of the problem. Next there is one or more small-scale systems, which have fine resolution but only cover small subsets of the domain The method allows these different scales to be solved sequentially while maintaining consistency with an implicit time discretization. A single time step needs only be solved once at every scale, there is no need to cycle between them to maintain consistency. The method is useful for problems where highly localized small-scale interactions are present, two of which are tested in Chapters 7 and 8. This method is consistent with many others using wavelets. For instance, the different scales could use adaptive decompositions (see Section 1.1).

Chapter 2

The Wavelet Transform

2.1 Introduction

Wavelets are specialized tools for the expression of functions in a sequential (coefficient based) form. They have two major properties. One is that they are ready made with a 'fast transform' (e.g., the Fast Fourier Transform, FFT), simply referred to as the Fast Wavelet Transform or FWT, which we will get to later. The second is their localization. Contrasting with the Fourier series (Section A.2) can be illustrative here. Recall that Fourier series are made of infinitely differentiable functions that have support across the entire domain. There is a breakdown into frequencies, but not into location. Wavelets, on the other hand, are localized to varying degrees, as well as into different resolutions. The downside for this is, primarily, reduced differentiability. This chapter, and the following one, follow the subject of wavelets in much the same way as [5], an excellent introductory source. A more rigorous study can be found in [10]. Further reading can be found in [14].

At their most general, wavelets are a family of functions based on the 'mother' wavelet ψ. The general form will be written

$$\psi_{a,b}(x) = \frac{1}{|a|^{1/2}}\,\psi\left(\frac{x-b}{a}\right),\ a, b \in \mathbb{R}, \tag{2.1.1}$$

b determining location, a determining resolution. There are, of course, many restrictions on ψ.

Note: We will use the convention that any integral written without an explicit domain (so anything like $\int f(x)\,dx$) will be assumed to be over the entirety of $\mathbb{R}$. Most of our integrals will be of this form.

Definition 2.1.1 *A mother wavelet $\psi : \mathbb{R} \to \mathbb{C}$ is an L^2 function with $\|\psi\| = 1$ with*

$$2\pi \int \frac{\left|\widehat{\psi}(\xi)\right|^2}{|\xi|}\, d\xi = C_\psi < \infty. \tag{2.1.2}$$

Note that the function $\widehat{\psi}$ is the Fourier transform of ψ (see Section A.1). This is, technically, the only restriction upon ψ. The continuous transform related to ψ is

$$W_\psi f(a,b) = \langle f, \psi_{a,b} \rangle = \frac{1}{|a|^{1/2}} \int f(x)\, \overline{\psi\left(\frac{x-b}{a}\right)}\, dx, \quad a \neq 0, \tag{2.1.3}$$

and the inverse is

$$f(x) = \frac{1}{C_\psi}\, \frac{1}{a^2} \int\int W_\psi f(a,b)\; \psi\left(\frac{x-b}{a}\right)\, da\, db. \tag{2.1.4}$$

The inverse would be undefined if C_ψ were infinite, so we have that restriction. Definition 2.1.1 is enough to make ψ a valid wavelet, but not enough to make it a useful one. Wavelets with practical value will, according to [10, p. 7], have the additional requirement $\psi \in L^1$, which implies that $\widehat{\psi}$ is continuous (see Section A.1). If $\widehat{\psi}$ is continuous then $C_\psi < \infty$ implies that $\widehat{\psi}(0) = 0$ which means that $\int \psi(x)\, dx = 0$. In [5], a less general characterization is used:

Property 2.1.2 *If $\psi \in L^2$ and $x\psi(x) \in L^1$ then ψ satisfies Definition 2.1.1 if and only if*

$$\int \psi(x)\, dx = 0 \quad \Longleftrightarrow \quad \widehat{\psi}(0) = 0. \tag{2.1.5}$$

Proof.

Found in [5, p. 61]. ■

We will not use the continuous transform very often (for reading on the subject, see [10, Chapter 2]). What we will use is the discrete transform, which converts L^2 functions into l^2 sequences. It is convenient to use a constant change of resolution,

a zoom step, of $\sigma > 1$, so $a_n = \sigma^n$, and a variable spatial step $b_{j,k} = k\ \sigma^j\ \beta$. We will usually maintain this pattern: k for location, j for resolution. With the current setup, this results in a easily understood general form.

Definition 2.1.3 *The family of wavelets derived from a mother wavelet using scaling (zoom) step σ and spatial step β is*

$$\psi_{j,k}(x) = \frac{1}{\sqrt{\sigma^j}}\,\psi\left(\frac{x - k\,\sigma^j\,\beta}{\sigma^j}\right) = \frac{1}{\sqrt{\sigma^j}}\,\psi\left(\frac{x}{\sigma^j} - k\beta\right), \tag{2.1.6}$$

with $j \in \mathbb{Z}$ the scale term and $k \in \mathbb{Z}$ the translation term. Our standard values will be $\sigma = 2$ and $\beta = 1$, making the functions of the form

$$\psi_{j,k}(x) = \frac{1}{2^{j/2}}\,\psi\left(\frac{x}{2^j} - k\right) = 2^{-j/2}\,\psi\left(2^{-j}x - k\right). \tag{2.1.7}$$

The discrete transform will yield coefficients

$$c_{j,k} = Wf(a_j, b_{j,k}) = \frac{1}{\sigma^{j/2}}\int f(x)\,\overline{\psi\left(\frac{x}{\sigma^j} - \beta k\right)}\,dx, \tag{2.1.8}$$

equal to $\langle f, \psi_{j,k}\rangle$ in L^2. For this to yield a meaningful analysis, the $\psi_{n,k}$ should be orthonormal (in L^2). We already have them normalized, but orthogonality will prove to be a challenge. Even more important is making sure the $\psi_{j,k}$ span $L^2(\mathbb{R})$. Now, however, we will take a quick look at the simplest wavelet.

Example 2.1.4 *The Haar wavelet*

This is the most basic wavelet, and the best for explaining how wavelet decompositions work. We will go back to this example repeatedly. The Haar wavelet has a simple definition,

$$\psi_H(x) = \begin{cases} 1 & x \in [0, 1/2) \\ -1 & x \in [1/2, 1) \\ 0 & x \notin [0, 1) \end{cases} \tag{2.1.9}$$

A quick note about the Haar wavelet. We will identify it by ψ_H, but it can be justifiably written ψ_0, due to its status as the simplest wavelet in several families.

The Haar wavelet works based on $\sigma = 2$, $\beta = 1$. Notice that $\|\psi_H\|_2 = 1$ and $\int \psi_H(x)\,dx = 0$. The properties of different wavelets can be found using their Fourier transforms. In this case, we get

$$\widehat{\psi}_H(\xi) = \frac{i}{\sqrt{2\pi}} \frac{\sin^2(\xi/4)}{\xi/4} e^{-i\xi/2}. \tag{2.1.10}$$

The slow decay as $\xi \to \pm\infty$ indicates (and is caused by) ψ_H being discontinuous.

Property 2.1.5 *The family of functions*

$$\psi_{j,k}(x) = \frac{1}{2^{j/2}} \psi_H\left(\frac{x}{2^j} - k\right),\ j,k \in \mathbb{Z} \tag{2.1.11}$$

is an orthonormal basis of $L^2(\mathbb{R})$.

Proof.

Found in [5, page 22]. ■

There are many ways to prove this, some more direct than others. Some begin by proving that the functions $\mathbf{1}_{[k,k+1)}$ are in the span for any $k \in \mathbb{Z}$. From there you can get any function $\mathbf{1}_{[2^j k, 2^j(k+1))}$, for any $j,k \in \mathbb{Z}$, and those span $L^2(\mathbb{R})$.

2.2 Introducing the Scaling Function, ϕ

From now on we will fix the scaling factor to $\sigma = 2$ and the spatial step to $\beta = 1$. These values are standard, and make the resulting functions easier to visualize. Using $\beta = 1$ means that our wavelet family from Equation (2.1.6) with $j = 0$ will differ from each other by integer translates. Furthermore, those with a general j will differ from each other by 2^j sized translates. Converting between the different resolutions will be done with an operator D_σ (D_2, really), with

$$D_\sigma f(x) = f\left(\frac{x}{\sigma}\right) \quad \Longrightarrow \quad D_2(f(x)) = f\left(\frac{x}{2}\right). \tag{2.2.1}$$

What we want to construct is a multi-resolution analysis, a nested sequence of spaces V_a, $a \in \mathbb{Z}$, such that

$$\{0\} \subset \cdots \subset V_2 \subset V_1 \subset V_0 \subset V_{-1} \subset V_{-2} \subset \cdots \subset L^2 \tag{2.2.2}$$

with three key properties:

$$\overline{\bigcup_{j\in\mathbb{Z}} V_j} = L^2, \qquad \bigcap_j V_j = \{0\}, \quad \text{and} \quad V_{j+1} = D_2 V_a. \tag{2.2.3}$$

We start with some V_j spaces defined based upon a single function.

Definition 2.2.1 *A mother scaling function ϕ belongs to $L^1 \bigcap L^2$ and defines the spaces*

$$V_j = \text{Span}\left\{\phi\left(2^{-j}x - k\right), k \in \mathbb{Z}\right\}, \quad j \in \mathbb{Z}, \tag{2.2.4}$$

satisfying (2.2.2) and (2.2.3). An orthogonal scaling function ϕ will be such that $\{\phi(x-k),\, k \in \mathbb{Z}\}$ form an orthonormal basis of V_0.

Definition 2.2.2 *The family of functions derived from a mother scaling function ϕ contains the functions*

$$\phi_{j,k}(x) = 2^{-j/2}\,\phi\left(2^{-j}x - k\right), \tag{2.2.5}$$

with $j \in \mathbb{Z}$ relating to the scale and $k \in \mathbb{Z}$ the translation.

Notice that if the set $\{\phi(x-k), k \in \mathbb{Z}\}$ forms an orthonormal basis of V_0 then the set $\{\phi_{j,k}\}_{k\in\mathbb{Z}}$ will form an orthonormal basis of V_j.

Now we need to find necessary conditions and sufficient conditions for every component of Definition 2.2.1.

Theorem 2.2.3 *Assume a function $\phi \in L^1 \bigcap L^2$, that $\{\phi_{0,k} | k \in \mathbb{Z}\}$ is an orthonormal basis of V_0, and that*

$$|\phi(x)| \leqslant \frac{C}{1+x^2}, \;\; \textit{for some } C \in \mathbb{R}, \quad \forall\, x \in \mathbb{R}. \tag{2.2.6}$$

Those assumptions give us the V_j spaces from (2.2.2) with $V_{a+1} = D_2 V_a$ and

$$\bigcap_j V_j = \{0\}. \tag{2.2.7}$$

More importantly:

$$\left|\int_{\mathbb{R}} \phi(x)\,dx\right| = 1 \quad \Longleftrightarrow \quad V := \overline{\bigcup_j V_j} = L^2. \tag{2.2.8}$$

Proof.

This proof is quite lengthy. A good version can be found in [5, p. 131]. ■

The requirements from Theorem 2.2.3 are sufficient to make ϕ a proper scaling function, so it is possible to span L^2 with the V_j spaces. Our next step is to bring wavelets into the picture. We will find them in the pairwise orthogonal spaces

$$W_j = V_{j-1} \bigcap V_j^{\perp}, \quad j \in \mathbb{Z}. \tag{2.2.9}$$

This definition, based on the orthogonal complement of the smaller V space, has two immediate effects:

$$V_{j-1} = V_j \bigoplus W_j \quad \text{and} \quad W_j \perp V_j, \quad \forall j \in \mathbb{Z}. \tag{2.2.10}$$

The mother wavelet related to ϕ will be the generating function for W_0. Since $D_2(V_j) = V_{j+1}$ (D defined in Equation (2.2.1)) we also get

$$D_2(V_{j-1}) = D_2(V_j \bigoplus W_j) \qquad \Longrightarrow \qquad V_j = V_{j+1} \bigoplus D_2(W_j). \tag{2.2.11}$$

So, $D_2(W_j)$ contains the space W_{j+1}, as it has the necessary functions to extend V_{j+1} to V_j. Next, we notice that

$$W_j \perp V_j \qquad \Longrightarrow \qquad D_2(W_j) \perp D_2(V_j) \qquad \Longrightarrow \qquad D_2(W_j) \oplus V_{j+1}, \tag{2.2.12}$$

so $D_2(W_j) = W_{j+1}$.

Next, we prove that the W_j spaces, taken together, span L^2.

Theorem 2.2.4 *Assume we have a multi-resolution analysis*

$$0 \subset \cdots \subset V_j \subset V_{j-1} \subset \cdots L^2, \qquad \overline{\bigcup_{j \in \mathbb{Z}} V_j} = L^2, \qquad \bigcap_{j \in \mathbb{Z}} V_j = 0, \qquad D_2(V_j) = V_{j+1}.$$

This setup implies that the spaces $W_j = V_{j-1} \bigcap V_j^{\perp}$ (as discussed above) are pairwise orthogonal and that

$$\overline{\bigoplus_j W_j} = L^2. \tag{2.2.13}$$

Proof.

First, orthogonality follows directly from the definition of the W_j spaces.

Now to deal with the second result, Equation (2.2.13). Take $f \perp W_j$ for all $j \in \mathbb{Z}$. By the completeness of $\bigcup_j V_j$ we know that for any $\epsilon > 0$ there exists a $j \in \mathbb{Z}$ and an $h_0 \in V_j$ such that $\|f - h_0\| < \epsilon$. For simplicity, we will assume that $j = 0$. We now write f in terms of the W_j spaces. As $V_0 = V_1 \oplus W_1$, there exists $h_1 \in V_1$ and $g_1 \in W_1$ such that $h_0 = h_1 + g_1$. As $V_1 = V_2 \oplus W_2$ then there exists $h_2 \in V_2$ and $g_2 \in W_2$ such that $h_1 = h_2 + g_2$ and, in turn, $h_0 = h_2 + g_2 + g_1$. This process continues indefinitely:

$$h_0 = h_n + \sum_{k=1}^{n} g_k, \qquad \forall\, n > 1 \in \mathbb{N}. \tag{2.2.14}$$

As the functions g_k must be orthogonal, being in different W_k spaces, we get

$$\|h_n\|^2 + \sum_{k=1}^{n} \|g_k\|^2 = \|h_0\|^2 \tag{2.2.15}$$

which implies that the series $\{\|g_k\|\}$ is convergent in l^2. This, in turn, implies that the series $\{h_n\}_{n\in\mathbb{N}}$ is convergent in L^2, converging to some function h. Because of the structure of this series, $h \in \bigcap_k V_k$, so $h = 0$ by the properties of the V_k spaces. Using this on (2.2.14) as $n \to \infty$ gives us

$$\sum_{k=1}^{\infty} g_k = h_0. \tag{2.2.16}$$

As f is orthogonal to all W_k spaces, this gives us

$$\langle f, h_0 \rangle = \sum_{k=1}^{\infty} \langle f, g_k \rangle = 0 \tag{2.2.17}$$

so

$$\|f\|^2 \;=\; \left(\|f\|^2 + \|h_0\|^2\right) - \|h_0\|^2 \;=\; \|f - h_0\|^2 - \|h_0\|^2 \;<\; \epsilon - \|h_0\|^2 \tag{2.2.18}$$

using that orthogonality. So, $\|h_0\|^2 < \epsilon$, where $\epsilon = \|h_0 - f\|$. If $f \neq 0$, we get a contradiction, so $f = 0$, and the W_j spaces span L^2. ■

2.3 More Key Properties of Scaling Functions

Here we will build on the results from the previous sections. Recall Definition 2.2.2 for $\phi_{j,k}$, and that $\phi = \phi_{0,0}$ is the mother scaling function. Also recall the sufficient requirements found in Theorem 2.2.3 for the functions $\{\phi_{j,k}\}_{j,k}$ to span L^2. We will assume that any function ϕ satisfies the definition of a scaling function, as well as

$$\int \phi(x)\,dx = 1 \quad \Longrightarrow \quad \widehat{\phi}(0) = \frac{1}{\sqrt{2\pi}}. \tag{2.3.1}$$

Some key properties of the V_j spaces will put further restrictions on ϕ.

Theorem 2.3.1 *For the V_j spaces and scaling function ϕ from Definition 2.2.1,*

$$V_0 \subset V_{-1} \quad \Longleftrightarrow \quad \phi(x) = \sqrt{2}\sum_{k\in\mathbb{Z}} h_k\,\phi(2x-k) = \sum_k h_k\,\phi_{-1,k}(x) \tag{2.3.2}$$

for some $h_k \in \mathbb{R}$.

Proof.

The space V_{-1} is spanned by $\phi(2x-k)$, $k \in \mathbb{Z}$, and V_0 is spanned by $\phi(x-k)$, $k \in \mathbb{Z}$. If $V_0 \subset V_{-1}$ then, clearly, $\phi(x)$ needs to be writable in terms of functions $\phi(2x-k)$. Also, if $\phi(x)$ is equal to a sum of functions $\phi(2x-k)$, then so is any $\phi(x-k)$, $k \in \mathbb{Z}$. This means that any function in V_0 is in V_{-1} and $V_0 \subset V_{-1}$. ■

The expression of ϕ as a weighted sum of functions $\phi(2x-k)$ will typically be referred to as the 'scaling step.' The weights h_k are going to be very important from now on, so we will avoid calling anything else h. We can find

$$\delta_{0d} = \int_{t\in\mathbb{R}} \phi(x-d)\,\overline{\phi(x)}\,dx = 2\sum_{k,j\in\mathbb{Z}} h_k\overline{h_j}\int_{x\in\mathbb{R}} \phi(2x-2d-k)\,\overline{\phi(2x-j)}\,dx \tag{2.3.3}$$

by taking advantage of the scaling step. Now a simple change of variable for

$$\begin{aligned}\delta_{0d} &= \sum_{k,j\in\mathbb{Z}} h_i\overline{h_j}\int \phi(t-2d-k)\overline{\phi(x-j)}\,dx \\ &= \sum_{k,j\in\mathbb{Z}} h_k\overline{h_j}\,\delta_{(2d+k)(j)} \quad = \sum_k h_k\overline{h_{2d+k}}.\end{aligned} \tag{2.3.4}$$

The expression for $d = 0$ gives us $\sum_k |h_k|^2 = 1$. We can also get a simple sum using the integral over $\mathbb{R}$:

$$\begin{aligned} \phi(x) &= \sqrt{2} \sum_k h_k \phi(2x-k), \\ \int \phi(x)\, dx &= \sqrt{2} \sum_k h_k \int \phi(2x-k) dt, \\ 1 &= \sqrt{2} \sum_k h_k \frac{1}{2}, \end{aligned} \tag{2.3.5}$$

which simplifies to $\sqrt{2} = \sum_k h_k$. Also, since the $\phi(x-k)$, $k \in \mathbb{Z}$ are orthonormal,

$$\phi(x) = \sqrt{2} \sum_k h_k\, \phi(2x-k) = \sqrt{2} \sum_k \langle \phi\,,\, \phi(2\cdot - k) \rangle\, \phi(2x-k). \tag{2.3.6}$$

As a result, for h_k to be non-zero requires the related ϕ to have overlapping support. A direct implication can be stated:

Property 2.3.2 *If the scaling function ϕ has compact support then only finitely many h_k are non-zero.*

Proving the related converse to the above property takes a little more effort.

Definition 2.3.3 *To restrict the support of functions we will use two values:*

$$a(f) = \inf\{x | f(x) \neq 0\}, \qquad b(f) = \sup\{x | f(x) \neq 0\}, \tag{2.3.7}$$

applied to any function f. This means that $\operatorname{supp}(f) \subset [a(f), b(f)]$.

Theorem 2.3.4 *If the scaling function ϕ has compact support then $a(\phi)$ and $b(\phi)$ are integers. Furthermore, all h_k with $k \notin [a(\phi), b(\phi)]$ are equal to zero.*

Proof.

First, we have a finite number of h_k not equal to zero, so we have a $k_{\min}$ and $k_{\max}$ such that $h_k = 0$ if $k > k_{\max}$ or $k < k_{\min}$. For arbitrary $k \in \mathbb{Z}$, we can find that

$$a(\phi_{-1,k}) = \frac{1}{2}[a(\phi) + k], \quad \text{and} \quad b(\phi_{-1,k}) = \frac{1}{2}[b(\phi) + k]. \tag{2.3.8}$$

Taking the a value of both sides of $\phi(x) = \sum_k h_k \phi_{-1,k}(x)$ leads to

$$a(\phi) = a(\phi_{-1,k_{\min}}) \qquad b(\phi) = b(\phi_{-1,k_{\max}}), \tag{2.3.9}$$

since any $\phi_{-1,k}$ for $k < k_{\min}$ or $k > k_{\max}$ would be multiplied by $h_k = 0$. Combining those with the previous results leads to

$$\begin{aligned} a(\phi) = \frac{1}{2}\big[a(\phi) + k_{\min}\big] \quad &\Longrightarrow \quad a(\phi) = k_{\min}, \\ b(\phi) = \frac{1}{2}\big[b(\phi) + k_{\max}\big] \quad &\Longrightarrow \quad b(\phi) = k_{\min}, \end{aligned} \tag{2.3.10}$$

and we are done. ■

Finitely supported scaling functions (and wavelets) are more convenient for our purposes. We will be working with finite domains, and finite scaling functions simply fit better. Furthermore, we are looking to use the functions ϕ and ψ for numerical simulations. A computer program can only work with a finite number of values h_k at a time.

Next we take a closer look at the orthonormality property, specifically, an equivalent property tied to $\widehat{\phi}$. Scaling functions are in L^2 and L^1, so their Fourier transform is well defined. If we divide up the integral we get

$$\|\phi\|^2 = \left\|\widehat{\phi}\right\|^2 = \int_{\mathbb{R}} \left|\widehat{\phi}(\xi)\right|^2 d\xi = \sum_k \int_0^{2\pi} \left|\widehat{\phi}(\xi + 2\pi k)\right|^2 d\xi. \tag{2.3.11}$$

Now we reverse the order there, put the sum inside the integral. We get

$$\|\phi\|^2 = \int_0^{2\pi} \Phi(\xi)\, d\xi, \quad \text{with} \quad \Phi(\xi) = \sum_k \left|\widehat{\phi}(\xi + 2\pi k)\right|^2. \tag{2.3.12}$$

This new function Φ is going to be defined almost everywhere, be 2π periodic and belong to $L^2([0, 2\pi])$. Its purpose is almost entirely related to orthonormality.

Theorem 2.3.5 *A set of functions $\{\phi(\cdot - k)\}_{k\in\mathbb{Z}}$ is an orthonormal system if and only if*

$$\Phi(\xi) = \sum_k \left|\widehat{\phi}(\xi + 2\pi k)\right|^2 = \frac{1}{2\pi} \quad \textit{almost everywhere.} \tag{2.3.13}$$

Proof.

Consider the inner product $\langle \phi, \phi_{0,k} \rangle$:

$$\langle \phi, \phi_{0,k} \rangle = \left\langle \widehat{\phi}, \widehat{\phi}_{0,k} \right\rangle = \int \widehat{\phi}(\xi)\, \overline{e^{ik\xi}\, \widehat{\phi}(\xi)}\, d\xi = \int \left|\widehat{\phi}(\xi)\right|^2 e^{-ik\xi}\, d\xi. \tag{2.3.14}$$

Next, we break up that integral into 2π sized segments, making

$$\langle \phi, \phi_{0,k} \rangle = \sum_l \int_0^{2\pi} \left|\widehat{\phi}(\xi + 2\pi l)\right|^2 e^{-ik\xi}\, d\xi = \int_0^{2\pi} \Phi(\xi) e^{-ik\xi}\, d\xi. \tag{2.3.15}$$

So, the $\phi_{0,k}$ are an orthonormal set if and only if

$$\langle \phi, \phi_{0,k} \rangle = \delta_{0,k} = \int_0^{2\pi} \Phi(\xi) e^{-ik\xi}\, d\xi. \tag{2.3.16}$$

The integral on the right of (2.3.16) is best though of in relation to the Fourier decomposition of Φ (Section A.2). Each integral for a particular k will be related to the Fourier coefficient for $e^{ik\xi}$. As a result, the integral will be zero for all $k \neq 0$ if and only if $\Phi = C$ almost everywhere (as in, if and only if Φ is a constant function almost everywhere). The other requirement is for the $k = 0$ integral to be equal to one, which requires $\Phi = \frac{1}{2\pi}$. Therefore, the $\phi(x - k)$ are orthonormal if and only if Φ is constant and equal to $\frac{1}{2\pi}$ almost everywhere. ■

Example 2.3.6 *The Haar scaling function ϕ_H is simply the indicator function for $[0, 1)$, $\mathbf{1}_{[0,1)}$ (equal to one in that range, zero elsewhere).*

It is obvious that $\langle \phi_H(\cdot - k), \phi_H(\cdot) \rangle = \delta_{0k}$, and that the progressively finer V_j spaces can approximate any integrable function arbitrarily well.

2.4 Defining the Wavelet, ψ

The wavelet, ψ, is actually defined in terms of the extensions from spaces V_j to spaces V_{j-1}, the next, larger, space on the chain. Proving the properties of ψ will involve making heavy use of the scaling step. It is inconvenient to deal with all the translations

of ϕ that are applied in the step. However, if we take the Fourier transform on both sides of the scaling equation

$$\phi(x) = \sqrt{2} \sum_k h_k \, \phi(2x - k) \tag{2.4.1}$$

we get

$$\widehat{\phi}(\xi) = \frac{1}{\sqrt{2}} \sum_k h_k \, e^{-ik\xi/2} \, \widehat{\phi}\left(\frac{\xi}{2}\right), \tag{2.4.2}$$

nicely removing the translation component. We actually have the scaling step written as a multiplication with a simple function, which we will call H.

Definition 2.4.1 *For a given ϕ (Definition 2.2.1),*

$$H(\xi) := \frac{1}{\sqrt{2}} \sum_k h_k \, e^{-ik\xi}. \tag{2.4.3}$$

The function H simplifies Equation (2.4.2), the scaling step, into

$$\widehat{\phi}(\xi) = H\left(\frac{\xi}{2}\right) \widehat{\phi}\left(\frac{\xi}{2}\right). \tag{2.4.4}$$

It is important to notice that H is a function of period 2π, due to its basic structure.

Property 2.4.2 *Assume that we have a scaling function ϕ, with scaling step related function H. If $\{\phi(x-k),\, k \in \mathbb{Z}\}$ forms an orthonormal basis for V_0 then we get*

$$|H(\xi)|^2 + |H(\xi + \pi)|^2 = 1 \;\; \textit{almost everywhere.} \tag{2.4.5}$$

Proof.

This comes directly from the use of Φ. First, we use Theorem 2.3.5, so $\langle \phi(x), \phi(x-k) \rangle = \delta_{0k}$ for $k \in \mathbb{Z}$ if and only if

$$\Phi(\xi) = \sum_k \left|\widehat{\phi}(\xi + 2\pi k)\right|^2 = \frac{1}{2\pi}, \qquad \text{almost everywhere.} \tag{2.4.6}$$

Next, we re-write the sum in a manner that will be used fairly often: breaking up a sum of 2π translates into two separate sums of 4π translates. Simply,

$$\Phi(\xi) = \sum_k \left|\widehat{\phi}(\xi + 2\pi k)\right|^2 = \sum_k \left|\widehat{\phi}(\xi + 4\pi k)\right|^2 + \sum_k \left|\widehat{\phi}(\xi + 2\pi + 4\pi k)\right|^2. \tag{2.4.7}$$

The next step is to include the Fourier transformed scaling step, so

$$\Phi(\xi) = \sum_k |H(\xi/2 + 2\pi k)|^2 \left|\widehat{\phi}(\xi/2 + 2\pi k)\right|^2 + \sum_k |H(\xi/2 + \pi + 2\pi k)|^2 \left|\widehat{\phi}(\xi/2 + \pi + 2\pi k)\right|^2. \tag{2.4.8}$$

The function H is 2π periodic, so that equation can be rewritten

$$\begin{aligned}\Phi(\xi) &= |H(\xi/2)|^2 \sum_k \left|\widehat{\phi}(\xi/2 + 2\pi k)\right|^2 + |H(\xi/2 + \pi)|^2 \sum_k \left|\widehat{\phi}(\xi/2 + 2\pi k)\right|^2 \\ &= |H(\xi/2)|^2 \Phi(\xi/2) + |H(\xi/2 + \pi)|^2 \Phi(\xi/2 + \pi). \end{aligned} \tag{2.4.9}$$

Recall that $\Phi(\xi) = \frac{1}{2\pi}$ almost everywhere, so we can, effectively, factor Φ out for $|H(\xi/2)|^2 + |H(\xi/2 + \pi)|^2 = 1$ almost everywhere. ■

There are two things to notice here. First, what we have above is a requirement for orthonormality, not a sufficient condition. An additional, easy to satisfy, condition will be added in a later section (in Theorem 3.1.6) to rectify this situation. One more thing to notice: $H(0) = H(2\pi) = 1$ due to the properties of the coefficients h_k, meaning that we get $H(\pi) = 0$ for a continuous H.

Now we start discussing $W_0 = V_{-1} \bigcap V_0^{\perp}$. First, we will characterize the two necessary requirements (using the Fourier transforms). Next, we will prove that those requirements are tied up with a specific function ψ, our wavelet.

Lemma 2.4.3 *Take a scaling function ϕ (Definition 2.2.1) and resulting spaces V_j. A function f belongs to $W_0 = V_{-1} \bigcap V_0^{\perp}$ if and only if*

$$\widehat{f}(\xi) = m_f\left(\frac{\xi}{2}\right) \widehat{\phi}\left(\frac{\xi}{2}\right), \tag{2.4.10}$$

with m_f a complex function in L_0^2 such that

$$\begin{bmatrix} m_f(\omega) \\ m_f(\omega + \pi) \end{bmatrix} = \lambda(\omega) \begin{bmatrix} H(\omega + \pi) \\ -H(\omega) \end{bmatrix} \qquad \textit{almost everywhere}, \tag{2.4.11}$$

for some function λ and the H function related to ϕ (Definition 2.4.1).

Proof.

First, we have to take a look at the requirement that $f \in V_{-1}$, which is

$$f = \sum_k \langle f, \phi_{-1,k} \rangle \, \phi_{-1,k} = \sum_k f_k \, \phi_{-1,k}. \tag{2.4.12}$$

Taking the Fourier transform of both sides results in the equivalent expression

$$\widehat{f}(\xi) = \frac{1}{\sqrt{2}} \sum_k f_k \, e^{-ik\xi/2} \, \widehat{\phi}(\xi/2), \tag{2.4.13}$$

so define

$$m_f(\xi) = \frac{1}{\sqrt{2}} \sum_k f_k \, e^{-ik\xi}. \tag{2.4.14}$$

This means that

$$f \in V_{-1} \quad \Longleftrightarrow \quad \widehat{f}(\xi) = m_f\left(\frac{\xi}{2}\right) \widehat{\phi}\left(\frac{\xi}{2}\right), \quad m_f \text{ is } 2\pi \text{ periodic.} \tag{2.4.15}$$

So, we have characterized $f \in V_{-1}$, now we need to look into $f \perp V_0$, so $\langle f, \phi_{0,k} \rangle = 0$ for all $k \in \mathbb{Z}$. This gives us

$$0 = \langle f, \phi_{0,k} \rangle = \left\langle \widehat{f}, \widehat{\phi}_{0,k} \right\rangle = \int \widehat{f}(\xi) \, \overline{\widehat{\phi}(\xi)} \, e^{ik\xi} \, d\xi. \tag{2.4.16}$$

If we break up the integral into $[0, 2\pi]$ sections then it is equal to

$$0 = \int_0^{2\pi} \left[\sum_l \widehat{f}(\xi + 2\pi l) \, \overline{\widehat{\phi}(\xi + 2\pi l)} \right] e^{ik\xi} \, d\xi. \tag{2.4.17}$$

This means that $f \perp V_0$ if and only if the 2π-periodic function in the square brackets has no non-zero terms in its Fourier decomposition. Since $\left\{ e^{ikx}, k \in \mathbb{Z} \right\}$ spans L_0^2 (see Section A.2), this means that

$$f \perp V_0 \Longleftrightarrow \quad \sum_l \widehat{f}(\xi + 2\pi l) \, \overline{\widehat{\phi}(\xi + 2\pi l)} = 0 \text{ almost everywhere on } [0, 2\pi]. \tag{2.4.18}$$

Now, we can take the sum (2.4.18) and break it up yet again into

$$\begin{aligned} 0 &= \sum_l \widehat{f}(\xi + 2\pi l) \, \overline{\widehat{\phi}(\xi + 2\pi l)} \\ &= \sum_l \widehat{f}(\xi + 4\pi l) \, \overline{\widehat{\phi}(\xi + 4\pi l)} + \sum_l \widehat{f}(\xi + 4\pi l + 2\pi) \, \overline{\widehat{\phi}(\xi + 4\pi l + 2\pi)}. \end{aligned} \tag{2.4.19}$$

Now, we assume that $f \in V_{-1}$. This gives us the function m_f from above, which we use along with the scaling step on ϕ for

$$\begin{aligned}
0 &= \sum_l \left(m_f(\xi/2)\, \widehat{\phi}(\xi/2 + 2\pi l) \right) \overline{\left(H(\xi/2)\, \widehat{\phi}(\xi/2 + 2\pi l) \right)} \\
&\qquad + \sum_l \left(m_f(\xi/2 + \pi)\, \widehat{\phi}(\xi/2 + 2\pi l + \pi) \right) \overline{\left(H(\xi/2 + \pi)\, \widehat{\phi}(\xi/2 + 2\pi l + \pi) \right)} \\
&= \sum_l m_f(\xi/2)\, \overline{H(\xi/2)} \left| \widehat{\phi}(\xi/2 + 2\pi l) \right|^2 \\
&\qquad + \sum_l m_f(\xi/2 + \pi)\, \overline{H(\xi/2 + \pi)} \left| \widehat{\phi}(\xi/2 + 2\pi l + 2\pi) \right|^2 . \qquad (2.4.20)
\end{aligned}$$

Since l appears only in the functions $\widehat{\phi}$, and those sum to $\frac{1}{2\pi}$, we get a final result where $f \in V_{-1}$ is perpendicular to V_0 if and only if

$$m_f(\xi/2)\, \overline{H(\xi/2)} + m_f(\xi/2 + \pi)\, \overline{H(\xi/2 + \pi)} = 0 \quad \text{almost everywhere.} \qquad (2.4.21)$$

A simple change of variable can get rid of the 2, and the whole expression can be written in $\mathbb{C}^2$ as $f \in V_{-1}$ is $\perp V_0$ if and only if

$$\begin{bmatrix} m_f(\omega) \\ m_f(\omega + \pi) \end{bmatrix} \cdot \begin{bmatrix} H(\omega) \\ H(\omega + \pi) \end{bmatrix} = 0 \qquad \text{almost everywhere.} \qquad (2.4.22)$$

What we do now is characterize the m_f based vector using the fact that it is orthogonal to the vector based on H. If the H based vector has a norm of zero, this would be a useless exercise, but Property 2.4.2 makes it clear that the H based vector has a norm of one for almost all ω. As a result,

$$f \perp V_0 \quad \Longleftrightarrow \quad \begin{bmatrix} m_f(\omega) \\ m_f(\omega + \pi) \end{bmatrix} = \lambda(\omega) \begin{bmatrix} H(\omega + \pi) \\ -H(\omega) \end{bmatrix} \quad \text{almost everywhere} \qquad (2.4.23)$$

for some function λ. That only applies for $f \in V_{-1}$.

Now to put it all together. We are after proving the equivalency of $f \in W_0$ to $\widehat{f}(\xi) = m_f(\xi/2)\, \widehat{\phi}(\xi/2)$, m_f a 2π periodic function satisfying Equation (2.4.11). So, for one direction we start by assuming $f \in W_0$, so $f \in V_{-1}$ and $f \perp V_0$. Since $f \in V_{-1}$ we can use Equation (2.4.23), where $f \perp V_0$ gives us the required function m_f.

For the other direction we start with an m_f that is 2π periodic, so we have $f \in V_{-1}$ using the result in Equation (2.4.15). As a result, we get to use Equation (2.4.23) making $f \perp V_0$ so $f \in W_0$. ■

Lemma 2.4.4 *A function $f \in W_0$ has a Fourier transform of the form*

$$\widehat{f}(\xi) = v(\xi)\, e^{i\xi/2}\, \overline{H(\xi/2+\pi)}\, \widehat{\phi}(\xi/2) \tag{2.4.24}$$

for a function $v \in L_0^2$.

Proof.

We start with Equation (2.4.11), and our first objective is to re-express the function m_f in terms of H. Finding λ is surprisingly easy. What we do is take the dot product of both sides of Equation (2.4.11) with the vector $[H(\omega+\pi)\,,\, -H(\omega)]^T$. That vector has a norm of one for almost all ω due to Property 2.4.2, so we get

$$\lambda(\omega) = m_f(\omega)H(\omega+\pi) - m_f(\omega+\pi)H(\omega). \tag{2.4.25}$$

Notice that

$$\begin{aligned} \lambda(\omega+\pi) &= m_f(\omega+\pi)H(\omega+2\pi) - m_f(\omega+2\pi)H(\omega+\pi) \\ &= m_f(\omega+\pi)H(\omega) - m_f(\omega)H(\omega+\pi) \;=\; -\lambda(\omega) \end{aligned} \tag{2.4.26}$$

implying that $\lambda(\omega) = e^{i\omega}\, v(2\omega)$ for some v with period 2π. Next, we substitute that back into our expression for m_f for

$$m_f(\omega) = e^{i\omega}\, v(2\omega)\, \overline{H(\omega+\pi)} \tag{2.4.27}$$

and, finally,

$$\widehat{f}(\xi) = e^{i\xi/2}\, v(\xi)\, \overline{H(\xi/2+\pi)}\, \widehat{\phi}(\xi/2). \tag{2.4.28}$$

We have not quite confirmed that v is in L_0^2, as that requires a finite norm. Take v and re-write as $v(\xi) = e^{-i\xi/2}\, \lambda(\xi/2)$. This results in

$$|v(\xi)|^2 = |\lambda(\xi/2)|^2 \left|e^{-i\xi}\right|^2 = |\lambda(\xi/2)| \tag{2.4.29}$$

which is precisely the norm of $\left[m_f(\xi/2)\,,\, m_f(\xi/2+\pi)\right]^T$, so

$$|v(\xi)|^2 = |m_f(\xi/2)|^2 + |m_f(\xi/2+\pi)|^2\,. \tag{2.4.30}$$

Now to get the norm of v,

$$\begin{aligned}\|v\|^2 = \frac{1}{2\pi}\int_0^{2\pi} |v(\xi)|^2\, d\xi &= \frac{1}{2\pi}\int_0^{2\pi} |m_f(\xi/2)|^2 + |m_f(\xi/2+\pi)|^2\, d\xi \\ &= \frac{1}{\pi}\int_0^{2\pi} |m_f(\omega)|^2 + |m_f(\omega+\pi)|^2\, d\omega \\ &= 2\,\|m_f\|_{L_0^2}^2 = \sum |f_k|^2 = \|f\|^2 \ < \infty \end{aligned} \tag{2.4.31}$$

and we are done. ■

Theorem 2.4.5 *A function $f \in L^2$ belongs to W_0 if and only if there exists a function $v \in L_0^2$ such that the Fourier transform $\widehat{f}$ can be written as in Equation (2.4.28).*

Proof.

We have already seen one direction, so we know that if $f \in W_0$ then there is a function v that satisfies Equation (2.4.28).

Next, assume that $\widehat{f}$ is in the form of Equation (2.4.28). Showing f to be orthogonal to V_0 is actually the easiest part. We need to look at the function m_f, with

$$\begin{bmatrix} m_f(\omega) \\ m_f(\omega+\pi) \end{bmatrix} = \begin{bmatrix} e^{i\omega}\, v(2\omega)\, \overline{H(\omega+\pi)} \\ -e^{i\omega}\, v(2\omega)\, \overline{H(\omega)} \end{bmatrix} = v(2\omega)\, e^{i\omega} \begin{bmatrix} \overline{H(\omega+\pi)} \\ -\overline{H(\omega)} \end{bmatrix}, \tag{2.4.32}$$

taking advantage of both v and H being 2π periodic. That expression meets the orthogonality requirement from Equation (2.4.22), so $f \perp V_0$.

Our next requirement is for f to be in V_{-1}. There are two requirements. The first is for the resulting m_f to have a finite norm. This works out fairly easily. The first line of Equation (2.4.31) links up the norms of v, m_f and f. Next we need f to be written in terms of $\phi(2x-k)$ terms. This is equivalent to having $\widehat{f}$ written in terms of $\widehat{\phi}(\xi/2)$, which it is, and to m_f being 4π periodic. Remember that converting to

Fourier conjugates inverts the multipliers inside functions. The function m_f is defined in terms of $e^{i\xi/2}$, $v(\xi/2)$ and $H(\xi/2+\pi)$, and all are 4π periodic, so m_f is as well. ■

Now to characterize ψ in terms of H and $\widehat{\phi}$. Take a look at the identity

$$\widehat{f}(\xi) = v(\xi)\, e^{i\xi/2}\, \overline{H(\xi/2+\pi)}\, \widehat{\phi}(\xi/2) \tag{2.4.33}$$

for any $f \in W_0$. Notice that $v(\xi)$ is the only function in the expression that relates particularly to f: the rest is always there for all functions in W_0. As a result, we take

$$\widehat{\psi}(\xi) = e^{i\xi/2}\, \overline{H(\xi/2+\pi)}\, \widehat{\phi}(\xi/2) \tag{2.4.34}$$

as a possible candidate for the mother wavelet.

Theorem 2.4.6 *If we have a scaling function ϕ and related function H, then the $\psi(\cdot - k)$ from Equation (2.4.34) is an orthonormal basis for W_0.*

Proof.

First, we prove that the system is orthonormal using Equation (2.3.5), which requires that the sum $\sum_k \left|\widehat{\psi}(\xi+2\pi k)\right|^2$ add up to $\frac{1}{2\pi}$. So,

$$\sum_k \left|\widehat{\psi}(\xi+2\pi k)\right|^2 = \sum_k \left|\widehat{\psi}(\xi+4\pi k)\right|^2 + \sum_k \left|\widehat{\psi}(\xi+2\pi+4\pi k)\right|^2. \tag{2.4.35}$$

We next use (2.4.34) for

$$\begin{aligned}\sum_k \left|\widehat{\psi}(\xi+2\pi k)\right|^2 &= \left|H\left(\frac{\xi}{2}+\pi\right)\right|^2 \sum_k \left|\widehat{\phi}\left(\frac{\xi}{2}+2\pi k\right)\right|^2 \\ &\qquad + \left|H\left(\frac{\xi}{2}\right)\right|^2 \sum_k \left|\widehat{\phi}\left(\frac{\xi}{2}+\pi+2\pi k\right)\right|^2 \\ &= \left|H\left(\frac{\xi}{2}+\pi\right)\right|^2 \left(\frac{1}{2\pi}\right) + \left|H\left(\frac{\xi}{2}\right)\right|^2 \left(\frac{1}{2\pi}\right) \quad = \frac{1}{2\pi}\end{aligned} \tag{2.4.36}$$

since $\phi(\cdot - k)$, $k \in \mathbb{Z}$ forms an orthonormal system.

Next, we have

$$\widehat{\psi}(\xi) = e^{i\xi/2}\, \overline{H\left(\frac{\xi}{2}+\pi\right)}\, \widehat{\phi}(\xi/2), \tag{2.4.37}$$

which can be written in the form of Equation (2.4.28) using $v = 1$, which is in L_0^2. Combining this with Theorem 2.4.5 means that ψ and its integer translates are in W_0, since the translates will merely have an L_0^2 term of the form e^{ik} in front.

So, the $\{\psi_k\}_{k\in\mathbb{Z}}$ are orthonormal and are in W_0. Next we need the most important property: they need to span W_0. We have

$$\widehat{f}(\xi) = e^{i\xi/2} v(\xi) \overline{H\left(\frac{\xi}{2}+\pi\right)}\, \widehat{\phi}(\xi/2) = v(\xi)\widehat{\psi}(\xi), \quad v \in L_0^2 \tag{2.4.38}$$

for any f in W_0. Recall that any $v(\xi) \in L_0^2$ is the limit of a Fourier series, so

$$\widehat{f}(\xi) = \sum_k v_k\, e^{-ik\xi}\, \widehat{\psi}(\xi), \tag{2.4.39}$$

which is just an inverse Fourier transform from

$$f(x) = \sum_k v_k\, \psi(x-k) \tag{2.4.40}$$

using the Fourier transform identities. That can easily be re-written for

$$f(x) = \sum_k v_k\, \psi_{0,k}(x), \tag{2.4.41}$$

and we are done. ■

Notice that (2.4.34) does not completely define ψ. Its actual location is not specified, so we could, for instance, add an integer value to x in $\psi(x)$ and the resulting function could still be used to characterize W_0. Also, multiplying the resulting function by -1 would not change its essential properties. Since the shape is very specific, we just need the location to be imposed at some point. The main issue is that we have $\widehat{\psi}$, the Fourier transform of ψ, and not the original function. From the way the W_0, V_0 and V_{-1} are defined, we are certain to get ψ as a combination of $\phi(2x-k)$ terms. The sum is governed by an H based term in (2.4.34),

$$e^{i\xi/2}\, \overline{H(\xi/2+\pi)} = \frac{1}{\sqrt{2}} \sum_k \overline{h_k}\, e^{ik(\xi/2+\pi)}\, e^{i\xi/2}. \tag{2.4.42}$$

Rearranging that expression, and substituting $-k-1$ for k, leads to

$$\begin{aligned} e^{i\xi/2}\,\overline{H(\xi/2+\pi)} &= \frac{1}{\sqrt{2}}\sum_k \overline{h_k}\,e^{ik\xi/2+i\xi/2}\,(e^{i\pi})^k \\ &= \frac{1}{\sqrt{2}}\sum_k (-1)^k\,\overline{h_k}\,e^{i(k+1)\xi/2} = \frac{1}{\sqrt{2}}\sum_k (-1)^{k+1}\,\overline{h_{-k-1}}\,e^{-ik\,\xi/2}. \end{aligned} \tag{2.4.43}$$

Using (2.4.42), (2.4.34) can be re-written as

$$\widehat{\psi}(\xi) = \frac{1}{\sqrt{2}}\sum_k (-1)^{k+1}\,\overline{h_{-k-1}}\,e^{-ik\xi/2}\,\widehat{\phi}(\xi/2). \tag{2.4.44}$$

Using fairly standard Fourier transform identities we get

$$\psi(x) = \sqrt{2}\sum_k (-1)^{k-1}\,\overline{h_{-k-1}}\,\phi(2x-k). \tag{2.4.45}$$

To save space, we will define coefficients $g_k = (-1)^{k-1}\,\overline{h_{-k-1}}$, making

$$\psi(x) = \sqrt{2}\sum_k g_k\,\phi(2x-k). \tag{2.4.46}$$

Notice that this gives us a compactly supported ψ if ϕ is compact.

So, we can now define the family of functions

$$\psi_{j,k}(x) = 2^{-j/2}\,\psi\left(\frac{x}{2^j}-k\right), \qquad j,k\in\mathbb{Z} \tag{2.4.47}$$

with each set $\{\psi_{j,k}|k\in\mathbb{Z}\}$ being an orthonormal basis for W_j since $D_2(W_{k+1}) = W_k$ (D_2 is the standard operator from $f(x)$ to $f(2x)$) (see the earlier work with D in Section 2.2).

Example 2.4.7 *The Haar wavelet*

$$\psi_H(x) = \begin{cases} 1 & x\in(0,1/2] \\ -1 & x\in(1/2,1] \\ 0 & x\notin(0,1] \end{cases} \tag{2.4.48}$$

can be derived from its scaling function

$$\phi_H(x) = \begin{cases} 1 & x\in(0,1] \\ 0 & x\notin(0,1]. \end{cases} \tag{2.4.49}$$

Proof.

We will avoid the Fourier space, since most of our knowledge of ψ_H relates to the function itself.

The scaling step for ϕ_H is simple:

$$\phi_H(x) = \phi_H(2x) + \phi_H(2x-1) = \sqrt{2}\left(\frac{1}{\sqrt{2}}\,\phi_H(2x) + \frac{1}{\sqrt{2}}\,\phi_H(2x-1)\right), \qquad (2.4.50)$$

so $h_0 = h_1 = \frac{1}{\sqrt{2}}$.

Using Equation (2.4.45) we get

$$\begin{aligned}\psi(x) &= \sqrt{2}\sum_k (-1)^{k-1}\, h_{-k-1}\,\phi(2x-k)\\ &= \sqrt{2}\left[(-1)^{-2}\, h_0\,\phi_H(2x+1) + (-1)^{-3}\, h_1\,\phi(2x+2)\right]\\ &= \frac{1}{2}\,\phi_H(2x+1) - \frac{1}{2}\,\phi_H(2x+2). \qquad (2.4.51)\end{aligned}$$

This is not precisely how the Haar wavelet is usually defined. It is located an integer to the left and multiplied by -1. However, the ψ above is orthogonal to $\phi_H(x-k)$ for all k, and combines with V_0 to create V_{-1}, so it is a valid choice. Changing to the function given in (2.4.48) preserves its necessary properties, and makes it easier to work with. ■

2.5 An Important Restriction

Before going on to the practicalities of calculating wavelets and scaling functions, and then on to actually using them on differential equations, we should discuss a particular restriction of orthogonal wavelets. This restriction on the behavior of any function $\psi \in \mathcal{S}$ is called the Uncertainty Principle, and, frequently, the Heisenberg Uncertainty Principle. The eventual result is

$$\|(x-x_0)\,\psi(x)\|\,\left\|(\xi-\xi_0)\,\widehat{\psi}(\xi)\right\| \geqslant \frac{1}{2}\,\|\psi\|^2\,, \qquad (2.5.1)$$

using the L^2 norm. This means that it is impossible for both ψ and $\widehat{\psi}$ to be well localized. The easiest way to see why is to view ψ as a probability distribution with

x_0 as the mean (and the same for $\widehat{\psi}$ with ξ_0). This would make $\|(x-x_0)\psi(x)\|$ very similar to a standard deviation, with the same true for $\widehat{\psi}$. Recall the direct correlation between the differentiability of ψ and the speed of convergence of $\widehat{\psi}(\xi)$ to zero as $\xi \to \pm\infty$ (Section A.1). This result means that no wavelet can be both well localized and have high derivatives. The extreme on one side is the Haar wavelet, discontinuous but heavily localized. The other extreme would be infinitely differentiable and have a total lack of localization. The functions e^{ikx} (or $\cos(kx)$ and $\sin(kx)$) used in Fourier decomposition are good examples. Anyway, proving the result is not very difficult, but requires the result for $x_0 = \xi_0 = 0$ first.

Theorem 2.5.1 *For $\psi \in L^2$,*

$$\|x\,\psi(x)\|\,\left\|\xi\,\widehat{\psi}(\xi)\right\| \geqslant \frac{1}{2}\,\|\psi(x)\|^2\,. \tag{2.5.2}$$

Proof.

Take $\psi \in \mathcal{S}$. We will concentrate on the case where both norms on the left are finite. The other cases are trivial. Next,

$$\left\|\xi\,\widehat{\psi}(\xi)\right\| = \left\|\widehat{\psi'(x)}\right\| = \|\psi'(x)\| \tag{2.5.3}$$

using standard Fourier transform identities for derivatives. So, we have

$$\begin{aligned}\|x\,\psi(x)\|\,\left\|\xi\,\widehat{\psi}(\xi)\right\| = \|x\psi(x)\|\,\|\psi'(x)\| &\geqslant |\langle x\psi(x), \psi'\rangle| \\ &\geqslant |\mathrm{Re}\,\langle x\psi(x), \psi'(x)\rangle| \\ &= \frac{1}{2}\big|\,\langle x\psi(x), \psi'(x)\rangle + \langle \psi'(x), x\psi(x)\rangle\,\big| \end{aligned} \tag{2.5.4}$$

We have to take a look at the resulting integral, so

$$\begin{aligned}\|x\,\psi(x)\|\,\left\|\xi\,\widehat{\psi}(\xi)\right\| &\geqslant \frac{1}{2}\left|\int x\big(\psi(x)\,\overline{\psi'(x)} + \psi'(x)\,\overline{\psi(x)}\big)\,dx\right| \\ &= \frac{1}{2}\left|x\,|\psi(x)|^2\,\big|_{-\infty}^{\infty} - \int |\psi(x)|^2\,dx\right| \;= \frac{1}{2}\,\|\psi(x)\|^2 \end{aligned} \tag{2.5.5}$$

using integration by parts and the fact that $x|\psi(x)| \to 0$ as $x \to \pm\infty$.

Since $\mathcal{S}$ is dense in L^2, we get Equation (2.5.2) for all of L^2. ■

Now to extend that to the full result.

Corollary 2.5.2 *For $\psi \in L^2$,*

$$\|(x - x_0)\,\psi(x)\| \left\|(\xi - \xi_0)\,\widehat{\psi}(\xi)\right\| \geqslant \frac{1}{2}\,\|\psi\|^2 . \tag{2.5.6}$$

Proof.

Define $g(x) = e^{-i\xi_0 x}\psi(x + x_0)$. This implies

$$\|g\|^2 = \int e^{-i\xi_0 x}\,\psi(x + x_0)\,e^{i\xi_0 x}\,\overline{\psi(x + x_0)}\,dx = \|\psi\|^2 . \tag{2.5.7}$$

Also,

$$\|xg\|^2 = \int x^2\,|\psi(x + x_0)|\;dx = \int (x - x_0)^2\,|\psi(x)|^2\;dx \tag{2.5.8}$$

using a simple change of variable. Similar steps can be taken for $\widehat{g}$ and $\widehat{f}$, which just need to be placed into Theorem 2.5.1. ■

This restriction proves the inverse correlation between smoothness and localization. As always, one cannot have everything. As wavelets and scaling functions become smoother, they also start becoming less localized.

Chapter 3

Calculating Scaling Functions and Wavelets

As before, we will be following the (general) route taken by [5], with the same, standard, results. Further reading can be found in [14], as well as any other book on wavelets. In this chapter we will find ways to calculate scaling functions, written ϕ, and then their associated wavelets, written ψ. The calculations will be heavily based upon what we have called the 'scaling step,' as well as upon the associated function H (Definition 2.4.1) and related results from Chapter 2.

3.1 Basics

We want to calculate scaling functions, written ϕ, which have the following properties:

$$a) \qquad \phi \in L^2(\mathbb{R}),\ \text{supp}(\phi) \text{ compact} \tag{3.1.1}$$

$$b) \qquad \phi(t) = \sqrt{2} \sum_k h_k\, \phi(2t-k), \ \text{ almost everywhere, for } h_k \in \mathbb{R} \tag{3.1.2}$$

$$c) \qquad \int \phi(t)\, dt = 1 \tag{3.1.3}$$

$$d) \qquad \int \phi(t)\, \overline{\phi(t-k)}\, dt = \delta_{0k} \tag{3.1.4}$$

Part b above is what we call the scaling step, the property of ϕ that allows this function to be written in terms of higher resolution version of itself. The scaling step

coefficients h_k are going to be our focus for this chapter. We have already seen that if ϕ is compactly supported then only finitely many h_k are non-zero (Theorem 2.3.4). In Chapter 2, it was helpful to look at the Fourier transform of ϕ, $\widehat{\phi}$, and at the Fourier transformed scaling equation:

$$\widehat{\phi}(\xi) = \frac{1}{\sqrt{2\pi}} \sum_k h_k \, e^{-ik\xi} \, \widehat{\phi}\left(\frac{\xi}{2}\right). \tag{3.1.5}$$

This can be further re-written as simply $\widehat{\phi}(\xi) = H(\xi/2)\,\widehat{\phi}(\xi/2)$, using

$$H(\xi) = \frac{1}{\sqrt{2}} \sum_k h_k \, e^{-ik\xi}. \tag{3.1.6}$$

A further advantage of a compact support for ϕ is that H will simply be a complex trigonometric polynomial, one that (as we have seen in Theorem 2.4.2) satisfies

$$|H(\xi)|^2 + |H(\xi+\pi)|^2 = 1 \qquad \text{almost everywhere,} \tag{3.1.7}$$

with $H(0) = 1$ and $H(\pi) = 0$. Actually, we can get more out of H.

Theorem 3.1.1 *Using an orthonormal scaling function ϕ (Section 2.3) and associated H (Equation (3.1.6)) we get*

$$\widehat{\phi}(\xi) = \frac{1}{\sqrt{2\pi}} \lim_{n\to\infty} \prod_{j=1}^{n} H\left(\frac{\xi}{2^j}\right). \tag{3.1.8}$$

Proof.

Equation (2.4.4) gives us $\widehat{\phi}(\xi) = H(\xi/2)\,\widehat{\phi}(\xi/2)$, which we can iterate for the expression

$$\widehat{\phi}(\xi) = \lim_{n\to\infty} \left[\prod_{j=1}^{n} H\left(\frac{\xi}{2^j}\right)\right] \widehat{\phi}\left(\frac{\xi}{2^n}\right). \tag{3.1.9}$$

Next, we include the fact that $\widehat{\phi}(\xi) \to \frac{1}{\sqrt{2\pi}}$ as $\xi \to 0$. ■

We have $\widehat{\phi}$ and, therefore, ϕ, defined entirely based upon the function H. The problem is that we have not yet confirmed the convergence of (3.1.8). First, we will want

$$|H'(\xi)| \leqslant M \qquad \Longrightarrow \qquad \left|H\left(\frac{\xi}{2^j}\right)\right| \leqslant M\left|\frac{\xi}{2^j}\right| + 1, \tag{3.1.10}$$

which will be true if there are only finitely many $h_k \neq 0$.

Now we take a look at calculating $\widehat{\phi}$ using only the associated function H.

Lemma 3.1.2 *If H satisfies (3.1.10) and (3.1.7), is $C^1(\mathbb{R})$ and has $H(0) = 1$, then*

$$\frac{1}{\sqrt{2\pi}} \lim_{n\to\infty} \prod_{j=1}^{n} H\left(\frac{\xi}{2^j}\right) \tag{3.1.11}$$

converges locally uniformly to the continuous function $\widehat{\phi}$.

Proof.

We know (3.1.11) is equal to $\widehat{\phi}$, due to Theorem 3.1.1. Next, we take $M = \max|H'(\xi)|$ which leads to

$$|H(\xi) - 1| \leqslant |H(\xi) - H(0)| \leqslant M\,|\xi| \quad \Longrightarrow \quad \left|H\left(\frac{\xi}{2}\right) - 1\right| \leqslant \frac{M}{2^j}\,|\xi|\,. \tag{3.1.12}$$

Since $\sum_{j>0} \frac{1}{2^j} = 1$, we get (3.1.11) converging uniformly to a continuous function, in this case, $\widehat{\phi}$. ■

Lemma 3.1.3 *Define a new set of functions $\widehat{f}_r$, $r = 0, 1, 2, \ldots$, with*

$$\widehat{f}_0 = \frac{1}{\sqrt{2\pi}}\,\mathbf{1}_{[-\pi,\pi]}(\xi), \qquad \widehat{f}_r(\xi) = H\left(\frac{\xi}{2}\right)\widehat{f}_{r-1}\left(\frac{\xi}{2}\right), \tag{3.1.13}$$

*with **1** representing an indicator function (see glossary).*

We get $\widehat{\phi} = \lim_{r\to\infty} f_r$. Furthermore, for every $r \geqslant 0$, the $\{f_r(x-k), k \in \mathbb{Z}\}$ are an orthonormal set.

Proof.

The limit of these as $r \to \infty$ is clearly $\widehat{\phi}(\xi)$, and the convergence is locally uniform, via Lemma 3.1.2. Next, we need $\{f_r(\cdot - k), k \in \mathbb{Z}\}$ to be an orthonormal system. We will, as before, use the result of Theorem 2.3.5, so what we need is

$$\sum_k \left|\widehat{f}_r(\xi + 2\pi k)\right|^2 = \frac{1}{2\pi} \quad \text{almost everywhere.} \tag{3.1.14}$$

First, consider $\widehat{f}_0$, equal to $\frac{1}{\sqrt{2\pi}}$ on $[-\pi, \pi[$ and zero elsewhere, so

$$\sum_k \left|\widehat{f}_0(\xi + 2\pi k)\right|^2 = \frac{1}{2\pi} \quad \forall\,\xi \in \mathbb{R}. \tag{3.1.15}$$

Next, assume that $\widehat{f}_{r-1}$ has this property, and proceed by induction:

$$\begin{aligned}
&\sum_k \left|\widehat{f}_r(\xi+2\pi k)\right|^2 \\
&= \sum_k \left|\widehat{f}_r(\xi+4\pi k)\right|^2 + \sum_k \left|\widehat{f}_r\left(\xi+2\pi+4\pi k\right)\right| \\
&= \sum_k \left|H\left(\frac{\xi}{2}+2\pi k\right)\widehat{f}_{r-1}\left(\frac{\xi}{2}+2\pi k\right)\right|^2 + \sum_k \left|H\left(\frac{\xi}{2}+\pi+2\pi k\right)\widehat{f}_{r-1}\left(\frac{\xi}{2}+\pi+2\pi k\right)\right|^2 \\
&= \left|H\left(\frac{\xi}{2}\right)\right|^2 \sum_k \left|\widehat{f}_{r-1}\left(\frac{\xi}{2}+2\pi k\right)\right|^2 + \sum_k \left|H\left(\frac{\xi}{2}+\pi\right)\right|^2 \sum_k \left|\hat{f}_{r-1}\left(\frac{\xi}{2}+\pi+2\pi k\right)\right|^2 \\
&= \left(\left|H\left(\frac{\xi}{2}\right)\right|^2 + \left|H\left(\frac{\xi}{2}+\pi\right)\right|^2\right)\frac{1}{2\pi} = \frac{1}{2\pi}, \qquad (3.1.16)
\end{aligned}$$

using the fact that H is 2π periodic, as well as Equation (3.1.7). So each set $\{f_r(\cdot-k), k\in\mathbb{Z}\}$ forms an orthonormal system. ■

Theorem 3.1.4 *If H satisfies Equations (3.1.10) and (3.1.7), is $C^1(\mathbb{R})$ and has $H(0)=1$, then*

$$\widehat{\phi}(\xi) = \frac{1}{\sqrt{2\pi}} \lim_{n\to\infty} \prod_{j=1}^{n} H\left(\frac{\xi}{2^j}\right) \in L^2, \qquad (3.1.17)$$

the product converges locally uniformly, and $\widehat{\phi}$ is continuous.

Proof.

Most of the theorem is proven in Lemma 3.1.2. All that is left is confirming that $\widehat{\phi}\in L^2$. We do this via the f_r functions from Lemma 3.1.3. Recalling that $\|f_r\|_2 = 1$,

$$\int \left|\widehat{\phi}(\xi)\right|^2 d\xi \leqslant \liminf_{r\to\infty} \int \left|\widehat{f}_r(\xi)\right|^2 d\xi = \liminf_{r\to\infty} 1 = 1, \qquad (3.1.18)$$

using Fatou's Lemma (explained in [22, p. 86]). So, $\widehat{\phi}\in L^2$. ■

We have a means of calculating ϕ using a fixed H, which means a fixed set of h_k. Now we need to make sure that we can get compact support. Recall that in Theorem 2.3.4 we used values a and b of functions f defined as

$$a(f) = \inf_x\{x|f(x)\neq 0\} \qquad b(f) = \sup_x\{x|f(x)\neq 0\} \qquad (3.1.19)$$

and proved that for reasonable assumptions, a compactly supported ϕ would have $a(\phi)$ and $b(\phi)$ in $\mathbb{Z}$. We assume now that $a(\phi) = 0$ and $b(\phi) = 2N - 1$ for $N \in \mathbb{N}$. Now, for a valid set of coefficients h_k, ϕ is actually equivalent to a fixed point of the operator $S : L^2 \to L^2$ defined

$$Sg(x) = \sqrt{2} \sum_{k=0}^{2N-1} h_k \, g(2x - k). \tag{3.1.20}$$

If we have a function $g(t)$ with support in $[0, 2N - 1]$ then $Sg(x) = 0$ for $x < 0$. Now let us consider what we get for $x > 2N - 1$ and any given h_k term in the sum. The argument $2x - k$ inside a given $g(2x - k)$ will be restricted to

$$2x - k \; > \; 2(2N - 1) - k \; = \; 4N - 2 - k \; \geqslant \; 4N - 2 - (2N - 1) \; = \; 2N - 1 \tag{3.1.21}$$

for any k within $[0, 2N - 1]$ and $x > 2N - 1$. All other k will result in an h_k of zero. As a result, if we start the process using a function with support in $[0, 2N - 1]$ then each element in the series will have support in $[0, 2N - 1]$. There are still a few things we do not know for certain. The first is whether we can simply iterate this operator on, say, the Haar scaling function, and get ϕ at the end. The best way is to take the Fourier transform of Sg,

$$\widehat{Sg}(\xi) = H\left(\frac{\xi}{2}\right) \widehat{g}\left(\frac{\xi}{2}\right), \tag{3.1.22}$$

with the iterations leading to

$$\lim_{r \to \infty} \left[\prod_{j=1}^{r} H\left(\frac{\xi}{2^j}\right) \right] \widehat{g}_0\left(\frac{\xi}{2^r}\right) \tag{3.1.23}$$

which is going to be simply

$$\prod_{j=1}^{\infty} H\left(\frac{\xi}{2^j}\right) \widehat{g}_0(0). \tag{3.1.24}$$

So as long as we start with a function g_0 which has an L^1 norm of 1 (the Haar scaling function is perfect), this will converge to $\frac{1}{\sqrt{2\pi}} \sum_j H(\xi/2^j) = \widehat{\phi}(\xi)$. Next we have to confirm that the support will be compact.

Theorem 3.1.5 *If only a finite number of the coefficients h_k in H are non-zero, and*

$$\widehat{\phi}(\xi) = \lim_{r \to \infty} \widehat{g}_r = \lim_{r \to \infty} \prod_{j=1}^{r} H\left(\frac{\xi}{2^j}\right) \widehat{g}_0\left(\frac{\xi}{2^r}\right), \tag{3.1.25}$$

then ϕ has compact support within $[0, 2N-1]$.

Proof.

First, the g_k will converge locally uniformly due to the properties of H (see Theorem 3.1.4).

Take $u \in C^2$ with compact support that does not intersect $(0, 2N-1)$ and an $\epsilon > 0$. What we are after is

$$\langle \phi, u \rangle = \langle g_k, u \rangle + \langle \phi - g_k, u \rangle = \left\langle \widehat{\phi} - \widehat{g}_k, \widehat{u} \right\rangle, \tag{3.1.26}$$

since the supports of g_k and u are disjoint. Due to the properties of Fourier transforms, we can find $M > 0$ such that

$$\int_{|\xi| \geqslant M} |\widehat{u}(\xi)| \, d\xi \leqslant \epsilon. \tag{3.1.27}$$

Next, using the locally uniform convergence of the $\hat{g}_k$ terms we can find $r \geqslant 0$ such that

$$\left| \widehat{g}_r - \widehat{\phi}(\xi) \right| \leqslant \epsilon \qquad \forall \, |\xi| \leqslant M. \tag{3.1.28}$$

Next, we will break up the integral from the inner product in Equation (3.1.26) for

$$\begin{aligned} |\langle \phi, u \rangle| &\leqslant \int_{-M}^{M} \left| \widehat{\phi}(\xi) - \widehat{g}_r(\xi) \right| |\widehat{u}(\xi)| \, d\xi + \int_{|\xi|>M} \left(\left| \widehat{\phi}(\xi) \right| + |\widehat{g}_r(\xi)| \right) |\widehat{u}(\xi)| \, d\xi \\ &\leqslant \int_{-M}^{M} \epsilon \, |\widehat{u}(\xi)| \, d\xi + \int_{|\xi|>M} \left(\left| \widehat{\phi}(\xi) \right| + |\widehat{g}_r(\xi)| \right) \epsilon \, d\xi \\ &= \left(\|\widehat{u}\|_1 + \left\| \widehat{\phi} \right\|_1 + \|\widehat{g}_r\|_1 \right) \epsilon. \end{aligned} \tag{3.1.29}$$

We chose an arbitrary ϵ, and the other terms are constant or obviously bounded (the g_r converge uniformly to ϕ), so the result is $\langle u, \phi \rangle = 0$ for all u with support outside $[0, 2N-1]$, which implies the support of ϕ is in $[0, 2N-1]$. ■

Now we take another look at orthonormality. Recall Theorem 2.4.2, where we came up with a necessary condition for the orthonormality of ϕ. We will now work out sufficient conditions.

Theorem 3.1.6 *Take $H \in C^1$, with $H(0) = 1$, $|H(\xi)|^2 + |H(\xi+\pi)|^2 = 1$ almost everywhere. Furthermore, have $H(\xi) \neq 0$ for all $|\xi| \leqslant \frac{\pi}{2}$. Taken together, this implies that $\{\phi_{0,k}, k \in \mathbb{Z}\}$ is an orthonormal basis of V_0.*

This is mostly based on the Lebesgue theorem, sometimes referred to as the Dominated convergence Theorem [22, p. 91]. The theorem is based on a sequence of measurable functions $\{f_n\}_{n\in\mathbb{N}}$ such that $|f_n| \leqslant g$ almost everywhere for integrable function g. Under those assumptions, if $f(x) = \lim_{n\to\infty} f_n(x)$ then we get

$$\int f(x)\,dx = \int \lim_{n\to\infty} f_n(x)\,dx = \lim_{n\to\infty} \int f_n(x)\,dx. \tag{3.1.30}$$

Proof.

We start with

$$\widehat{\phi}(\xi) = H(\xi/2)\widehat{\phi}(\xi/2) \quad \Longrightarrow \quad \widehat{\phi}(\xi) = \prod_{j=1}^{r} H\left(\frac{\xi}{2^j}\right)\widehat{\phi}\left(\frac{\xi}{2^r}\right). \tag{3.1.31}$$

If we have $|\xi| \leqslant \pi$ then $H\left(\frac{\xi}{2^j}\right) \neq 0$ for all $j \geqslant 1$ so $\widehat{\phi}(\xi) \neq 0$. The convergence to $\widehat{\phi}$ is locally uniform and $\widehat{\phi}$ is continuous (via Theorem 3.1.4). Also, our assumptions on H mean that $\widehat{\phi}(\xi) \neq 0$ for $\xi \leqslant \pi$. This means that $\delta > 0$ exits such that $\widehat{\phi}(\xi) \geqslant \delta > 0$ on $|\xi| \leqslant \pi$ (we get $\delta > 0$ since $\widehat{\phi}(0) > 0$).

Next, we will bring out the functions f_r as defined in (3.1.13), and note that we could easily write them as

$$\widehat{f}_r(\xi) = \begin{cases} \frac{1}{\sqrt{2\pi}} \frac{\widehat{\phi}(\xi)}{\widehat{\phi}(\xi/2^r)}, & \xi \in [-2^r\pi, 2^r\pi] \\ 0, & \xi \notin [-2^r\pi, 2^r\pi]. \end{cases} \tag{3.1.32}$$

Notice that $|\epsilon/2^r| \leqslant \pi$ for $\xi \in [-2^r\pi, 2^r\pi]$, so $\widehat{\phi}(\xi/2^r) \geqslant \delta$ when $\xi \in [-2^r\pi, 2^r\pi]$. The other intervals have $\widehat{f}_r = 0$, so we get the upper bound

$$\widehat{f}_r(\xi) \leqslant \frac{1}{\sqrt{2\pi}} \frac{\widehat{\phi}(\xi)}{\delta}. \tag{3.1.33}$$

We know that $\widehat{\phi}$ is a bounded L^2 function, so

$$\begin{aligned} \int \phi(t)\overline{\phi(t-k)}\,dt &= \int \widehat{\phi}(t)\overline{\widehat{\phi}(t)}\,e^{ikt}\,dt \\ &= \int \lim_{r\to\infty} \widehat{f}_r(\xi)\overline{\widehat{f}_r(\xi)}\,e^{ik\xi}\,dt = \lim_{r\to\infty} \int \widehat{f}_r(\xi)\overline{\widehat{f}_r(\xi)}\,e^{ik\xi}\,dt \end{aligned} \tag{3.1.34}$$

which is δ_{0k} because the $\{f_r(\cdot - k)\}_{k\in\mathbb{Z}}$ are an orthonormal set (Theorem 3.1.3). ■

3.2 Calculating the Coefficients h_k

In Section 3.1, it was determined that the coefficients h_k of a given scaling function ϕ could be used, on their own, to calculate ϕ. Recall that the coefficients h_k are from the scaling step,

$$\phi(x) = \sqrt{2} \sum_k h_k \, \phi(2x - k). \tag{3.2.1}$$

We will, eventually, be calculating ϕ based on the h_k, so the first step is to find them. Recall the function H (Definition 2.4.1),

$$H(\xi) = \frac{1}{\sqrt{2}} \sum_k h_k \, e^{-ik\xi}. \tag{3.2.2}$$

Recall that $H(0) = 1$, equivalent to the h_k summing to $\sqrt{2}$ (Equation (2.3.4)). Since we want $\phi$to be a real function, we will insist on $h_k \in \mathbb{R}$. Now for a closer look at ψ. Equation (2.4.34)) shows that

$$\widehat{\psi}(\xi) = e^{i\frac{\xi}{2}} \overline{H\left(\frac{\xi}{2} + \pi\right)} \, \widehat{\phi}\left(\frac{\xi}{2}\right). \tag{3.2.3}$$

The order of ψ relates to the convergence of the wavelet transform (detailed in the appendix, Section B.1). The higher the order, the more efficiently a finite resolution decomposition will model a function. By definition (Section B.1, again), the order of ψ relates directly with how quickly $\widehat{\psi}(\xi)$ converges to 0 as $\xi \to 0$. Equation (3.2.3) makes it clear that the convergence in question is tied to how quickly $H(\xi)$ converges to zero as $\xi \to \pi$. As a result, we will make functions H of the form

$$H(\xi) = \left(\frac{1 + e^{-i\xi}}{2}\right)^N B(\xi) \tag{3.2.4}$$

with B some polynomial of $e^{i\xi}$. This gives us order N, at least.

Next we will deal with the requirement that $|H(\xi)|^2 + |H(\xi + \pi)|^2 = 1$ almost everywhere, from Theorem 2.4.2, which only requires looking closely at $|H(\xi)|^2$. Notice first that

$$|H(\xi)|^2 = H(\xi)\overline{H(\xi)} = H(\xi)H(-\xi) = |H(-\xi)|^2$$

so $|H(\xi)|^2$ must be an even function, so it will be a polynomial of $\cos\xi$. Substituting the H from Equation (3.2.4) gives us

$$\begin{aligned}|H(\xi)|^2 &= \left(\frac{1+e^{-i\xi}}{2}\right)^N \left(\frac{1+e^{i\xi}}{2}\right)^N B(\xi)B(-\xi) \\ &= \left[\cos^2\left(\frac{\xi}{2}\right)\right]^N B(\xi)\overline{B(\xi)} = \left[1-\sin^2\left(\frac{\xi}{2}\right)\right]^N |B(\xi)|^2 . \end{aligned} \tag{3.2.5}$$

Since $|H|^2$ is symmetric around zero (an even function), and so is $\cos^2(\xi/2)$, then $|B(\xi)|^2$ must be as well. So, $|B(\xi)|^2$ is a polynomial of $\cos\xi = 1-2\sin^2\left(\frac{\xi}{2}\right)$ and can be written $|B(\xi)|^2 = P\left(\sin^2\left(\frac{\xi}{2}\right)\right)$ for a polynomial P. One more identity, $\sin^2\left(\frac{\xi+\pi}{2}\right) = \cos^2\left(\frac{\xi}{2}\right)$ leads to the expression

$$\begin{aligned} 1 &= |H(\xi)|^2 + |H(\xi+\pi)|^2 \\ &= \left[1-\sin^2\left(\frac{\xi}{2}\right)\right]^N P\left(\sin^2\left(\frac{\xi}{2}\right)\right) + \left[1-\cos^2\left(\frac{\xi}{2}\right)\right] P\left(\cos^2\left(\frac{\xi}{2}\right)\right) \\ 1 &= \left[1-\sin^2\left(\frac{\xi}{2}\right)\right]^N P\left(\sin^2\left(\frac{\xi}{2}\right)\right) + \sin^2\left(\frac{\xi}{2}\right) P\left(1-\sin^2\left(\frac{\xi}{2}\right)\right) \end{aligned} \tag{3.2.6}$$

Substituting $y = \sin^2\left(\frac{\xi}{2}\right)$ simplifies the expression to

$$1 = (1-y)^N P(y) + (y)^N P(1-y), \qquad y \in [0,1], \tag{3.2.7}$$

with our underlying functions written as

$$H(\xi)\overline{H(\xi)} = (1-y)^N P(y). \tag{3.2.8}$$

Now we need to solve for P, which is the only unknown in our expression for $|H|^2$. The next step is to divide both sides by $y^N(1-y)^N$, leading to

$$\frac{1}{y^N(1-y)^N} = \frac{P(y)}{y^N} + \frac{P(1-y)}{(1-y)^N}. \tag{3.2.9}$$

Now, partial fractions give us constants $\{C_k\}_{k=0,\ldots,N}$ and $\{C'_k\}_{k=0,\ldots,N}$ for

$$\frac{1}{y^N(1-y)^N} = \sum_{k=0}^{N} \frac{C_k}{y^k} + \sum_{k=0}^{N} \frac{C'_k}{(1-y)^k} = \sum_{k=0}^{N} \frac{C_k y^{N-k}}{y^N} + \sum_{k=0}^{N} \frac{C'_k (1-y)^{N-k}}{(1-y)^N}. \tag{3.2.10}$$

Since those divisions were basically identical, we get $C_k = C'_k$. This results in a specific polynomial P_N of degree $N-1$ satisfying

$$(1-y)^N P_N(y) + y^N P_N(1-y) = 1 \Longleftrightarrow \frac{P_N(y)}{y^N} + \frac{P_N(1-y)}{(1-y)^N} = \frac{1}{y^N(1-y)^N}. \quad (3.2.11)$$

The polynomial is

$$P_N(y) = \sum_{k=0}^{N-1} \left(\begin{array}{c} N+k-1 \\ k \end{array} \right) y^k, \quad (3.2.12)$$

which combines into our new expression for H,

$$|H(\xi)|^2 = \left[1 - \sin\left(\frac{\xi}{2}\right)\right]^N P_N\left[\sin\left(\frac{\xi}{2}\right)\right]. \quad (3.2.13)$$

This expression will satisfy the $|H(\xi)|^2 + |H(\xi+\pi)|^2 = 1$ requirement, but is not the only solution.

Theorem 3.2.1 *A trigonometric polynomial M satisfies $M(\xi) + M(\xi+\pi) = 1$ if and only if*

$$M(\xi) = \left[\cos^2\left(\frac{\xi}{2}\right)\right]^N P\left(\sin^2\left(\frac{\xi}{2}\right)\right) \quad (3.2.14)$$

with $P(y) = P_N(y) + y^N R(1-2y)$, R an odd polynomial.

Proof.

If we have any function P that satisfies Equation (3.2.7) then

$$(1-y)^N\big(P(y) - P_N(y)\big) + y^N\big(P(1-y) - P_N(1-y)\big) = 0. \quad (3.2.15)$$

First, note that the second term is multiplied by a y^N, so N is the lowest degree of any of its non-zero coefficients. The same is true of the first term, meaning that we can write $P(y) - P_N(y) = y^N P^*(y)$, with P^* a polynomial. Substitute that into Equation (3.2.15) and the result is

$$\begin{aligned} (1-y)^N y^N P^*(y) + y^N (1-y)^N P^*(1-y) &= 0, \\ P^*(y) + P^*(1-y) &= 0. \end{aligned} \quad (3.2.16)$$

A change to $y + 1/2$ leads to $P^*(y + 1/2) = -P^*(1/2 - y)$, meaning that P^* is antisymmetric around $1/2$, and since it is a polynomial we get $P^*(y) = R(1-2y)$ for an odd polynomial R. ■

We now have a general expression for $|H(\xi)|^2$, so we will need to take the square root, in a manner of speaking. The first component is easy, but $P(\xi)$ will be a problem. Thankfully, we have something that at least gives the existence of the required factoring of $P(\xi)$.

Theorem 3.2.2 (Riesz' Lemma) *If*

$$A(\xi) = \sum_{k=0}^{n} a_k \cos^k(\xi),\ a_k \in \mathbb{R},\ a_n \neq 0, \tag{3.2.17}$$

and $A(\xi) \geqslant 0$ *for* $\xi \in \mathbb{R}$*, and* $A(0) = 1$*, then there exists*

$$B(\xi) = \sum_{k=0}^{n} b_k e^{-ik\xi},\ b_k \in \mathbb{R},\ B(0) = 1 \tag{3.2.18}$$

such that $A(\xi) = B(\xi)B(-\xi)$.

Proof.

This takes some work, and can be found in [5, p 171]. ■

So, our full result becomes

$$|H(\xi)|^2 = \left[\cos^2\left(\frac{\xi}{2}\right)\right]^N P\left(\sin^2\left(\frac{\xi}{2}\right)\right) = \left[\frac{1+e^{-i\xi}}{2}\right] B(\xi) \left[\frac{1+e^{i\xi}}{2}\right] B(-\xi), \tag{3.2.19}$$

recalling that

$$\cos^2\left(\frac{\xi}{2}\right) = \frac{1}{2}(1+\cos\xi) = \frac{1}{4}\left(2+2\cos\xi\right) = \frac{1}{4}\left(1+e^{i\xi}+e^{-i\xi}+1\right) = \left|\frac{1+e^{-i\xi}}{2}\right|^2. \tag{3.2.20}$$

This means that

$$H(\xi) = \left(\frac{1+e^{-i\xi}}{2}\right)^N B(\xi). \tag{3.2.21}$$

Daubechies wavelets are those that use $P = P_N$. We just take an $N > 0$, calculate $P_N(y)$, factor the resulting function $B(\xi)$, and substitute it into Equation (3.2.21) to calculate H. In terms of y, P_N is going to be of degree $N-1$. In terms of $e^{i\xi}$, the resulting $B(\xi)$ will also be degree $N-1$, since $y = \sin^2\left(\frac{\xi}{2}\right)$, degree 2 in relation to $e^{i\xi}$. The eventual product, H, will be degree $2N-1$. If we use $N = 1$ then B has a degree of zero and

$$H(\xi) = \frac{1}{2}\left(1 + e^{-i\xi}\right). \tag{3.2.22}$$

This creates the Haar wavelet.

Example 3.2.3 *The H function for the second order Daubechies wavelet is*

$$H(\xi) = \frac{1}{8}\left[\left(1+\sqrt{3}\right) + \left(3+\sqrt{3}\right)e^{i\xi} + \left(3-\sqrt{3}\right)e^{-2\xi} + \left(1-\sqrt{3}\right)e^{-3i\xi}\right]. \tag{3.2.23}$$

Proof.

To make a Daubechies wavelet we use an H from Equation (3.2.21) with some power N and the smallest B that is possible. For the $N = 2$ case it will be

$$H_2(\xi) = \left(\frac{1+e^{i\xi}}{2}\right)^2 B(\xi). \tag{3.2.24}$$

The function B will be a degree 1 trigonometric polynomial with

$$|B(y)|^2 = P_2(y) = \sum_{k=0}^{1}\begin{pmatrix} 1+k \\ k \end{pmatrix} y^k = 1 + 2y \tag{3.2.25}$$

with $y = \sin^2(\xi/2)$. That means

$$|B(\xi)|^2 = 1 + 2\left(\frac{1}{2} - \frac{1}{2}\cos\xi\right) = 2 - \cos\xi = 2 - \frac{1}{2}\left(e^{i\xi} + e^{-i\xi}\right). \tag{3.2.26}$$

We have $|B(\xi)|^2$, now we need $B(\xi)$ itself. It will be of the form $B(\xi) = a + be^{-i\xi}$, so

$$|B(\xi)|^2 = a^2 + b^2 + ab\left(e^{i\xi} + e^{-i\xi}\right) = 2 - \frac{1}{2}\left(e^{i\xi} + e^{-i\xi}\right), \tag{3.2.27}$$

leading to

$$B(\xi) = \left(\frac{1+\sqrt{3}}{2}\right) + \left(\frac{1-\sqrt{3}}{2}\right)e^{-i\xi} \quad \text{or} \quad \left(\frac{1-\sqrt{3}}{2}\right) + \left(\frac{1+\sqrt{3}}{2}\right)e^{-i\xi}. \tag{3.2.28}$$

If we use the first of those options for B we get

$$H_2(\xi) = \frac{1}{8}\left[\left(1+\sqrt{3}\right) + \left(3+\sqrt{3}\right)e^{i\xi} + \left(3-\sqrt{3}\right)e^{-2\xi} + \left(1-\sqrt{3}\right)e^{-3i\xi}\right]. \quad (3.2.29)$$

This leads to the coefficients

$$h_0 = \frac{1+\sqrt{3}}{4\sqrt{2}}, \quad h_1 = \frac{3+\sqrt{3}}{4\sqrt{2}}, \quad h_2 = \frac{3-\sqrt{3}}{4\sqrt{2}}, \quad h_3 = \frac{1-\sqrt{3}}{4\sqrt{2}}, \quad (3.2.30)$$

used in the H function (Definition 2.4.1). ■

3.3 Numerical Computation of the Wavelet and Scaling Function

So, we have gone through the whole mess of trigonometric polynomials and found a way to calculate the coefficients h_k for a given ϕ, before calculating ϕ itself. Approximating ϕ is our obvious next step. We will set up a method that will give the exact values of ϕ at any point of the form $x = k2^r$, $r, k \in \mathbb{Z}$. We will divide that set of points, which is dense on $\mathbb{R}$, into different resolutions

$$\mathbb{D}_r = \{k2^{-r} | k \in \mathbb{Z}\}, \quad r = 0, 1, 2, \ldots, \quad (3.3.1)$$

with

$$\mathbb{Z} = \mathbb{D}_0 \subset \mathbb{D}_1 \subset \cdots \subset \mathbb{D}_r \subset \cdots \subset \mathbb{D} = \bigcup_{r=0}^{\infty} \mathbb{D}_r \subset \overline{\mathbb{D}} = \mathbb{R}. \quad (3.3.2)$$

Here is the key observation: because

$$\phi(x) = \sqrt{2} \sum_{k\in\mathbb{Z}} h_k \, \phi(2x-k), \quad (3.3.3)$$

the value of ϕ in $\mathbb{D}_r$ is influenced only by the values in $\mathbb{D}_{r-1}$, so if we can get $\mathbb{D}_0$ then we can get $\mathbb{D}_1$ and so on. If we know ϕ is continuous, at least piecewise, then the fact that we have the exact values means we have a good, finite difference, approximation. Getting that first step is, as always, the difficult part. So, take $j \in \mathbb{Z}$, meaning that $\phi(j) = 0$ if $j \notin J = \{0, 1, 2, \ldots, 2N-1\}$. Recall that

$$\phi(j) = \sqrt{2} \sum_{k=0}^{2N-1} h_k \, \phi(2j-k), \quad (3.3.4)$$

with only the $2j-k$ terms in J resulting in a meaningful value. If we use a vector $\mathbf{x} = [\phi(0), \phi(1), \ldots, \phi(2N-1)]^T$ then we get the system $\mathbf{x} = \sqrt{2}\, M\, \mathbf{x}$, with M a $2N \times 2N$ matrix of the form

$$M = \begin{bmatrix} h_0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ldots & 0 \\ h_2 & h_1 & h_0 & 0 & 0 & 0 & 0 & 0 & \ldots & 0 \\ h_4 & h_3 & h_2 & h_1 & h_0 & 0 & 0 & 0 & \ldots & 0 \\ h_6 & h_5 & h_4 & h_3 & h_2 & h_1 & h_0 & 0 & \ldots & 0 \\ \vdots & & & & \ddots & & & & & \vdots \\ \vdots & & & & & \ddots & & & & \vdots \\ \vdots & & & & & & \ddots & & & \vdots \\ \vdots & & & & & & & \ddots & & \vdots \\ \vdots & & & & & & & & \ddots & \vdots \\ 0 & \ldots & & & & & \ldots & & 0 & h_{2N-1} \end{bmatrix}. \tag{3.3.5}$$

Notice that each row is a vector of coefficients h_k, simply in reverse order, and that the coefficient h_k is on the diagonal on the row $k-1$. The zeros in the matrix M are due to the fact that $h_k = 0$ for $k \notin [0, 2N-1]$.

If the h_k are valid, then we will get an eigenvalue of one, whose eigenvector will give the $\mathbb{D}_0$ approximation of ϕ. This will give a direction rather than a unique vector, so what we do is take the vector x such that $\sum_k x_k = 1$. That particular choice of vector is necessary, due to the following result.

Property 3.3.1 *Assume we have a valid H function (see Section 3.2). If ϕ is a continuous function and there exists $C \in \mathbb{R}$ such that*

$$|\phi(x)| \leqslant \frac{C}{1+x^2}, \quad \forall\, x \in \mathbb{R}, \tag{3.3.6}$$

then

$$g(x) = \sum_k \phi(x-k) = 1 \quad \forall\, x \in \mathbb{R}. \tag{3.3.7}$$

Proof.

This is not that difficult, thanks to all we know about ϕ. First, $g(x)$ is continuous, because ϕ is, and is also periodic over $\mathbb{Z}$. As a result, we can look at the Fourier series of $g(x)$, with coefficients

$$\begin{aligned} c_j = \int_0^1 g(x)e^{-2i\pi jx}\,dx &= \int_0^1 \sum_k \phi(x-k)e^{-2i\pi jx}\,dx \\ &= \sum_k \int_0^1 \phi(x-k)e^{-2i\pi j(x-k)}\,dx \\ &= \int \phi(x)e^{-2i\pi jx}\,dx = \sqrt{2\pi}\,\widehat{\phi}\,(2j\,\pi). \end{aligned} \tag{3.3.8}$$

Since $\widehat{\phi}(0) = \frac{1}{\sqrt{2\pi}}$ (see the beginning of Section 2.3), we get $c_0 = 1$. Also, notice that

$$\sum_j |c_j|^2 = \sum_j (2\pi)\left|\widehat{\phi}(2j\pi)\right|^2 = 2\pi\,\Phi(0) = 1, \tag{3.3.9}$$

using the Φ function from Theorem 2.3.5, and the fact that ϕ is an orthonormal scaling function. This means that c_0 is the only non-zero Fourier coefficient of $g(x)$, so $g(x)$ is constant. ■

Next we will calculate a classic scaling function and wavelet. Both Examples C.1.1 and C.1.2 relate to the methods used.

Example 3.3.2 *The $N = 2$ wavelet, the Daubechies wavelet ψ_2, and its scaling function.*

Recall Example 3.2.3, where we calculated the coefficients

$$h_0 = \frac{1+\sqrt{3}}{4\sqrt{2}}, \quad h_1 = \frac{3+\sqrt{3}}{4\sqrt{2}}, \quad h_2 = \frac{3-\sqrt{3}}{4\sqrt{2}}, \quad h_3 = \frac{1-\sqrt{3}}{4\sqrt{2}}, \tag{3.3.10}$$

for the second order Daubechies scaling function. If we take the matrix

$$A = \sqrt{2}\begin{bmatrix} h_0 & 0 & 0 & 0 \\ h_2 & h_1 & h_0 & 0 \\ 0 & h_3 & h_2 & h_1 \\ 0 & 0 & 0 & h_3 \end{bmatrix} \tag{3.3.11}$$

then we find that it has an eigenvalue of 1 with related eigenvector $\left[0, \frac{1+\sqrt{3}}{2}, \frac{1-\sqrt{3}}{2}, 0\right]^T$. So, those are our starting values. Now to make use of the equation

$$\begin{aligned}\phi(j) &= \sqrt{2}\sum_{k=0}^{2N-1} h_k\,\phi(2j-k)\\ &= \sqrt{2}\Big[h_0\,\phi(2j) + h_1\,\phi(2j-1) + h_2\,\phi(3j-2) + h_3\,\phi(2j-3)\Big].\end{aligned} \tag{3.3.12}$$

At the initial state we have four exact values of ϕ_2, evenly spaced on $[0, 3]$. This makes a $\Delta x = 1$. After one application of Equation (3.3.12) we will have seven, evenly spaced, for a $\Delta x = \frac{1}{2}$. Continuing on, each step will halve the Δx. There will be $3 \cdot 2^n + 1$, exact, values at the step n, that term resulting from the support of the function, $[\,0\,,\,3\,]$, and the resolution. The calculations can be arranged in matrix form, each step using a $3 \cdot 2^n + 1$ by $3 \cdot 2^{n-1} + 1$ sized matrix A at each step. For the second step, A will have the form

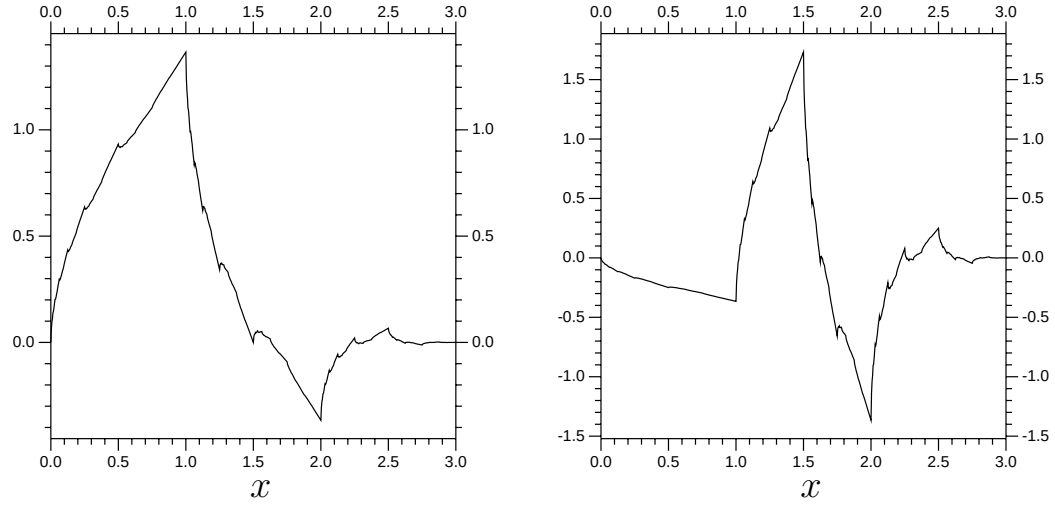
$$A = \begin{bmatrix} 1 & 0 & 0 & 0 \\ \sqrt{2}\,h_1 & \sqrt{2}\,h_0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \sqrt{2}\,h_3 & \sqrt{2}\,h_2 & \sqrt{2}\,h_1 & \sqrt{2}\,h_0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \sqrt{2}\,h_3 & \sqrt{2}\,h_2 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{3.3.13}$$

This pattern, with an identity matrix with its rows separated by terms of the form $\sqrt{2}\,h_k$ and zeros, will work for each step if we get the even rows right. Take the elements in the $(2j)$th row, for example. This row calculates

$$\phi\left(\frac{2j-1}{2^n}\right) = \sqrt{2}\sum_k h_k\,\phi\left(\frac{2j-1}{2^{n-1}} - k\right) = \sqrt{2}\sum_k h_k\,\phi\left(\frac{2j - k2^{n-1} - 1}{2^{n-1}}\right). \tag{3.3.14}$$

Recalling that the input vector is operating under the coarser resolution, this means that the $\sqrt{2}\,h_k$ will need to appear on the $2j - k\,2^{n-1}$ position, which is consistent with the pattern above.

A simple procedure in MATLAB (see Example C.1.1) can get a fair approximation of the scaling function, and can easily be used to gain an approximation of the



(a) $\phi_2(x)$ (b) $\psi_2(x)$

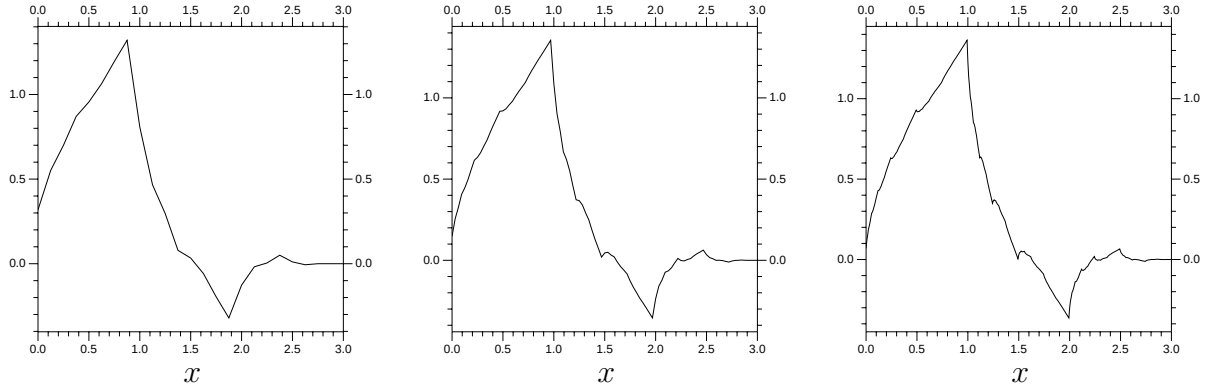
Figure 1: Plot of the second Daubechies scaling/wavelet function pair, at 2^{10} per unit resolution.

corresponding mother wavelet (Figure 1). One could also approximate the derivatives of these functions, weak derivatives, and so forth.

Using 10 steps, that is, an eventual step size of 2^{-10}, takes a nearly imperceptible amount of time on a standard, out of date, system. Furthermore, the eventual numerical integral of the resulting ϕ with $\phi(\cdot - 1)$ is approximately -5.2452×10^{-6}. The numerical approximation of $\|\phi\|^2$ has an error of 8.1062×10^{-6} (remember it is supposed to equal one). Results for that level of accuracy and several others, for comparison, are in Table 1. The value N, relating to the step size $\Delta x = \frac{1}{2^N}$, is in the top row.

Table 1: Inner product errors for ϕ_2.

	Resolution			
	4	6	8	10
$\|\phi\|^2 - 1$	1.43×10^{-2}	1.32×10^{-3}	1.06×10^{-4}	8.10×10^{-6}
$\langle \phi(\cdot), \phi(\cdot - 1) \rangle$	-8.95×10^{-3}	-8.38×10^{-4}	-6.83×10^{-5}	-5.24×10^{-6}
$\langle \phi(\cdot), \phi(\cdot - 2) \rangle$	1.78×10^{-3}	1.79×10^{-4}	1.52×10^{-5}	1.19×10^{-6}



(a) $\eta_3(x)$ (b) $\eta_5(x)$ (c) $\eta_7(x)$

Figure 2: Cascade Algorithm results, all calculated using a starting function of $\phi_H = \iota_{[0,1)}$. The only difference is the number of cascade algorithm cycles. The main feature to notice here is the tip of the tallest part of the function. Notice how it is left of $x = 1$ at three cycles, migrating to the right as more cycles are calculated.

3.4 The Cascade Method

There is a recurring problem with the $\phi(x)$ calculating method outlined in Section 3.3, and that problem is stability. Recall that the method starts with an eigenvalue of a matrix (see around Equation (3.3.5)). The matrix can have multiple eigenvectors for $\lambda = 1$, of course, and some of the eigenvectors will generate unstable results. This is generally easy to spot, when it happens. If the wrong eigenvector is chosen, the algorithm will output ϕ function data in the 10^4 range after a modest number of cycles. It is hard to prevent this from happening. We will now look into the cascade method, which is totally reliable when it comes to stability, though not quite as accurate as the earlier algorithm. For a more extensive look at the advantages of the cascade algorithm, look at [26, p. 55].

The cascade method is actually an application of something we used earlier, in Theorem 3.1.8. We wanted $\widehat{\phi}$, so we used

$$\lim_{r\to\infty} \prod_{j=0}^{r} H\left(\frac{\xi}{2^r}\right) \widehat{\phi}_H\left(\frac{\xi}{2^r}\right). \tag{3.4.1}$$

The cascade method works towards ϕ, not its Fourier transform. It does so using the scaling step for ϕ (last seen in (3.3.4)) to a carefully chosen starting function. Doing so repeatedly will result in convergence to ϕ. There are quite a few details to outline.

First, the algorithm calculates approximations of ϕ at 2^{-n} intervals, much like the

earlier algorithm. It does so via step functions of the form

$$\eta_n(x) = \sum_l c_l \mathbf{1}_{[2^{-n}l, 2^{-n}(l+1))}(x). \tag{3.4.2}$$

The η_n are step functions that are constant on intervals $\left[\, 2^{-n}k, 2^{-n}(k+1)\,\right)$, so

$$\eta_n(x) = \sum_l \eta_n(2^{-n}l)\mathbf{1}_{[2^{-n}l, 2^{-n}(l_1))}(x). \tag{3.4.3}$$

Also, for each η_n we get

$$\int \eta_n(x)\,dx = \sum_l \eta_n\left(2^{-n}\ l\right) 2^{-n}. \tag{3.4.4}$$

We only need to keep track of the values $\eta_n(2^{-n}l)$. Notice that the η_n functions will become finer as the algorithm progresses. This is necessary, due to the structure of the scaling step

$$\eta_{n+1}(x) = \sqrt{2}\sum_k h_k \eta_n(2x-k), \tag{3.4.5}$$

notice that the translates of $\eta_n(2x)$ will be constant on $\left[\, 2^{-n-1}l, 2^{-n-1}(l+1)\,\right)$ intervals.

The first function, the starting function, is η_0, an integer step function. If we want to calculate an orthonormal ϕ, we should have $\langle \eta_0, \eta_0(\cdot - l)\rangle = \delta_{0l}$ (this is easy, chose a translate of the Haar scaling function). We also require $\int \eta_0\,dx = 1$, which, again, is satisfied if we use some version of the Haar scaling function. If we have $\langle \eta_n, \eta_n(\cdot - l)\rangle = \delta_{0l}$ and the coefficients h_k from an orthonormal ϕ then

$$\begin{aligned}\langle \eta_{n+1}, \eta_{n+1}(\cdot - l)\rangle &= 2\sum_k h_k \sum_{k'} h_{k'} \int \eta_n(2(x-l)-k)\eta_n(2x-k')\,dx \\ &= \sum_k \sum_{k'} h_k h_{k'} \delta_{(k+2l)(k')} \ = \ \delta_{l0}\end{aligned} \tag{3.4.6}$$

using Equation (2.3.4). So, the η_n will preserve the orthogonality property and, more importantly, preserve the L^2. There is one more value that is preserved.

If we use the scaling step on η_n, we get

$$\eta_{n+1}(x) = \sqrt{2}\sum_k h_k \eta_n(x) = \sqrt{2}\sum_k h_k \sum_l \eta_n\left(2\cdot 2^{-n}l - k\right)\ \mathbf{1}_{[2^{-n}l, 2^{-n}(l+1))}(2x-k). \tag{3.4.7}$$

We can use this on the integral of η_{n+1}, from (3.4.4), for

$$\begin{aligned}\int \eta_{n+1}(x)\,dx &= \sqrt{2}\sum_l \eta_{n+1}(2^{-n-1}l)\frac{1}{2^{n+1}} \\ &= \sqrt{2}\sum_k h_k \sum_l \eta_n\left(2\cdot 2^{-n-1}l - k\right)\frac{1}{2^{n+1}} \\ &= \frac{1}{2^{n+1}}\left[\sqrt{2}\sum_k h_k \sum_l \eta_n\left(2^{-n}l - k\right)\right]. \end{aligned} \tag{3.4.8}$$

Since k is always a multiple of 2^{-n+1}, each coefficient h_k gets multiplied by each $\eta_n\left(2^{-n}l\right)$. As a result, the integral becomes

$$\int \eta_{n+1}(x)\,dx = \frac{1}{2^{n+1}}\left(\sqrt{2}\sum_k h_k\right)\left(\sum_l \eta_{n-1}\left(2^{-n}l\right)\right). \tag{3.4.9}$$

Since the coefficients h_k sum to $\sqrt{2}$ (see near Equation (2.3.5)), we get

$$\int \eta_{n+1}(x) = \frac{1}{2^n}\sum_l \eta_{n-1}\left(2^{-n}l\right) = \int \eta_n(x)\,dx, \tag{3.4.10}$$

by (3.4.4). So, we have the integral preserved and the L^2 norm preserved. Stability is helpful, but convergence is more important, of course. Thankfully, Theorem 3.1.8 proves convergence using the Fourier transformed equivalent to this procedure, so we will get convergence here as well. As before, the eventual convergence is to ϕ, independent of pretty much anything except that integral of the starting function should be 1.

The fact that we do not get exact terms for any given point is the necessary side effect for the method's stability. For instance, if we are calculating the Daubechies $n = 2$ wavelet (discussed in Example 3.3.2) starting with ϕ_H, we get a clear migration from left to right (top of Figure 2) (look at the tallest part of the function).

A simple cascade method program can be found in Appendix C, Example C.1.4.

Chapter 4

Biorthogonal Wavelets

The most stringent restriction we have applied until now was keeping the $\psi_{j,k}$ orthogonal. The scaling functions and wavelets created in this way are quite clearly not ideal for our purposes, being difficult to calculate explicitly, not symmetric, and, most importantly, not smooth enough to model differentiable functions effectively. What we will now do is relax the orthogonality requirement, creating a class of scaling functions and wavelets that are symmetric, smooth, and have compact support.

4.1 Biorthogonal Decompositions

What we want is a set of two bases, each the dual of the other. The dual of a function f, $\widetilde{f}$, has the property $\left\langle f, \widetilde{f} \right\rangle = 1$. When applied to sets the requirement becomes more stringent. In the case of wavelets, we get two sets

$$\{\psi_{j,k}\}_{j,k\in\mathbb{Z}} \quad \text{and} \quad \left\{\widetilde{\psi}_{j',k'}\right\}_{j',k'\in\mathbb{Z}}, \tag{4.1.1}$$

such that both sets span our L^2 space, and, finally,

$$\left\langle \psi_{j,k}\,,\, \widetilde{\psi}_{j',k'} \right\rangle = \delta_{j,j'}\,\delta_{k,k'}. \tag{4.1.2}$$

The inner product requirement in Equation (4.1.2) is key, since it means that if

$$f = \sum_{j,k} a_{j,k}\psi_{j,k} = \sum_{j,k} \widetilde{a}_{j,k}\widetilde{\psi}_{j,k} \tag{4.1.3}$$

then

$$\left\langle f\,,\,\widetilde{\psi}_{j',k'}\right\rangle = \left\langle \sum_{j,k} a_{j,k}\psi_{j,k}\,,\,\widetilde{\psi}_{j',k'}\right\rangle = \sum_{j,k} a_{j,k}\left\langle \psi_{j,k}\,,\,\widetilde{\psi}_{j',k'}\right\rangle = a_{j,k}. \tag{4.1.4}$$

As a result, we can actually write

$$f = \sum_{j,k}\left\langle \widetilde{\psi}_{j,k}\,,\,f\right\rangle \psi_{j,k} = \sum_{j,k}\langle \psi_{j,k}\,,f\rangle\,\widetilde{\psi}_{j,k}. \tag{4.1.5}$$

Note that we started with f equal to a sum of ψ terms. It would be nice if projections could be calculated using inner products. Unfortunately, it is not that simple for biorthogonal bases. If we set up a space $V = \text{Span}\{f_k\,,\,k \in K\}$, then assuming the projection of h onto V is

$$P_V h = \sum_{k\in K}\left\langle \widetilde{f}_k, h\right\rangle f_k \tag{4.1.6}$$

only works when h actually belongs to V in the first place, or when the f_k and g_k sets are actually orthogonal. To illustrate this, assume that f_m is not actually part of the spanning set of J (as in, $m \notin K$). The inner product based projection would, obviously, be zero. However, if we take a single function f_1 out of $\{f_k\}_{k\in K}$ then the norm of the difference is

$$\|af_1 - f_m\|^2 = a^2\,\|f_1\|^2 - 2a\,\langle f_1, f_m\rangle + \|f_m\|^2\,. \tag{4.1.7}$$

The smallest norm value will use $a = \frac{\langle f_1, f_m\rangle}{2\|f_1\|^2}$, which is zero if and only if $\langle f_1, f_m\rangle$. We could have used any function f_k, so the projection of f_m onto $\text{Span}\,\{f_k,\,k \in K\}$ will not be zero unless the f_k are all orthogonal to f_m. Note that this does not imply that the inner product based operation will not provide a good approximation of the projection.

So, our bases are not orthonormal, but we do get unique decompositions via inner products. Also, while this system is more complicated than orthogonal ones, there is the advantage that one of the ϕ and ψ sets could easily be pretty much ideal. For example, we will be only using biorthogonal systems where ϕ is a B-spline. What this means is that it will be easy to confirm that one of the sets spans the space, which actually implies the same for the dual functions.

Property 4.1.1 *Assume that* $\{\psi\}_{k\in\mathbb{Z}}$ *and* $\left\{\widetilde{\psi}\right\}_{k\in\mathbb{Z}}$ *are biorthogonal sets in the Hilbert space* X *and* $\{\psi\}_{k\in\mathbb{Z}}$ *spans* X*. This implies that the* $\widetilde{\psi}$ *set also spans* X *and that*

$$\|f\|^2 = \sum_k \langle f, \psi_k\rangle \left\langle f, \widetilde{\psi}_k\right\rangle. \tag{4.1.8}$$

Proof.

First, we take

$$f = \sum_k a_k\psi_k = \sum_k \left\langle f, \widetilde{\psi}_k\right\rangle \phi_k \in X \tag{4.1.9}$$

and its approximation in the span of the $\widetilde{\psi}$,

$$g = \sum_k b_k\widetilde{\psi}_k = \sum_k \langle f, \psi_k\rangle \widetilde{\psi}_k. \tag{4.1.10}$$

Now to consider the error, $f - g$, along with a fixed ψ element, ψ_l:

$$\begin{aligned}\langle f - g, \psi_l\rangle &= \langle f, \psi_l\rangle - \sum_k \left(\langle f, \psi_k\rangle \left\langle \widetilde{\psi}_k, \psi_l\right\rangle\right) \\ &= \langle f, \psi_l\rangle - \langle f, \psi_l\rangle \left\langle \widetilde{\psi}_l, \psi_l\right\rangle \;=\; 0.\end{aligned} \tag{4.1.11}$$

Since the ψ span X, that error is orthogonal to all X and must be zero.

Proving Equation (4.1.8) is trivial:

$$\|f\|^2 = \langle f, f\rangle = \left\langle \sum_k \left\langle f, \widetilde{\psi}_k\right\rangle \psi_k, \sum_k \langle f, \psi_k\rangle \widetilde{\psi}_k \right\rangle = \sum_k \left\langle f, \widetilde{\psi}_k\right\rangle \langle f, \psi_k\rangle, \tag{4.1.12}$$

so we are done. ■

Now that the preliminaries are out of the way, we can consider what a biorthogonal scaling function and wavelet combination would look like. First, we will need both ϕ and $\widetilde{\phi}$, along with two scaling steps:

$$\phi(x) = \sqrt{2}\sum_k h_k\,\phi(2x-k), \qquad \widetilde{\phi}(x) = \sqrt{2}\sum_k \widetilde{h}_k\,\widetilde{\phi}(2x-k). \tag{4.1.13}$$

The wavelet will be developed much like in the previous chapters, except there will be a flip between the h_k and $\tilde{h}_k$ due to which particular function will be used for the inner products. We get

$$\psi(x) = \sqrt{2}\sum_k g_k\,\phi(2x-k), \qquad \widetilde{\psi}(x) = \sqrt{2}\sum_k \widetilde{g}_k\,\widetilde{\phi}(2x-k) \tag{4.1.14}$$

with

$$g_k = (-1)^k \widetilde{h}_{1-k}, \qquad \widetilde{g}_k = (-1)^k h_{1-k}. \tag{4.1.15}$$

This is, again, exactly what we get for the orthonormal functions.

4.2 Basic Setup for Biorthogonal Scaling Functions

Calculating biorthogonal wavelets is something that texts on the subject typically gloss over, it seems, along with the actual calculation of the scaling functions and wavelets. There are some good reasons for this. First, the results, and the means to prove them, are almost completely identical to those for orthogonal wavelets. Second, the expressions are twice as complex and are frequently very cumbersome to write. We will be taking a short look at the thinking behind biorthogonal wavelets, and have a few examples of how to convert the orthogonal results to the biorthogonal case. It is essential to recall the theorems in Section 2.2 and Chapter 3. The main difference here is that we will not have absolute values of the functions in question, we will instead have products. For instance, instead of $|H(\xi)|^2$ we will have $H(\xi)\overline{\widetilde{H}(\xi)}$. The result is that the proofs will be, for lack of a better expression, messier than before. The steps will be almost the same. As such, we will not bother re-writing many of the proofs for the biorthogonal case, just a select few. Recall that $\widehat{\psi}$ is the Fourier transform of ψ (Section A.1), $\widetilde{\psi}$ is the dual of ψ (see glossary or Section 4.1) and $\overline{\psi}$ is the complex conjugate of ψ. In these proofs, it is not uncommon to see $\overline{\widehat{\widetilde{\psi}}}$, the conjugate of the Fourier transform of the dual of ψ.

First, we deal with the function Φ and biorthogonality.

Theorem 4.2.1 *If* $\Phi(\xi) = \sum_k \widehat{\phi}(\xi + 2\pi k)\overline{\widehat{\widetilde{\phi}}(\xi + 2\pi k)}$ *then*

$$\left\langle \phi(x), \widetilde{\phi}(x-k) \right\rangle = \delta_{0k} \quad \Longleftrightarrow \quad \Phi(\xi) = \frac{1}{2\pi} \quad \textit{almost everywhere.} \tag{4.2.1}$$

This is a more general case of Theorem 3.1.6.

Proof.

This is done the same way as in Theorem 3.1.6: by expanding the inner product

$$\begin{aligned}\left\langle \phi, \widetilde{\phi}_k \right\rangle = \left\langle \widehat{\phi}, \widehat{\widetilde{\phi}}_k \right\rangle &= \int \widehat{\phi}(\xi)\, \overline{\widehat{\widetilde{\phi}}(\xi)}\, e^{-ik\xi}\, d\xi \\ &= \sum_l \int_0^{2\pi} \widehat{\phi}(\xi + 2\pi l)\, \overline{\widehat{\widetilde{\phi}}(\xi + 2\pi l)}\, e^{-ik\xi}\, d\xi \\ &= \int_0^{2\pi} \Phi(\xi) e^{-ik\xi}\, d\xi. \end{aligned} \tag{4.2.2}$$

The last integral in (4.2.2) is akin to a Fourier transform (it is missing a constant term), meaning that the biorthogonal arrangement requires that Φ be constant (in the L^2 sense), with a value of $\frac{1}{2\pi}$. ■

Next, we bring in the function $H(\xi)$, the Fourier space equivalent of the scaling step. The H function is defined in (2.4.1). We will have a dual equivalent of this function, identified with a $\sim$ above, so

$$\widetilde{H}(\xi) = \sum_k \widetilde{h}_k e^{-ik\xi} \quad \Longrightarrow \quad \widehat{\widetilde{\phi}}(\xi) = \widetilde{H}\left(\frac{\xi}{2}\right) \widehat{\widetilde{\phi}}\left(\frac{\xi}{2}\right). \tag{4.2.3}$$

Theorem 4.2.2 *The function $\Phi(\xi)$ equalling $\frac{1}{2\pi}$ almost everywhere requires*

$$H(\xi)\, \overline{\widetilde{H}(\xi)} + H(\xi + \pi)\, \overline{\widetilde{H}(\xi + \pi)} = 1 \quad \textit{almost everywhere.} \tag{4.2.4}$$

This is the biorthogonal version of Theorem 2.3.5.

Proof.

Again, this is much the same as the orthogonal version:

$$\begin{aligned}\Phi(\xi) &= \sum_k \widehat{\phi}(\xi + 2\pi k)\, \overline{\widehat{\widetilde{\phi}}(\xi + 2\pi k)} \\ &= \sum_k \widehat{\phi}(\xi + 2\pi + 4\pi k)\, \overline{\widehat{\widetilde{\phi}}(\xi + 2\pi + 4\pi k)} + \sum_k \widehat{\phi}(\xi + 4\pi k)\, \overline{\widehat{\widetilde{\phi}}(\xi + 4\pi k)}, \end{aligned} \tag{4.2.5}$$

by breaking the sum in two. Next, we include H (and $\widetilde{H}$) for

$$\begin{aligned}\Phi(\xi) = &\sum_k \Big(H(\xi/2 + \pi + 2\pi k)\, \widehat{\phi}(\xi/2 + \pi + 2\pi k) \Big) \overline{\Big(\widetilde{H}(\xi/2 + \pi + 2\pi k)\, \widehat{\widetilde{\phi}}(\xi/2 + \pi + 2\pi k) \Big)} \\ &+ \sum_k \Big(H(\xi/2 + 2\pi k)\, \widehat{\phi}(\xi/2 + 2\pi k) \Big) \overline{\Big(\widetilde{H}(\xi/2 + 2\pi k)\, \widehat{\widetilde{\phi}}(\xi/2 + 2\pi k) \Big)}. \end{aligned} \tag{4.2.6}$$

Next, we recall that H is 2π periodic for

$$\Phi(\xi) = H(\xi/2+\pi)\,\overline{\widetilde{H}(\xi/2+\pi)}\sum_k \widehat{\phi}(\xi/2+\pi+2\pi k)\,\overline{\widehat{\widetilde{\phi}}(\xi/2+\pi+2\pi k)} + H(\xi/2)\,\overline{\widetilde{H}(\xi/2)}\sum_k \widehat{\phi}(\xi/2+2\pi k)\,\overline{\widehat{\widetilde{\phi}}(\xi/2+2\pi k)}. \tag{4.2.7}$$

The next step is to notice $\Phi(\xi/2)$ and $\Phi(\xi/2+\pi)$ in Equation (4.2.5) for

$$\Phi(\xi) = H(\xi/2)\,\overline{\widetilde{H}(\xi/2)}\,\Phi(\xi/2) + H(\xi/2+\pi)\,\overline{\widetilde{H}(\xi/2+\pi)}\,\Phi(\xi/2+\pi). \tag{4.2.8}$$

So, if ϕ and $\tilde{\phi}$ form a biorthogonal system then $\Phi(\xi) = \frac{1}{2\pi}$ almost everywhere, so

$$H(\xi)\,\overline{\widetilde{H}(\xi)} + H(\xi+\pi)\,\overline{\widetilde{H}(\xi+\pi)} = 1, \tag{4.2.9}$$

almost everywhere. ■

Getting the other direction, as before, requires significantly more effort. First, we create the biorthogonal equivalents of the functions f_r, which were defined in Equation (3.1.13). Recall that $\mathbf{1}$ is an indicator function. We get the two sets

$$\begin{aligned} \widehat{f}_0 &= \frac{1}{\sqrt{2\pi}}\,\mathbf{1}_{[-\pi,\pi]}(\xi), \qquad \widehat{f}_r(\xi) = H\left(\frac{\xi}{2}\right)\widehat{f}_{r-1}\left(\frac{\xi}{2}\right), \\ \widehat{\widetilde{f}}_0 &= \frac{1}{\sqrt{2\pi}}\,\mathbf{1}_{[-\pi,\pi]}(\xi), \qquad \widehat{\widetilde{f}}_r(\xi) = \widetilde{H}\left(\frac{\xi}{2}\right)\widehat{\widetilde{f}}_{r-1}\left(\frac{\xi}{2}\right). \end{aligned} \tag{4.2.10}$$

Many of the original results for these will still apply, in a certain sense. Note that the $\left\{\widehat{f}_r\right\}$ and $\left\{\widehat{\widetilde{f}}_r\right\}$ will form biorthogonal sets.

Theorem 4.2.3 *If we have $H(0) = 1$, $H(\pi) = 0$ and $H(\xi) + H(\xi+\pi) = 1$ then*

$$\lim_{r\to\infty}\widehat{f}_r = \prod_{n=0}^{\infty} H\left(\frac{\xi}{2^n}\right) = \widehat{\phi}(\xi) \quad \text{and} \quad \lim_{r\to\infty}\widehat{\widetilde{f}}_r = \prod_{n=0}^{\infty}\widetilde{H}\left(\frac{\xi}{2^n}\right) = \widehat{\widetilde{\phi}}(\xi). \tag{4.2.11}$$

Both convergences are locally uniform and they form a biorthogonal system for all r.

Proof.

The convergences are proven in Theorem 3.1.6, being functionally identical to the original, orthogonal, case. The biorthogonal component requires only the use of the new function Φ. What we need is

$$\sum_k \widehat{f}_r(\xi + 2\pi k)\,\overline{\widehat{\widetilde{f}}_r(\xi + 2\pi k)} = \frac{1}{2\pi}, \quad \text{almost everywhere, for all } r \geqslant 0. \tag{4.2.12}$$

We get that result for $r = 0$, since both sets of functions begin equal to ϕ_{H}. Next, we proceed inductively, exactly as done in Theorem 3.1.6. All that is different is the use of two different functions, which changes very little, as we saw in the last two results. The procedure is much the same, just messier. ■

Corollary 4.2.4 *If*

$$H(\xi)\,\overline{\widetilde{H}(\xi)} + H(\xi + \pi)\,\overline{\widetilde{H}(\xi + \pi)} = 1, \quad \textit{almost everywhere} \tag{4.2.13}$$

and $H(\xi) = 1$, then the related ϕ and $\tilde{\phi}$ form an orthonormal system.

Proof.

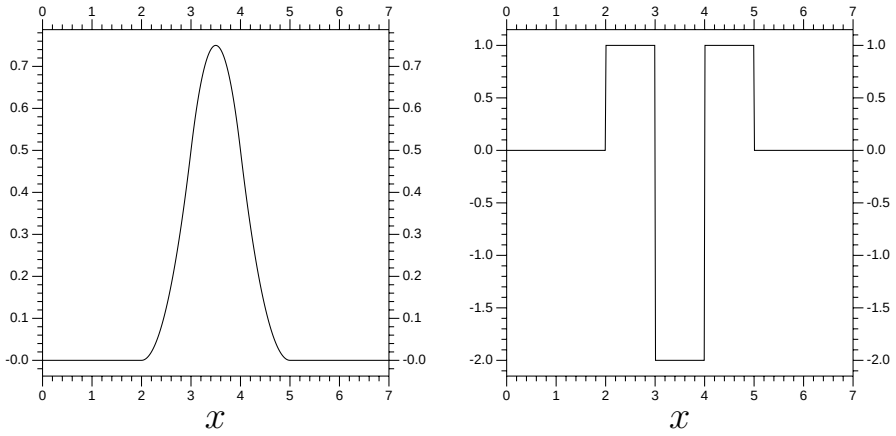
This uses the dominated convergence theorem like with Corollary 3.1.6, with both sets of f_r bounded using the same reasoning. The actual limit then becomes

$$\int \lim_{r\to\infty} \widehat{f}_r(\xi)\,\overline{\widehat{\widetilde{f}}_r(\xi + k)}\,d\xi = \lim_{r\to\infty} \int \widehat{f}_r(\xi)\,\overline{\widehat{\widetilde{f}}_r(\xi + k)}\,d\xi = \lim_{r\to\infty} \delta_{0k} = \delta_{0k}, \tag{4.2.14}$$

just like in the proof of 3.1.6. ■

4.3 Numerical Computation of Biorthogonal Scaling Functions and Wavelets

With the background results dealt with, we can now get to actually calculating ϕ, $\widetilde{\phi}$ and their wavelets. The most typical examples put forward are frequently referred to as dual scaling functions. The scaling function that will be used is created, then a dual function is calculated to go with it. As a result, we can work with pretty



(a) $\phi(x)$ (b) $\frac{\partial^2}{\partial x^2}\phi(x)$

Figure 3: The third order biorthogonal spline scaling function, and its second derivative.

much any function we like, and leave the dual for calculating inner products. Spline functions work well, being both fairly smooth and highly localized. For instance, consider the function in Figure 3, with H function

$$H(\xi) = \left(\frac{1 + e^{-i\xi}}{2}\right)^3. \tag{4.3.1}$$

One thing to note is that the function in question is the second order spline, but a third order scaling function (just like the Haar wavelet is first order in wavelet terms, but a zeroth order spline). This is almost certain to cause confusion, so we will, usually, refer to the order only with respect to wavelets and scaling functions (discussed in Appendix B.1). Anyway, recall that in Section 3.2 we focused on $|H(\xi)^2| = H(\xi)\overline{H(\xi)}$ to verify that a scaling function was an orthonormal one. We can use the same method here, with minimal variation to the proofs. We need to make sure that

$$M(\xi) = H(\xi)\overline{\widetilde{H}(\xi)} \tag{4.3.2}$$

has the required property, namely that $M(\xi) + M(\xi + \pi) = 1$ almost everywhere. This is actually quite easy in this case. We know that an $M(\xi)$ of this order should be of the form

$$M(\xi) = \left(\cos^2\left(\frac{\xi}{2}\right)\right)^N P\left(\sin^2\left(\frac{\xi}{2}\right)\right) \tag{4.3.3}$$

with $P = P_N + R$, with P_N fixed and R some odd polynomial (see Theorem 3.2.1 and related material). We will, as before, dispense with R, so we get

$$M(\xi) = \left|\frac{1+e^{-i\xi}}{2}\right|^6 P_3\left(\frac{1-\cos(\xi)}{2}\right). \tag{4.3.4}$$

So, we can get part of that using

$$H(\xi) = e^{-in\xi}\left(\frac{1+e^{-i\xi}}{2}\right)^3, \tag{4.3.5}$$

meaning that H results in a translation of our original spline function. That n is just a constant used to shift the coefficients h_k, and it has no effect whatsoever on the shape of ϕ. There will be an equal term in $\widetilde{H}$, to keep the $\widetilde{h}_k$ values for $k < 0$ equal to zero.

Factoring the H in (4.3.5) out of M leaves us with

$$\widetilde{H}(-\xi) = e^{in\xi}\left(\frac{1+e^{i\xi}}{2}\right)^3 P_3\left(\frac{1-\cos(\xi)}{2}\right). \tag{4.3.6}$$

That power of $\left(\frac{1+e^{i\xi}}{2}\right)$ in $\widetilde{H}$ can easily be included in $\widetilde{H}(-\xi)$ (which, you recall, needs to be a polynomial of $e^{i\xi}$). This leaves P_3, a polynomial of $\left(\frac{1-\cos(\xi)}{2}\right)$, which can easily be found to be

$$\frac{19}{4} - \frac{9}{2}\cos(\xi) + \frac{3}{4}\cos^2(\xi) = \frac{3}{8}e^{-2i\xi} - \frac{9}{4}e^{-i\xi} + \frac{19}{4} - \frac{9}{4}e^{i\xi} + \frac{3}{8}e^{2i\xi}. \tag{4.3.7}$$

Again, we want that to be a polynomial of $e^{i\xi}$, and the easiest way to get that is to make our final term, $e^{in\xi}$, with $n = 2$ for

$$\begin{aligned}\widetilde{H}(-\xi) &= \left(\frac{3}{8}e^{-2i\xi} - \frac{9}{4}e^{-i\xi} + \frac{19}{4} - \frac{9}{4}e^{i\xi} + \frac{3}{8}e^{2i\xi}\right)e^{2i\xi}\left(\frac{1+e^{i\xi}}{2}\right)^3 \\ &= \left(\frac{3}{8} - \frac{9}{4}e^{i\xi} + \frac{19}{4}e^{2i\xi} - \frac{9}{4}e^{3i\xi} + \frac{3}{8}e^{4i\xi}\right)\left(\frac{1+e^{i\xi}}{2}\right)^3. \end{aligned}\tag{4.3.8}$$

A quick note, though, about the $e^{\pm in\xi}$. This is done to maintain the pattern used with the Daubechies wavelets (Section 3.2, Example 3.2.21 particularly), and not out of necessity. The additional power of $e^{i\xi}$ is used to make the coefficients h_k equal to

zero for $k < 0$, so the meaningful h_k have $k \in [0, 2N-1]$. Again, this is purely out of keeping the pattern from Section 3.2. The eventual result is h and $\widetilde{h}$, like so:

$$\begin{array}{lcccccccc} k: & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ h_k: & 0 & 0 & \frac{1}{8} & \frac{3}{8} & \frac{3}{8} & \frac{1}{8} & 0 & 0 \\ \widetilde{h}_k: & \frac{3}{64} & -\frac{9}{64} & -\frac{7}{64} & \frac{45}{64} & \frac{45}{64} & -\frac{7}{64} & -\frac{9}{64} & \frac{3}{64} \end{array} . \tag{4.3.9}$$

The result of those coefficients h_k is the scaling function from Figure 3, in the location plotted. The dual ($\widetilde{\phi}$, via the $\widetilde{h}_k$) is a (much) less smooth function, as shown by Figure 4. Both ϕ and $\widetilde{\phi}$ are easily calculated using a modification of the earlier ϕ calculating programs. Basic examples can be found in the appendix, specifically in Section C.2. The newly calculated $\widetilde{\phi}$ satisfies the biorthogonality property $\left\langle \phi_{n,k}, \widetilde{\phi}_{n,k'} \right\rangle = \delta_{kk'}$, as shown by Figure 5. So, we have our biorthogonal scaling functions, or, rather, a spline with its dual. Our next step is to calculate the wavelets. We have two separate sets of scaling coefficients, h_k and $\widetilde{h}_k$. As has been stated, we are to construct g_k out of the coefficients $\widetilde{h}_k$ and the $\widetilde{g}_k$ out of the h_k, and we will get two functions ψ and $\widetilde{\psi}$ (Figure 4).

One thing that should now be worked through is getting the correct values into the decompositions. We know that for any j we can write $\phi_{-1,j}$ in terms of functions $\phi_{0,k}$ and $\psi_{0,k}$. What we need is the coefficients $a_{k,j}$ and $b_{k,j}$ such that

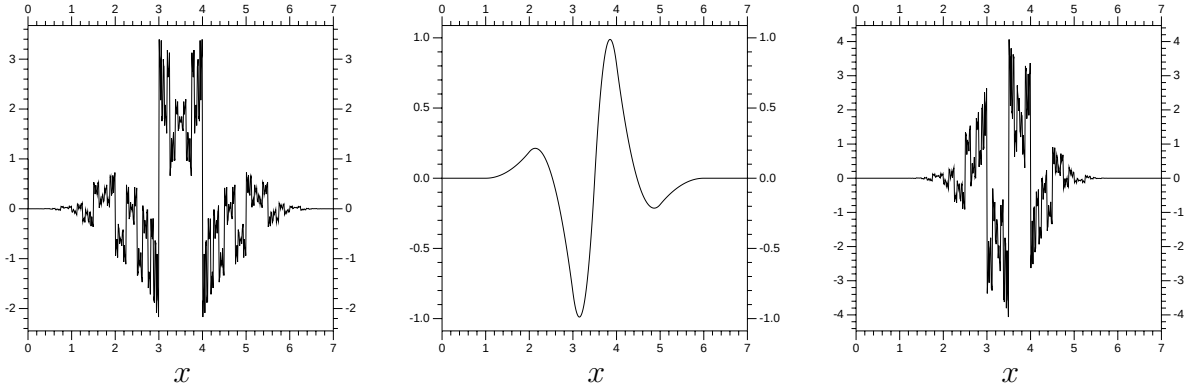
$$\phi_{-1,j}(x) = \sum_k a_{k,j}\phi_{0,k} + \sum_k b_{k,j}\psi_{0,k}. \tag{4.3.10}$$

We need to use the dual functions for this. What we get for the $a_{k,j}$ is

$$\begin{aligned} a_{k,j} = \left\langle \phi_{-1,j}, \widetilde{\phi}_{0,k} \right\rangle &= \left\langle \phi_{-1,j}, \sum_i \widetilde{h}_i \, \widetilde{\phi}_{-1,2k+i} \right\rangle \\ &= \sum_i \widetilde{h}_i \left\langle \phi_{-1,j}, \widetilde{\phi}_{-1,2k+i} \right\rangle = \widetilde{h}_{j-2k}. \end{aligned} \tag{4.3.11}$$

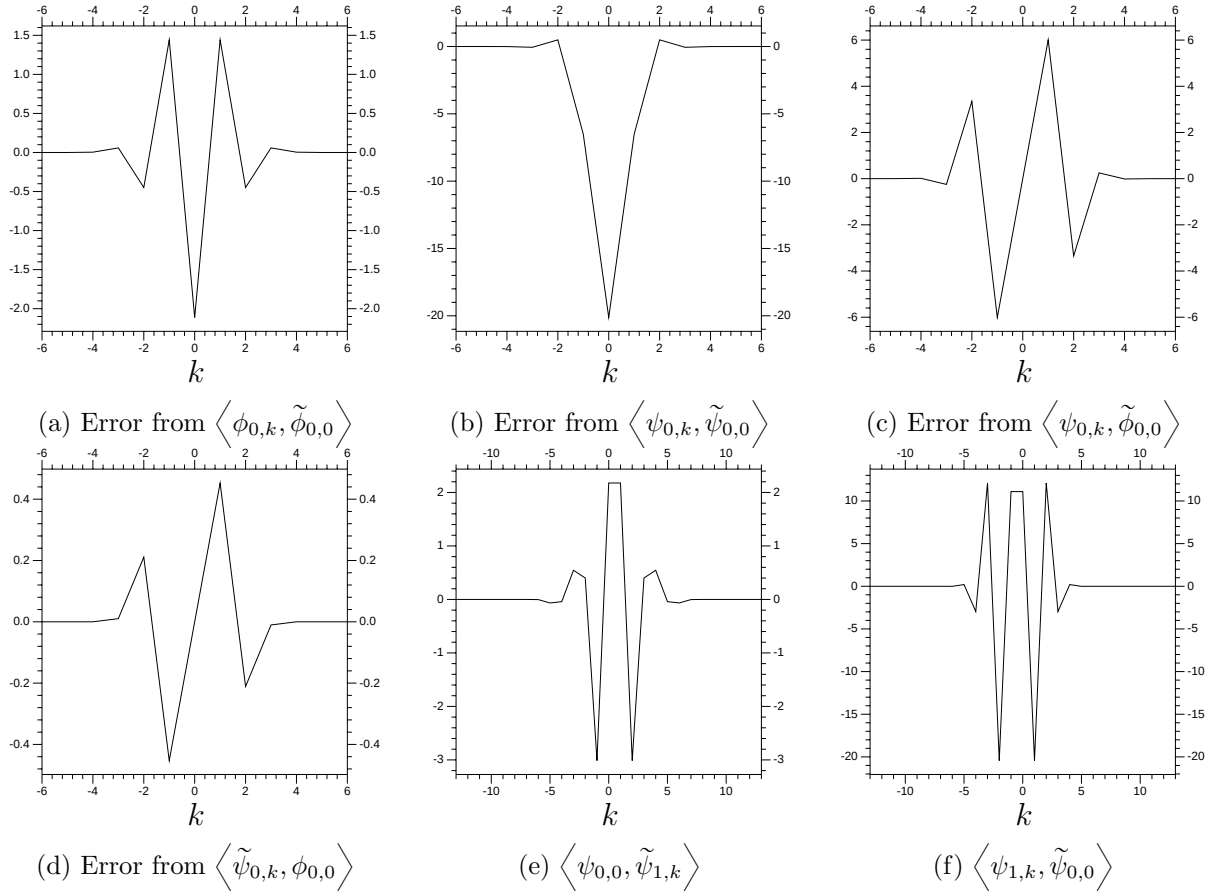
So, the coefficients used with the inverse fast wavelet transform are the duals of the coefficients h_k, namely the $\widetilde{h}_k$. The same will be true with the ψ related coefficients $\widetilde{g}_k$.

Example code can be found, again, in Appendix C.2.



(a) $\widetilde{\phi}_{B_2}(x)$ (b) $\psi_{B_2}(x)$ (c) $\widetilde{\psi}_{B_2}(x)$

Figure 4: The second order biorthogonal spline wavelet, and related duals. All at, or calculated at a resolution of $2^7 = 128$ elements per unit length in the x direction.



(a) Error from $\left\langle \phi_{0,k}, \widetilde{\phi}_{0,0} \right\rangle$ (b) Error from $\left\langle \psi_{0,k}, \widetilde{\psi}_{0,0} \right\rangle$ (c) Error from $\left\langle \psi_{0,k}, \widetilde{\phi}_{0,0} \right\rangle$

(d) Error from $\left\langle \widetilde{\psi}_{0,k}, \phi_{0,0} \right\rangle$ (e) $\left\langle \psi_{0,0}, \widetilde{\psi}_{1,k} \right\rangle$ (f) $\left\langle \psi_{1,k}, \widetilde{\psi}_{0,0} \right\rangle$

Figure 5: Errors from the inner products of ϕ_{B_2}, ψ_{B_2}, and their duals, in units of 10^{-6}. The vertical is the error calculated, in units of 10^{-6}, so our largest value is approximately 2×10^{-5}. The horizontal is the value $k \in \mathbb{Z}$ from the inner product involved, which need not get very large since all of the functions have bounded support. The first two, a) and b), are supposed to be equal to δ_{0k}. The others are supposed to be equal to zero. The first four are all at the base, $n = 0$, resolution, so the k relate to integer translates. Because they are functions at finer resolution, the last two graphs use translates of $\frac{1}{2}$ units.

Chapter 5

Numerical Application

5.1 Getting Started

This chapter will look at several practical considerations for the actual application of wavelets to solving partial differential equations. Some of the sections will include comparisons between the application of wavelets and the application of Fourier series to differential equations. This provides a stark contrast with the relevant properties of wavelets. Those unfamiliar with Fourier series can look to Section A.2 for a brief introduction.

Consider a decomposition based on the wavelet ψ, the projection of the function f onto the space V_J. First off, we can use series of the form

$$P_J f = \sum_{j>J} \sum_{k\in\mathbb{Z}} b_{j,k}\, \psi_{j,k}, \tag{5.1.1}$$

in theory, but in practice that would be either impractical or simply impossible. Even for a compact domain, we would need an infinite number of W_j spaces, and so an infinite number of coefficients $b_{j,k}$, to create any space V_J. We generally have a starting point in one of the V_j spaces, so a sum of $a_{j,k}\phi_{j,k}$ elements. Generally, the V_j space chosen will have insufficient resolution, so we add extra resolution in the form of spaces W_j (recall that $V_J \subset V_{J-1} = V_J \oplus W_J$). Generally, we use V_0 as the starting

point, so we have functions of the form

$$P_J f(x) = \sum_{k\in\mathbb{Z}} a_k\,\phi(x-k) + \sum_{j=0}^{J-1}\sum_{k\in\mathbb{Z}} b_{-j,k}\,\psi_{-j,k}(x) \quad \in V_{-J}. \tag{5.1.2}$$

Some books use V_0 as the highest resolution and have the wavelets expanding to V_0 from some practical base V_J (with $J \geqslant 0$). The others use V_0 as the base. Both work perfectly well, being theoretically identical. We will use the second option, so all of our setups will use V_0 as the base resolution and some larger space V_J, with $J < 0$, as the full space. We will keep this format, and use it in the numerical examples in future chapters. This means that our lowest resolution will always have a function ϕ at every integer, with functions $\psi_{j,k}$ included for greater accuracy.

5.2 Localization

The main issue with the Fourier decomposition is that the functions $e^{ik\xi}$ have support over the entire space. The uncertainty principle (Section 2.5) means that any ϕ/ψ pairing will be either localized in space or have high derivatives. As such, we can view the Fourier series as being one extreme: infinitely differentiable but with no localization. The other extreme would be the Haar scaling function: perfectly localized but discontinuous. Working with wavelets, which have both resolution and location, means that we can easily put extra resolution on a decomposition of a function if and when it needs it. This has a similar effect as, for instance, using smaller finite differences in a particularly important section of a domain. However, we are working with an orthonormal basis, so all that is needed is to start including coefficients $b_{j,k}$ (and associated functions $\psi_{j,k}$) in the calculations. Since that is all we need to do, it much easier to adapt a wavelet decomposition.

Wavelet decompositions also allow us to take advantage of localized high resolution to compress data. Take a function f, analyzed up to the resolution $J < 0$ (so V_J) in the domain $[0, N]$, $N \in \mathbb{Z}$ and > 0. It will be of the form

$$f(x) = \sum_{k=0}^{2^{-J}N-1} a_{J,k}\,\phi_{J,k}(x) = \sum_{k=0}^{2^{-(J+1)}N-1} a_{J+1,k}\,\phi_{J+1,k}(x) + \sum_{k=0}^{2^{-(J+1)}N-1} b_{J+1,k}\,\psi_{J+1,k}(x), \tag{5.2.1}$$

Table 2: A summary of the effect of cutting coefficients $b_{j,k}$ with magnitude $< T$. These results relate to Figure 6, and the function $f(x)$ plotted there.

T	Number of $b_{j,k}$ kept	L^2 Error	L^1 Error	L^∞ Error
	639	$7.484 \cdot 10^{-4}$	$3.573 \cdot 10^{-4}$	$5.368 \cdot 10^{-3}$
10^{-7}	611	$7.484 \cdot 10^{-4}$	$3.575 \cdot 10^{-4}$	$5.368 \cdot 10^{-3}$
10^{-6}	490	$7.484 \cdot 10^{-4}$	$3.631 \cdot 10^{-4}$	$5.368 \cdot 10^{-3}$
10^{-5}	365	$7.494 \cdot 10^{-4}$	$4.224 \cdot 10^{-4}$	$5.368 \cdot 10^{-3}$
10^{-4}	235	$8.146 \cdot 10^{-4}$	$9.443 \cdot 10^{-4}$	$5.368 \cdot 10^{-3}$
10^{-3}	148	$2.668 \cdot 10^{-3}$	$5.550 \cdot 10^{-3}$	$5.387 \cdot 10^{-3}$
10^{-2}	90	$3.099 \cdot 10^{-2}$	$6.702 \cdot 10^{-2}$	$7.503 \cdot 10^{-2}$

since $V_J = V_{J+1} \oplus W_{J+1}$. If, however, we only needed the J resolution over, say, a single value (at the location v, so the coefficient $b_{J+1,v}$) then we could express $f(x)$ accurately as

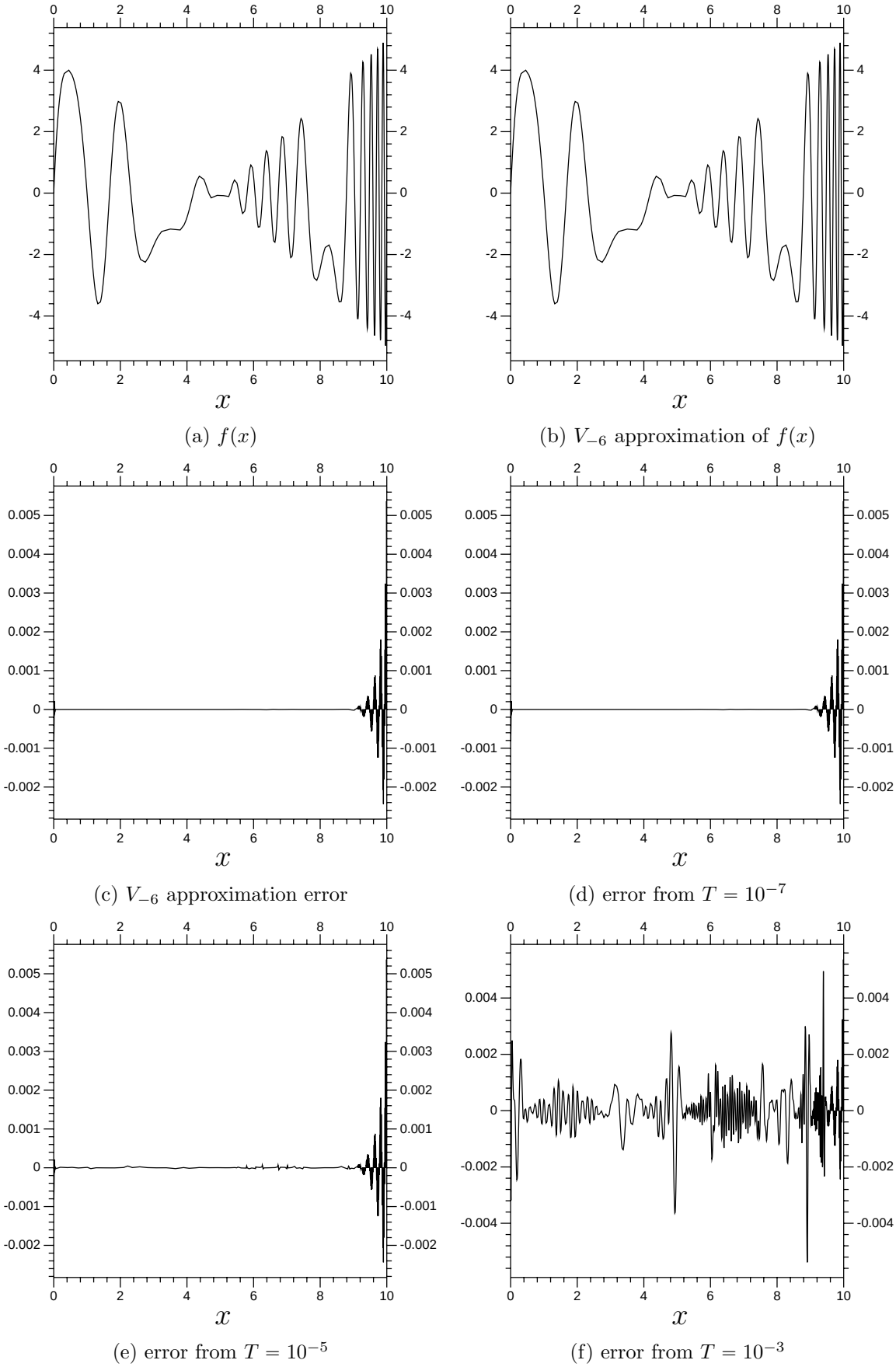
$$f(x) = \sum_{k=0}^{2^{-(J+1)}N-1} a_{J+1,k}\,\phi_{J+1,k}(x) + b_{J+1,v}\psi_{J+1,v}(x). \tag{5.2.2}$$

Our sum containing $2^{-J}N$ terms now has $2^{-(J+1)}N + 1$ terms (recall that $J < 0$). This is primarily of help in compressing images, which are likely to contain important details as well as fairly uniform sections.

Example 5.2.1 *A wavelet decomposition of the function*

$$f(x) = (x-5)\,\sin\left(\frac{x(x-1)(x-5)(x-10)}{10}\right) \quad \textit{on} \quad x \in [0, 10].$$

The plot of f is in Figure 6, part a). Notice the mix of smooth and irregular sections. The fine resolution wavelets $\psi_{j,k}$ will be unnecessary where the function is smooth. The later plots in Figure 6 are the errors when the coefficients $b_{j,k}$ with magnitude below a given $T > 0$ are removed (set to zero). Notice the error becoming larger (and coarser) as T increases from 10^{-7} to 10^{-3}. Table 2 has a summary of the results, using multiple norms. Notice that removing the 149 smallest coefficients $b_{j,k}$ results in a very small increase in the error.



(a) $f(x)$

(b) V_{-6} approximation of $f(x)$

(c) V_{-6} approximation error

(d) error from $T = 10^{-7}$

(e) error from $T = 10^{-5}$

(f) error from $T = 10^{-3}$

Figure 6: Wavelet decompositions of $f(x)$, and additional error from cutting all $b_{j,k}$ values below a tolerance T. All are plotted at 2^8 resolution. The wavelet is the fourth order biorthogonal spline function. Any coefficient $b_{j,k}$ below T is cut (set to zero).

5.3 Derivatives

We intend to approximate partial differential equations, so we will definitely need derivatives. This section is entirely about approximating derivatives using wavelets, first with projections then using a collocation method. In both cases, for an m^{th} order derivative, we will want an arrangement with a matrix M mapping a V_J approximation of a function $f(x)$ onto a V_J approximation of $\frac{\partial^m}{\partial x^m} f(x)$. If we have a vector $\mathbf{a}$ containing the coefficients $a_{J,k}$ from V_J approximation of f, then a proper setup will result in our approximation of the derivative being simply $M\mathbf{a}$.

We will be using finer resolutions eventually, but for now let us restrict ourselves to V_0. Consider the domain $[0, N]$ and the projection of a function f onto V_0,

$$P_0 f(x) = \sum_{k=0}^{N} \langle f, \phi_{k,0} \rangle \, \phi_{k,0} = \sum_{k=0}^{N} a_k \phi_{k,0}(x). \tag{5.3.1}$$

What we want is the projection of the derivative onto V_0, which will be

$$\begin{aligned} P_0 \left(\frac{d}{dx} P_0 f(x) \right) = \sum_{k}^{N} a'_k \phi_{0,k} &= \sum_{k=0}^{N} \left\langle \frac{d}{dx} P_0 f(x), \phi_{0,k}(x) \right\rangle \phi_{0,k}(x) \\ &= \sum_{m=0}^{N} \sum_{k=0}^{N} a_k \left\langle \frac{d}{dx} \phi_{0,k}(x), \phi_{0,m}(x) \right\rangle \phi_{0,m}(x) \\ &= \sum_{m=0}^{N} \sum_{k=0}^{N} a_k \left\langle \frac{d}{dx} \phi(x), \phi_{0,m-k}(x) \right\rangle \phi_{0,m}(x). \end{aligned} \tag{5.3.2}$$

If we have the coefficients $a_{0,k}$ arranged into a vector $\mathbf{a}_0$ and the coefficients $a'_{0,k}$ into a vector $\mathbf{a}'_0$, then we want to arrange the $\left\langle \frac{d}{dx} \phi(x), \phi_{0,m-k}(x) \right\rangle$ terms into a matrix D_0 such that $\mathbf{a}'_0 = D_0 \mathbf{a}_0$. Looking at the sum in (5.3.2) for an individual m shows us that the element

$$(D_0)_{m,n} = \left\langle \frac{d}{dx} \phi(x), \phi_{0,n-m}(x) \right\rangle. \tag{5.3.3}$$

How those inner product values are calculated is not important, as long as the method is accurate. The most basic is to calculate ϕ numerically, at very fine resolution, use a finite difference scheme to calculate ϕ', then calculate $\langle \phi', \phi(\cdot - k) \rangle$ numerically. This is likely to be time consuming, but it need only be done once.

Starting with a finer resolution, doing everything in V_J ($J < 0$) gives greater accuracy, of course. Using a matrix D_J, structured similarly to D_0 except with elements $\left\langle \phi'_{J,k}, \phi_{J,m} \right\rangle$, we can approximate the derivative in V_J. If we have f related coefficients $a_{J,k}$ arranged into the vector $\mathbf{a}_J$, then the f' related coefficients $a'_{J,k}$ arranged in $\mathbf{a}'_J$ can be calculated via $\mathbf{a}'_J = D_J \mathbf{a}_J$. We can get this approximation with respect to the standard $V_0 \oplus W_0 \oplus \ldots \oplus W_{J+1} = V_J$ form by applying the fast wavelet transform (and its inverse). We also can re-use the inner product calculations from the V_0 resolution, since

$$\begin{aligned}
\left\langle \frac{\partial^m}{\partial x^m} \phi_{-j,0}(x), \phi_{-j,k}(x) \right\rangle &= \left\langle \frac{\partial^m}{\partial x^m} \left(2^{j/2} \phi\left(2^j x\right)\right), 2^{j/2} \phi(2^j x - k) \right\rangle \\
&= 2^j \cdot 2^{jm} \left\langle \phi^{(m)}(2^j x), \phi(2^j x - k) \right\rangle \\
&= 2^j \cdot 2^{jm} \left(\frac{1}{2^j} \left\langle \phi^{(m)}(x), \phi(x - k) \right\rangle \right) \\
&= 2^{jm} \left\langle \phi^{(m)}(x), \phi_{0,k}(x) \right\rangle . \qquad (5.3.4)
\end{aligned}$$

This arrangement requires working in the largest V_J space (lowest value J), which is a plausible option when it is easy to convert from the more usual $V_0 \oplus W_0 \oplus \cdots \oplus W_{J+1}$ arrangement to V_J. It generally is, since the fast wavelet transform (FWT) and the inverse fast wavelet transform (IFWT) are both linear. There is, however, another option, one that is somewhat difficult to explain, but some later examples make use of it. This type of breakdown can be found in [9, p. 137] (in a paper by G. Beylkin and J. Keiser). While the system above uses only a set of values of the form $\left\langle \frac{d}{dx} \phi_{J,k}, \phi_{J,l} \right\rangle$, and at the highest resolution, this alternate strategy uses ψ related coefficients directly. This involves including matrices C_J, B_J and A_J, each including inner product values of the form

$$\left\langle \frac{d}{dx} \phi_{J,k}, \psi_{J,l} \right\rangle, \qquad \left\langle \frac{d}{dx} \psi_{J,k}, \phi_{J,l} \right\rangle, \text{ and } \left\langle \frac{d}{dx} \psi_{J,k}, \psi_{J,l} \right\rangle, \qquad (5.3.5)$$

respectively. Using Equation (5.3.4), we need only calculate these at the $J = 0$ resolution, and re-use them for every other resolution. Recall that the D_j matrices are used to calculate the projection onto V_j of the derivative of a function in V_j. The A_j, B_j and C_j are similar, relating to different spaces. The matrices A_j calculate the

projection onto W_j of the derivative of a function in W_j. The matrices B_j calculate the projection onto W_j of the derivative of a function in V_j. Lastly, C_j is for calculating the projection onto V_j of the derivative of a function in W_j. If we are just using V_0, then we only use $D_0[\mathbf{a}_0]$ as our ϕ coefficients approximating the derivative, (with $\mathbf{a}_0$ an ordered vector of $\phi_{0,k}$ related coefficients $a_{0,k}$). If we have $V_0 \oplus W_0 = V_{-1}$, then we need all four for a derivative approximation of the form

$$\begin{bmatrix} A_0 & B_0 \\ C_0 & D_0 \end{bmatrix} \begin{bmatrix} \mathbf{b}_0 \\ \mathbf{a}_0 \end{bmatrix}. \tag{5.3.6}$$

Things get a little more complicated with a larger setup. If we have V_{-2} then we use

$$\begin{bmatrix} A_{-1} & B_{-1} & O & O \\ C_{-1} & O & O & O \\ O & O & A_0 & B_0 \\ O & O & C_0 & D_0 \end{bmatrix} \begin{bmatrix} \mathbf{b}_{-1} \\ \mathbf{a}_{-1} \\ \mathbf{b}_0 \\ \mathbf{a}_0 \end{bmatrix}, \tag{5.3.7}$$

with O representing zero matrices. The reason there is no matrix D_{-1}, why that portion of the matrix in (5.3.7) is zero, is because the lower four matrices actually handle that part. That multiplication is, in fact, equivalent to $D_{-2}[\mathbf{a}_{-2}]$, it just can start pretty much from the $V_0 \oplus W_0 \oplus W_{-1}$ and not V_{-2}. This also ties in nicely with the fact that one needs only a single base resolution V space, everything else is done with the wavelets themselves in the W spaces.

This setup, especially the limited use of the D type matrices, may provide some insight into how the multi-leveled decomposition interacts with derivatives. While some of the multi-leveled thinking will be helpful in the future, it is better for us to calculate derivatives using the D_J matrix along with forward and inverse fast wavelet transforms, FWT and IFWT (see Section B.2), via matrix equivalents M_{FWT} and M_{IFWT}. Our first matrix is M_{IFWT}, which converts $[\mathbf{b}_{J+1}, \mathbf{b}_{J+2}, \ldots, \mathbf{b}_0, \mathbf{a}_0]^T$ into an equivalent $[\mathbf{a}_J]$ vector, then we have M_{FWT} which does in the opposite. When working with orthogonal wavelets and a translation invariant operator, this is all that is needed to get the orthogonal projection of the operator. The resulting matrix multiplication will simply be

$$M_{FWT} \times D_J \times M_{IFWT}, \tag{5.3.8}$$

which takes the coefficients, transforms them into the pure V_J space coefficients, converts them to the derivative values, then back to the coefficients of the usual $V_0 \oplus W_0 \oplus \cdots \oplus W_{J+1}$ decomposition. The matrix D_J is the same as before.

Unfortunately, the projection based setup is not practical for biorthogonal wavelets. Recall that in Section 4.1 we demonstrated that the inner product based calculations for projections does not work with biorthogonal decompositions. It may be helpful to switch to what is called a collocation method. The derivative component involves calculating the exact derivative of the function at an appropriate density of points, then solving for the coefficients that satisfy the calculated values at those points. Basically, calculation then interpolation. Assume a matrix based setup. We will need our old M_{FWT} and M_{IFWT} matrices to handle the FWT and IFWT. Next, we have P_J, a matrix that converts vectors of coefficients $a_{J,k}$ from

$$f(x) = \sum_k a_{J,k}\phi_{J,k}(x)$$

to vectors of physical values of $f(x)$, so $P_J\mathbf{a}_J$ will have components

$$f\left(2^J b\right) = \sum_k a_{J,k}\,\phi_{J,k}\left(2^J b\right)$$

assuming that we decide to use the 2^J locations. So, what P_J does is convert the scaling function coefficients from a function f into the actual values of f. Finally, we have P_J^m, which does the same except it uses values $\frac{d^m}{dx^m}\phi_{J,k}$, so it calculates exact values of $\frac{\partial^m}{\partial x^m}\left(P_J f\right)$. The vector $P_J^m\mathbf{a}_J$ will have components like

$$\sum_k a_{J,k}\,\phi_{J,k}^{(m)}\left(2^J b\right).$$

Using these matrices, the matrix for calculating the derivative would be

$$M_{Der} = M_{FWT} \times \left(P_J\right)^{-1} \times P_J^m \times M_{IFWT}, \tag{5.3.9}$$

assuming we were working with the usual $V_0 + W_0 + W_{-1} + \cdots + W_{J+1}$ setup.

5.4 Application to Higher Dimensions

We will now have a short discussion about extending the wavelet decomposition into more than one dimension. This is not as big a problem as it may look. We have a

nested sequence of subspaces

$$\cdots \subset V_J \subset \cdots \subset V_1 \subset V_0 \subset V_{-1} \subset \cdots \subset V_{-J} \subset \cdots \tag{5.4.1}$$

where $\overline{\bigcup V_j} = L^2(\mathbb{R})$ and so forth. As a result, it is easy to see that $\overline{\bigcup_j (V_j \times V_k)}$ will span $L^2(\mathbb{R}^2)$. Also, $V_j \times V_j$ will clearly be spanned by translations of $\phi_j(x)\phi_j(y)$. Defining the wavelets and the two dimensional equivalents of W_j spaces is a little more difficult. The first problem is one of notation. From now on, we will use $V_j^2 = V_j \times V_j$, which is simply the span of the functions $\phi_{j,k}(x)\phi_{j,l}(y)$. Also, $V_j \times W_j$ will refer to the space spanned by translates of $\phi_j(x)\psi_j(y)$, as one would expect. The two dimensional equivalent of the W spaces are

$$W_j^2 = (V_{j-1} \times V_{j-1}) \bigcap (V_j \times V_j)^\perp = V_{j-1}^2 \bigcap \left(V_j^2\right)^\perp, \tag{5.4.2}$$

which turns out to be distinct from $W_j \times W_j$, merely the span of $\psi_{j,k}(x)\psi_{j,l}(y)$ functions. What we get is

$$W_j^2 = (W_j \times W_j) \oplus (W_j \times V_j) \oplus (V_j \times W_j). \tag{5.4.3}$$

Here is the result:

$$\begin{aligned} V_j^2 \oplus W_j^2 &= (V_j \times V_j) \oplus (W_j \times W_j) \oplus (W_j \times V_j) \oplus (V_j \times W_j) \\ &= [V_j \times (V_j \oplus W_j)] \oplus [W_j \times (W_j \oplus V_j)] \\ &= (V_j \times V_{j-1}) \oplus (W_j \times V_{j-1}) = (V_j \oplus W_j) \times V_{j-1} \ = V_{j-1} \times V_{j-1} \ = V_{j-1}^2. \end{aligned} \tag{5.4.4}$$

So, which terms will, when translated, form a basis for that space? All of the individual components have simple answers. From the three component spaces we get basis functions

$$\psi_{j,k_1}(x)\psi_{j,k_2}(y), \qquad \psi_{j,k_1}(x)\phi_{j,k_2}(y), \quad \text{and} \quad \phi_{j,k_1}(x)\psi_{j,k_2}(y). \tag{5.4.5}$$

It is fairly easy to see that these elements are orthogonal to each other, with or without translation over $\mathbb{Z}$, and are still normalized.

If we need more dimensions we can just continue the process. This is sometimes referred to as the non-standard form. The standard form is less unique to wavelets.

It involves simply including all products between functions from the wavelet decomposition in the x direction with all functions from the wavelet decomposition in the y direction. As a result, we get basis vectors of the form $\psi_{k_1,j_1}(x)\psi_{k_2,j_2}(y)$. Note that j_1 and j_2 can be very different values, resulting in a mix between two very different resolutions.

Example 5.4.1 *The Haar basis expanded to two dimensions.*

The Haar based space V_{-2} for $[0,1] \times [0,1]$ can be written several ways, we will focus on three.

1. The set of functions ϕ_{-2}, indicator functions with disjoint supports, arranged like so:

$\phi_{0,3}$	$\phi_{1,3}$	$\phi_{2,3}$	$\phi_{3,3}$
$\phi_{0,2}$	$\phi_{1,2}$	$\phi_{2,2}$	$\phi_{3,2}$
$\phi_{0,1}$	$\phi_{1,1}$	$\phi_{2,1}$	$\phi_{3,1}$
$\phi_{0,0}$	$\phi_{1,0}$	$\phi_{2,0}$	$\phi_{3,0}$

.

2. The non-standard form, which starts with the $\phi_0(x,y) = 1$ for $x \in [0,1]$ and $y \in [0,1]$. It then uses sets of functions of the form

+	+	−	−
+	+	−	−
+	+	−	−
+	+	−	−

−	−	−	−
−	−	−	−
+	+	+	+
+	+	+	+

−	−	+	+
−	−	+	+
+	+	−	−
+	+	−	−

in addition to ϕ_0 to get to V_{-1}. The '+' represents where the function is positive and the '−' where it is negative, the actual values change to maintain orthonormality. To get to V_{-2}, the non-standard form uses more functions of the same form on the supports of the functions ϕ_{-1},

0	0	0	0
0	0	0	0
+	−	0	0
+	−	0	0

0	0	0	0
0	0	0	0
−	−	0	0
+	+	0	0

0	0	0	0
0	0	0	0
−	+	0	0
+	−	0	0

which are $\psi_{-1,0}(x)\phi_{-1,0}(y)$, $\phi_{-1,0}(x)\psi_{-1,0}(y)$ and $\psi_{-1,0}(x)\psi_{-1,0}(y)$ respectively.

The result is a total of $1+3\times 1+3\times 4=16$ different functions, but notice how they increase in localization.

3. The standard form, the tensor product form, uses, as before, $\phi_0(x,y)$. Next, it uses all $\phi_0(x)\times\psi_j(y)$, $\psi_i(x)\times\phi_0(y)$ and $\psi_i(x)\times\psi_j(y)$ for $i,j=0,-1$. This means we get, as before, the functions that were used earlier to get to V_{-1}. However, the next layer has functions like

0	0	0	0
0	0	0	0
–	–	–	–
+	+	+	+

$= \psi_{-1,0}(y)\times\phi_{0,0}(x).$

Notice that the localization in $\psi_{-1,0}(y)\phi_{0,0}(x)$ is only in one direction. This is why the tensor product is typically ignored in favor of the non-standard form, since the main purpose of wavelets is the localization. We will always use non-standard form when wavelets are used in multiple directions. This will not be common. Our major multi-dimensional example will be best served by a Fourier decomposition in one of the directions. This is even easier. In one direction, choose y, there will be a wavelet decomposition

$$\sum_k a_k\phi_{0,k}(y)+\sum_j\sum_k b_{j,k}\psi_{-j,k}(y). \tag{5.4.6}$$

In the other direction, choose x, we will have a Fourier decomposition, so a decomposition of the form $\sum_l c_l\, e^{ilx}$. Combining the two together results in the decomposition

$$f(x,y)=\sum_l e^{ilx}\left(\sum_k a_{l,k}\phi_{0,k}(y)+\sum_j\sum_k b_{l,j,k}\psi_{-j,k}(y)\right). \tag{5.4.7}$$

Basically, we end up with a full V_J wavelet decomposition in the y direction for each Fourier term used. The biggest change is that, depending upon how it is written, the new coefficients $a_{l,k}$ and $b_{l,j,k}$ could be complex.

Chapter 6

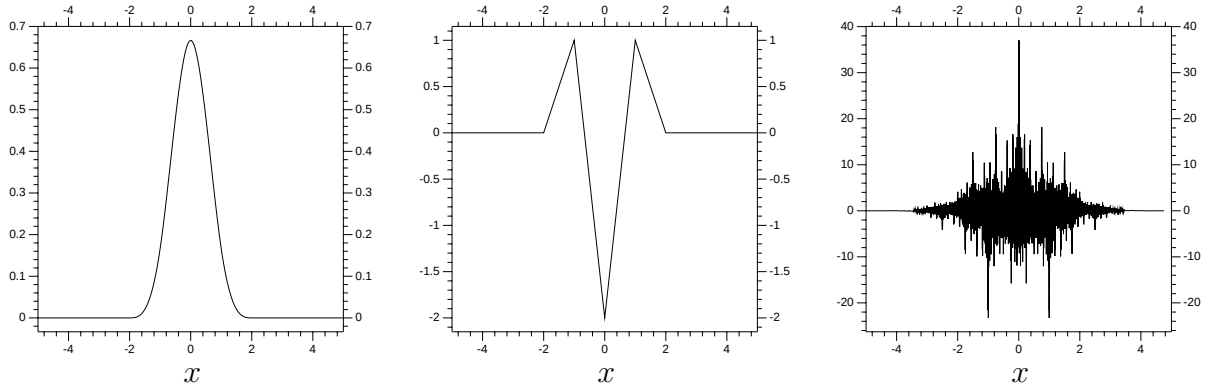
Numerical Examples

At this point we are ready to take a look at numerical solutions for partial differential equations. We will spend this chapter introducing components that will be necessary for numerically solving our main problem, one derived from Rossby waves (in Chapter 8). First, an introduction of our biorthogonal scaling function and wavelet followed by an introduction to a procedure for implementing zero boundary conditions. At this point a time discretization (time stepping scheme) will be chosen and applied to a simple, one dimensional, heat equation problem. The next step will be finding out how to apply arbitrary linear operators and non-zero boundary conditions, both will have their own example problems. Finally, we will move on to Burgers' equation, whose non-linear terms will provide a new challenge.

There are many books on wavelets themselves, or wavelets for signal processing. Sources specifically about application to differential equations are rarer. Here are two: [26] and [9].

6.1 Our Biorthogonal Decomposition

We will now work out a biorthogonal scaling function and wavelet pair to use for our calculations. Recall from Chapter 3 that any ϕ and ψ can be characterized entirely from the H function (Equation 3.1.8). Also, see Chapter 4 about biorthgonal wavelets in general.



(a) $\phi(x)$ (b) $\frac{\partial^2}{\partial x^2}\phi(x)$ (c) $\widetilde{\phi}(x)$

Figure 7: The spline function B_3 and its second derivative and dual, re-centered at $x = 0$. The spline function will be used as a scaling function, hence its labeling as ϕ. Notice it has a continuous second derivative, but only a weak third derivative.

The function H (see Definition 2.4.1) is of the form discussed in Section 4.3:

$$H(\xi) = \left(\frac{1 + e^{-i\xi}}{2}\right)^4 e^{-3i\xi}. \tag{6.1.1}$$

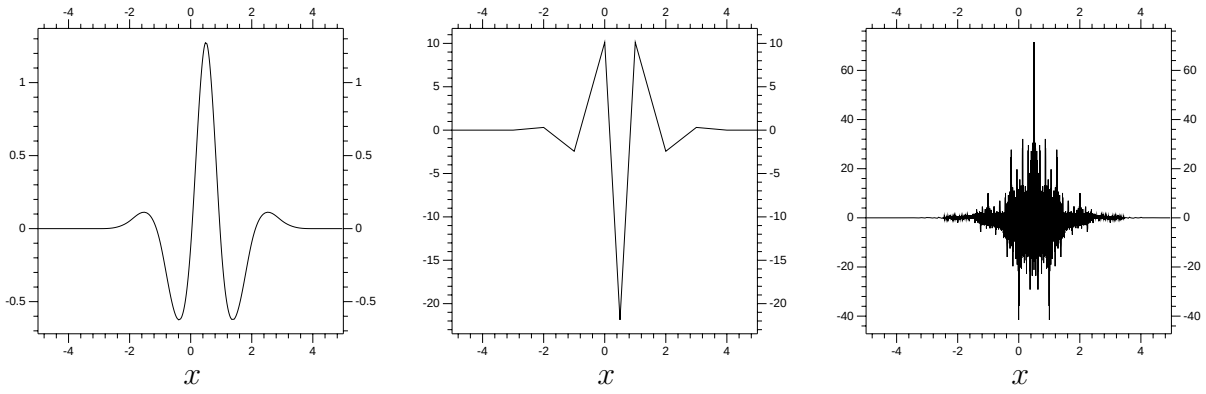
The dual is

$$\begin{aligned}\widetilde{H}(\xi) = \left(\frac{1}{256}\right)\big(&-5 + 20e^{-i\xi} - e^{-2i\xi} - 96e^{-3i\xi} + 70e^{-4i\xi} + 280e^{-5i\xi} \\ &+70e^{-6i\xi} - 96e^{-7i\xi} - e^{-8i\xi} + 20e^{-9i\xi} - 5e^{-10i\xi}\big).\end{aligned} \tag{6.1.2}$$

Results are in Figure 7. The $\widetilde{\phi}$ is, as expected, not a very nice function at all.

Next, using both sets of h_k, we can get the actual wavelets. As before, the extra coefficients used on ϕ do not have an adverse effect on its smoothness. Results are in Figure 8. The inner products work out to approximately what they should be. We get the error $\left\{\left\langle \phi_k, \widetilde{\phi}_0 \right\rangle - \delta_{0,k}, k \in \mathbb{Z}\right\}$ in the 10^{-5} range, with similar accuracy for $\left\langle \phi_k, \widetilde{\psi}_0 \right\rangle$ and $\left\langle \widetilde{\phi}_k, \psi_0 \right\rangle$.

Code related to applying the FWT and IFWT to the ϕ and ψ coefficients can be found in Section C.3. Note that this code does include the boundary condition arrangement discussed in the next section.



(a) $\psi(x)$ (b) $\frac{\partial^2}{\partial x^2}\psi(x)$ (c) $\widetilde{\psi}(x)$

Figure 8: The wavelet pair corresponding to the functions in Figure 7.

6.2 Boundary Conditions

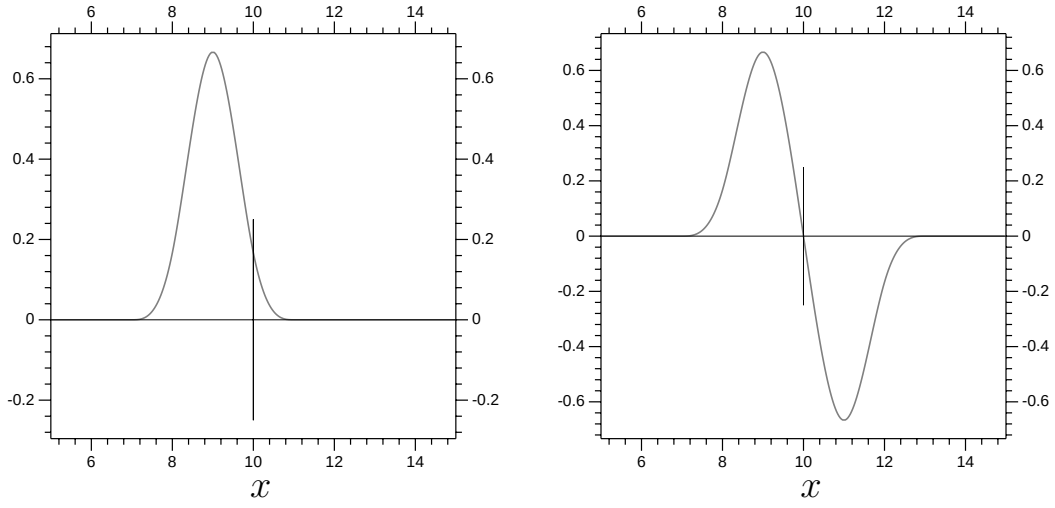
Our first problem will be

$$u_t = u_{xx}, \quad u(x,0) = f(x), \quad \text{on } \Omega = (0,10), \qquad u(0,t) = u(10,t) = 0. \tag{6.2.1}$$

We need to keep our approximation of u consistent with the boundary conditions. This will involve putting some restrictions on the coefficients a_k and b_k.

The problem is fairly simple to state. The function $\phi_{0,9}(x)$, the V_0 scaling function centered at $k = 9$, has a non-zero value at $x = 10$. Including this function, without modifying or correcting for it in some way, will contradict the boundary condition. We cannot just get rid of it (i.e., set $a_{0,9} = 0$), since that would make V_0 incomplete, making it impossible to use the FWT and IFWT around $x = 9$. All of our derivative calculation methods rely upon the wavelet transforms (both the FWT and IFWT, see Section B.2). In fact, using the IFWT on, say, $\phi_{-1,18}$, centered at $x = 9$, will re-write $\phi_{-1,18}$ as a combination of functions $\phi_{0,k}$ and $\psi_{0,k}$, some of which will be centered outside the domain $(0, 10)$. Setting to zero any coefficients relating to the functions ϕ and ψ that are not equal to zero at the boundary is not an option. One way to deal with this is to compensate using functions outside the boundary, as illustrated by Figure 9.

The method we will use involves keeping track of functions $\phi_{j,k}$ and $\psi_{j,k}$ with support outside Ω. Keep in mind that the ϕ and ψ we are using are symmetric around $x = 0$ and $x = \frac{1}{2}$, respectively (Figures 7 and 8). Notice that these points



(a) $\phi_{0,9}(x)$ (b) $(\phi_{0,9} - \phi_{0,11})(x)$

Figure 9: Representations of issues at the boundary. The plots include a line that points out the location $x = 10$ with a height of zero. The function $\phi_{0,9}$ is equal to $\frac{1}{6}$ at $x = 10$.

x are the exact center of the functions' supports, and the locations of the highest values of the functions. These are natural, individual, points to identify the location of a function $\phi_{j,k}$ or $\psi_{j,k}$. So, we will classify a scaling function or wavelet $\phi_{j,k}$ or $\psi_{j,k}$ as (mostly) inside Ω if this central value is inside Ω. We will keep independent the coefficients of scaling functions and wavelets that are (mostly) inside Ω. They will not be restricted in any way by the boundary conditions. The coefficients for those wavelets and scaling functions mostly outside the domain will have their values entirely dependent on those that are within. Those functions whose center value is exactly on the boundary will be set to zero. The relations between the coefficients will have to be set up so as to preserve the boundary conditions. The pattern for the coefficients $a_{0,k}$ associated with the $\phi_{0,k}$ scaling functions on the interval $(0, 10)$ will require the coefficients to be anti-symmetric around the boundary points. We get

$$\begin{array}{cccccccccccccc} \ldots & a_{-2} & a_{-1} & a_0 & a_1 & a_2 & \ldots & a_8 & a_9 & a_{10} & a_{11} & a_{12} & \ldots \\ \ldots & -a_2 & -a_1 & 0 & a_1 & a_2 & \ldots & a_8 & a_9 & 0 & -a_9 & -a_8 & \ldots \end{array} \tag{6.2.2}$$

with the coefficients on top and their assigned values on the bottom. Notice that those before a_1 and after a_9 are dependent on $a_1, \ldots, a_9$. It it easier to show that it works when only a single coefficient $a_{j,k}$ is considered. Take the coefficient a_k, related to $\phi_{0,k}$, (any resolution will do, so we will use the base resolution) and the boundary

$x = 10$. The setup gives us $a_{10+(10-k)} = a_{20-k} = -a_k$. The value of the resulting function at $x = 10$ is

$$\begin{aligned} a_k\phi_{0,k}(10) - a_k\phi_{20-k}(10) &= a_k\left[\phi(10-k) - \phi(10-(20-k))\right] \\ &= a_k\left[\phi(10-k) - \phi(k-10)\right] \\ &= a_k\left[\phi(10-k) - \phi(10-k)\right] = 0 \end{aligned} \tag{6.2.3}$$

using the fact that $\phi = \phi_{0,0}$ is symmetric around zero. This will be true for all $k = 1, 2, \ldots, 8, 9$ in V_0, and for $k = 1, \ldots, (10)2^j - 1$ for V_{-j}. We conclude that the boundary value will be kept to zero.

This setup, basically, leads to the function itself, $u(x,t)$, having values outside the domain, dependent upon values within. The values of u around the boundaries are

$$u(-x,t) = -u(x,t) \quad \text{and} \quad u(10-x,t) = -u(10+x,t), \tag{6.2.4}$$

so $u(x,t)$ is anti-symmetric at $x = 0$ and 10. Again, this sets the value at the boundary to zero, which is what we want, and also keeps $u_{xx}(0,t) = u_{xx}(10,t) = 0$. This will prove useful for most of our test problems.

We have not mentioned the ψ related coefficients yet, as they are not operated upon quite the same way. Recall that if ϕ is symmetric at some integer values, then the corresponding ψ will be symmetric (or anti-symmetric) around of $k + \frac{1}{2}$, $k \in \mathbb{Z}$. In our current case, $\psi_{0,0}$ is symmetric around $x = \frac{1}{2}$. Like with the a_k, the coefficient b_k will be kept independent if ψ_k is centered in $(0, 10)$. Those centered outside $(0, 10)$ will be dependent. Continuing with $\psi_{0,0}$ being centered at $x = \frac{1}{2}$, this means we will include $k = 0, \ldots, 9$ for W_0 and $0 \ldots 9 \cdot 2^j - 1$ for W_{-j}. Since the ψ are symmetric around $k + \frac{1}{2^{j+1}}$, to cancel them out requires a different pattern than for ϕ. As before, there will be an anti-symmetric pattern around $x = 0$ and $x = 10$. Unlike before, no ψ is actually centered at $x = 0$ or $x = 10$. For W_0 we get

$$\begin{array}{ccc|cccc|ccc} \ldots & b_{-2} & b_{-1} & b_0 & b_1 & \ldots & b_8 & b_9 & b_{10} & b_{11} & \ldots \\ \ldots & -b_1 & -b_0 & b_0 & b_1 & \ldots & b_8 & b_9 & -b_9 & -b_8 & \ldots \end{array} \tag{6.2.5}$$

with the vertical lines marking the locations of $x = 0$ and $x = 10$. Again, this cancels out all even derivatives and the function itself at the boundaries. This is perfect for

heat and heat related systems, but not so much for others. The question of what to do for non-zero boundary values will be covered in Section 6.5. An expansion of this concept, suitable for different derivatives, and more general ϕ and ψ, can be found in Section B.4.

In application, these linear relations between functions within and outside the domain are found virtually everywhere. Consider a matrix for converting $\phi_{0,k}$ related coefficients into those for $\phi_{-1,k}$. The columns will be composed of ordered coefficients h_k. In the new setup, the fact that $a_{0,-1} = -a_{0,1}$ (with $x = 0$ still the boundary) means that the $a_{0,1}$ ($\phi_{0,1}$ related) column is going to be, effectively, the $\phi_{0,-1}$ column as well. Each element of the column will need to include the effect of $\phi_{0,-1}$ on each $\phi_{-1,k}$. The column will have elements

$$\left[\begin{array}{ccccccc} (h_{-1} - h_3) & , & (h_0 - h_4) & , & (h_1 - h_5) & \dots \end{array} \right]^T . \tag{6.2.6}$$

Note that the term that might come to the left of the first element in (6.2.6) could be $h_{-2} - h_2$, following the pattern. The ϕ are symmetric, so the h_k are as well, meaning that difference would work out to be zero. This makes sense, because the $\phi_{j,0}$ related coefficients should be kept at zero. Suffices to say that the effect of this boundary setup requires building nearly all matrices with this effect in mind.

For the moment, we need a word about the number of elements in our spaces V_j and W_j. We will deal with one dimension, with a domain $\Omega = [0, N]$, $N \in \mathbb{N}$. The V_0 space for Ω is actually going to be composed of any functions $\phi_{0,k}$ whose support intersects the interior of Ω, so the number of terms will be dependent upon ϕ. If we specify boundary conditions, we will have a more specific result. If we have zero boundary conditions ($u = 0$) then we will have the terms outside the boundary dependent upon those inside, and the terms on the boundary will be zero. As a result, we can characterize any $u \in V_0$ using $N - 1$ terms (the $N + 1$ terms within or at the boundary, but we do not bother to keep track of the two that are directly on the boundary since they are always zero). If we look at V_j, for $j < 0$, we get $2^{-j}N - 1$, the $2^{-j}N + 1$ terms using the resolution of 2^j, and remove those on the boundary. So, the cardinality of a V_j space is simply $N \cdot 2^{-j} - 1$. The W_j spaces are simpler in this respect. For $j \leqslant 0$ the functions $\psi_{j,k}$ will be centered at $2^j + k2^{j+1}$ points. There

will, therefore, be $N \cdot 2^{-j}$ in any W_j for $j \leqslant 0$. As a result,

$$|V_0 + W_0 + W_{-1} + \cdots + W_{J+1}| = (N-1) + \sum_{j=0}^{-(J+1)} N2^j = 2^{-J}N - 1 = |V_j|\,, \quad (6.2.7)$$

(recall that $J < 0$) so the cardinality stays consistent, however we write the spaces.

6.3 The Heat Equation

Our first example will be the heat equation: $u(x,t)$ with $x \in (0,10)$ such that

$$u_t = u_{xx}, \qquad u(0,t) = u(10,t) = 0, \qquad u(x,0) = \mathbf{1}_{[3,6]}(x), \quad (6.3.1)$$

(with $\mathbf{1}_{[3,6]}$ the indicator function of $[3,6]$, see glossary). Notice that the initial condition is discontinuous. The solution is therefore the function $u(x,t)$ satisfying $u_t = u_{xx}$ for $x \in (0,10)$ and $t > 0$ that converges to $\mathbf{1}_{[3,6]}$ as $t \to 0^+$. We are using (6.3.1) as a test problem for two reasons. First, the discontinuities in $u(x,0)$ cause substantial instability, making the problem difficult to solve accurately using numerical time stepping schemes. Basically, this is a more stringent test than a problem with a smooth initial condition. The second reason is that this problem is actually very easy to solve via Fourier decomposition, as we will see next.

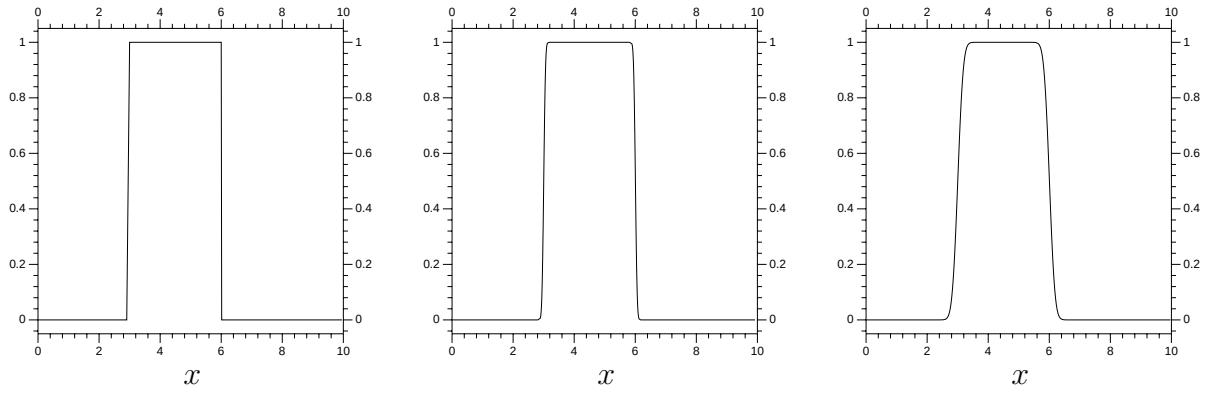
A Fourier decomposition (see Section A.2) of the initial condition will yield a series of the form

$$\mathbf{1}_{[3,6]}(x) = \sum_k f_k \sin\left(\frac{\pi k}{10}\, x\right), \qquad f_k \in \mathbb{R}. \quad (6.3.2)$$

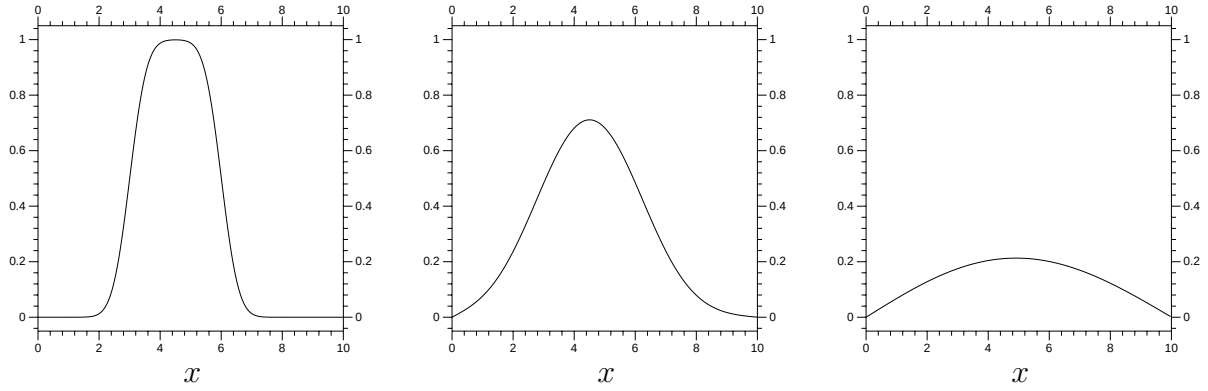
Using that series, the solution to the problem can be written in the form

$$u(x,t) = \sum_k g_k(t)\, \sin\left(\frac{\pi\, k}{10}\, x\right) = \sum_k \left(f_k e^{-\left(\frac{\pi\, k}{10}\right)^2 t}\right) \sin\left(\frac{\pi\, k}{10}\, x\right). \quad (6.3.3)$$

Figure 10 has some plots from $t = 0$ to $t = 1$ based on this Fourier decomposition. These results are, in fact, derived numerically. First, the the coefficients f_k from (6.3.2) are calculated directly via inner products. Next, the $g_k(t)$ in (6.3.3) are calculated for the given t. Finally, the IFFT is used to get the approximation for $u(x,t)$.



(a) $u(x, 0)$ (b) $u(x, 0.001)$ (c) $u(x, 0.01)$



(d) $u(x, 0.1)$ (e) $u(x, 1)$ (f) $u(x, 10)$

Figure 10: Fourier series derived solutions for our heat equation problem. Each of these is numerically derived using $k = 0, \ldots, 640$ in Equation (6.3.3).

The approximations used in this section use 640 coefficients f_k. By $t = 0.001$, the function $g_{640}(t)$ will have the value

$$f_k \, e^{-\frac{\pi^2}{10^2} \cdot (640)^2 \cdot 0.001} \approx f_k \, 2.77 \cdot 10^{-18}.$$

So, these will suffice for control results. Now to work towards solving (6.3.1) using wavelets.

Our first concern is to calculate $\frac{\partial^2}{\partial x^2}$ of ϕ and ψ. We will use a collocation based method (discussed in Section 5.3, near Equation (5.3.9)), which requires exact derivative values at specific points on the domain. These can be calculated using the matrix created by Example C.3.3. Combine that with the inverse of a plotting matrix (again,

around Equation (5.3.9)), and we get

$$M_{Der} = M_{FWT} \times (P_j)^{-1} \times P_j^2 \times M_{IFWT}, \tag{6.3.4}$$

a matrix that approximates the derivative. Check Figure 11 for the resulting approximations of $\frac{d^2}{dx^2}\phi(x)$. Every function we use will be composed of functions $\phi_{j,k}$, so M_{Der} is all that is needed for approximating $\frac{\partial^2}{\partial x^2}$.

So, we have a matrix that approximates the second derivative of a

$$\left[W_{j+1}, W_{j+2}, \ldots, V_0\right]^T$$

decomposition of any $f(x) \in V_j$. Next, we need a time stepping scheme. We choose the second order Adam's Moulton method:

$$u^{n+1} = u^n + \frac{h}{2}\left(\frac{\partial}{\partial t}u^n + \frac{\partial}{\partial t}u^{n+1}\right). \tag{6.3.5}$$

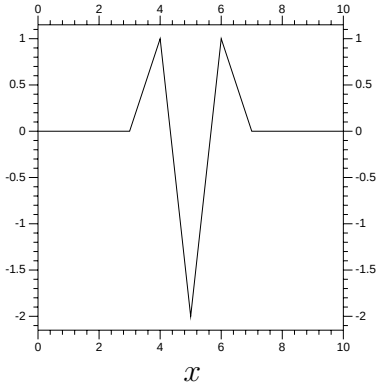
Using the matrix M_{Der} from above, we can write the time step calculation as

$$\mathbf{u}^{n+1} = \left[\left(I - \frac{h}{2}M_{Der}\right)^{-1}\left(I + \frac{h}{2}M_{Der}\right)\right]\mathbf{u}^n. \tag{6.3.6}$$

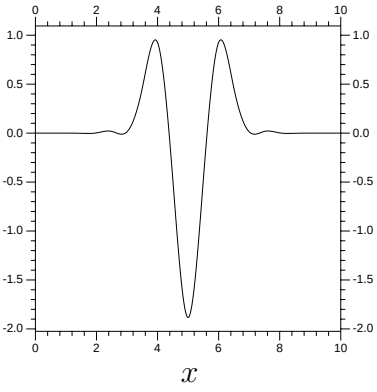
The second order Adams' Moulton method is what is called asymptotically stable. This means that any convergent differential equation will have a convergent numerical solution using the second order Adams' Moulton method (see any text on numerical methods for differential equations, like [16]). The method is ideal for stiff problems, and its stability means that we need not complicate matters by including adaptive time stepping, for instance. This does not mean that we do not need to be careful with the time step used. We will stick with a very small $h = 0.001$ time step, mostly to manage the discontinuous initial condition. Results using V_{-3} can be found in Figure 12, and the error at $t = 1$ for that, and other, resolutions can be found in Figure 13. Finally, Table 3 summarizes the L_2, L_1 and L_∞ errors for these approximations.

6.4 Modified Advection

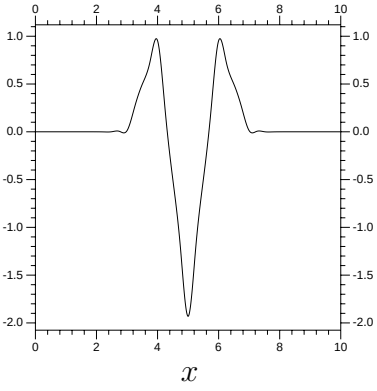
One further requirement is multiplication by a time independent function, call it $g(x)$. It is generally best to create a linear operator T_g to approximate the multiplication,



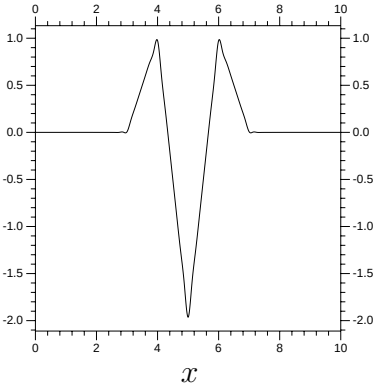
(a) Exact $\frac{d^2}{dx^2}\,\phi(x)$



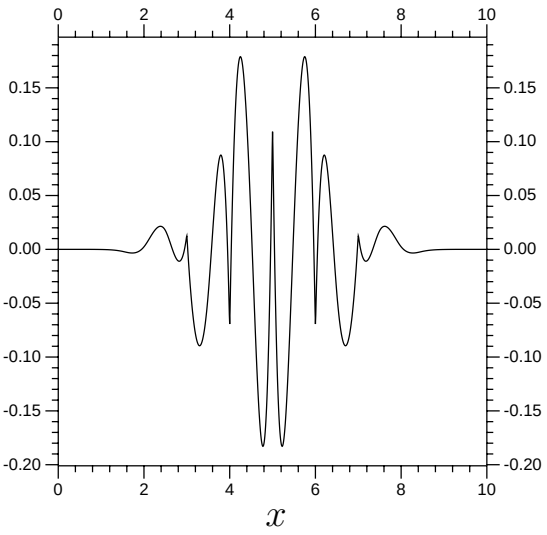
(b) V_{-1} approximation



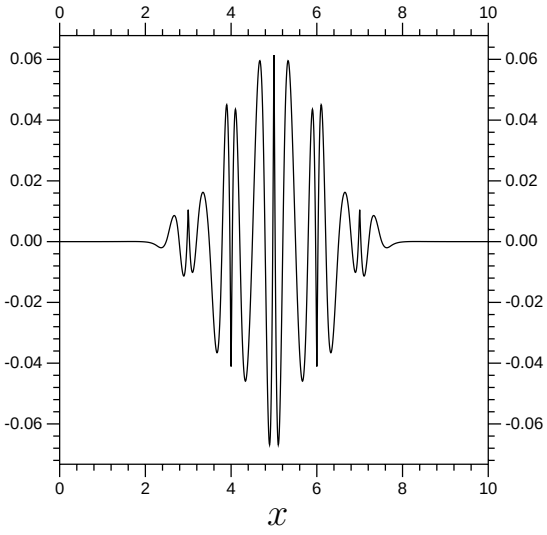
(c) V_{-2} approximation



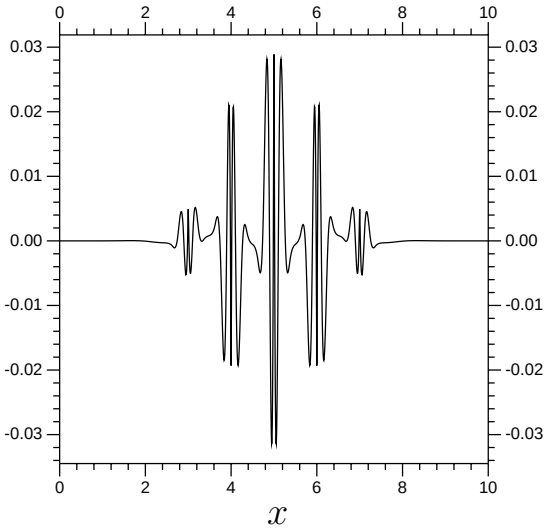
(d) V_{-3} approximation



(e) V_{-1} error

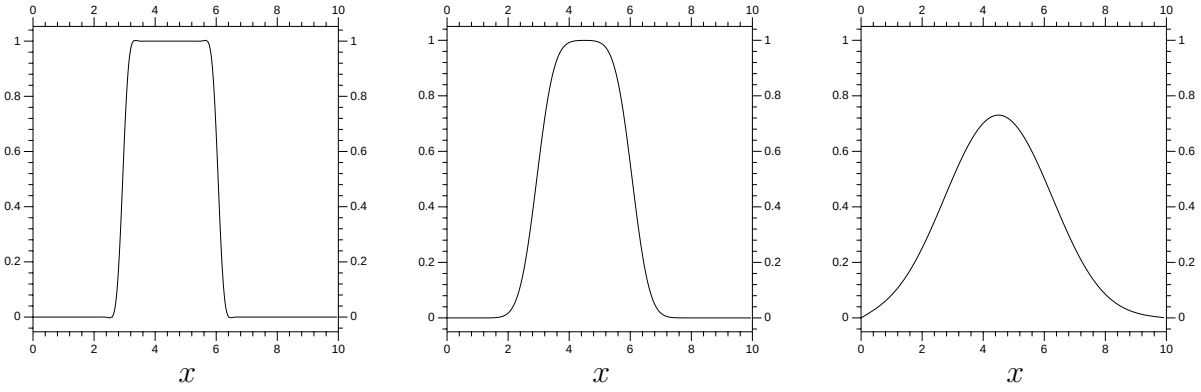


(f) V_{-2} error



(g) V_{-3} error

Figure 11: Approximations of $\frac{\partial^2}{\partial x^2}\,\phi(x)$, using $\phi = B_3$ and different V_j spaces. The plot of $\frac{\partial^2}{\partial x^2}\,\phi(x)$ in a) contains exact values calculated at the points $x = 2^{-8}k$. The error is just that, the difference between the exact values and the approximated values.



(a) V_{-3} result at $t = 0.01$ (b) V_{-3} result at $t = 0.1$ (c) V_{-3} result at $t = 1$

Figure 12: Heat equation results using V_{-3}. The time step is $h = 0.001$.

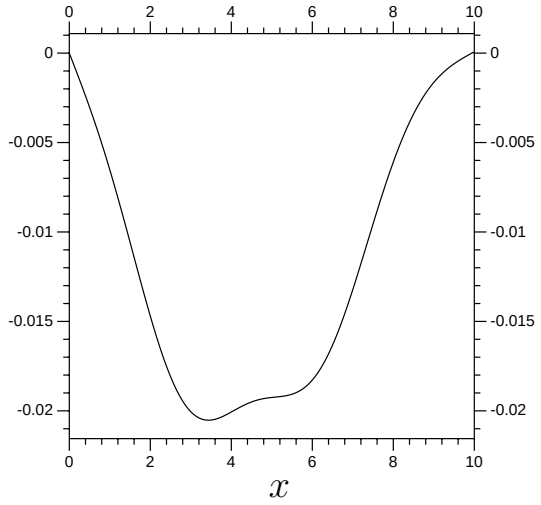
Table 3: Heat equation error

Resolution	L_∞ Error	L_1 Error	L_2 error
V_{-3}	$2.052 \cdot 10^{-2}$	$1.204 \cdot 10^{-1}$	$4.453 \cdot 10^{-2}$
V_{-4}	$1.062 \cdot 10^{-2}$	$5.900 \cdot 10^{-2}$	$2.205 \cdot 10^{-2}$
V_{-5}	$5.614 \cdot 10^{-3}$	$2.835 \cdot 10^{-2}$	$1.084 \cdot 10^{-2}$
V_{-6}	$3.103 \cdot 10^{-3}$	$1.310 \cdot 10^{-2}$	$5.369 \cdot 10^{-3}$

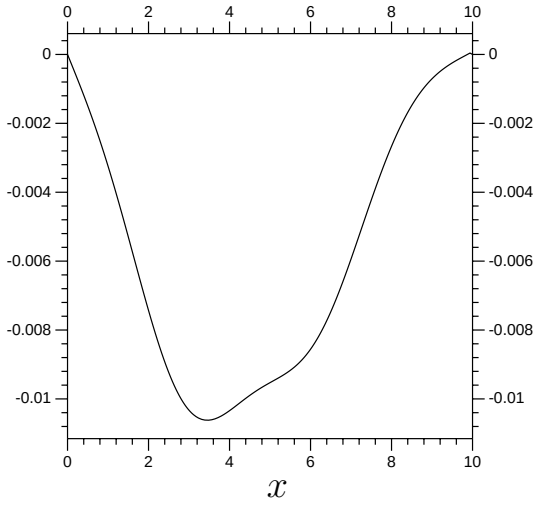
which can actually be done fairly easily. The procedure is nearly identical to the collocation method for derivatives (Section 5.3, near Equation (5.3.9)), and has the expression

$$T_g = M_{FWT} \times (P_j)^{-1} \times G \times P_j \times M_{IFWT}, \tag{6.4.1}$$

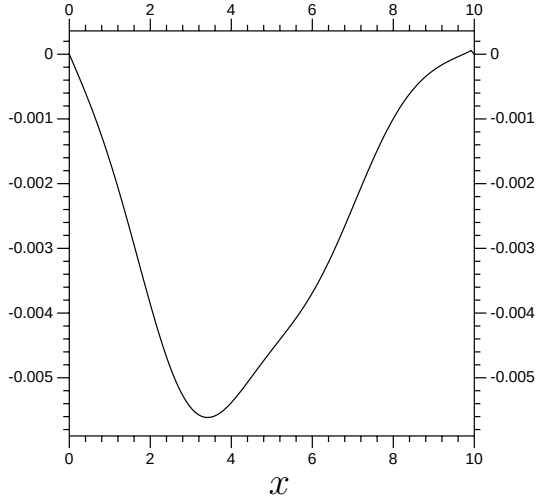
with G a matrix with the values of g down the diagonal, and otherwise zero. The location of the values would be dependent upon the V_j space used, almost certainly regularly spaced every 2^j (remember, j is going to be negative when V_j has fine resolution). Now to look at what happens when a set of function values u is multiplied by T_g. The first multiplication is u by M_{IFWT}, which converts the u vectors' coefficients into their equivalent $a_{j,k}$ ($\phi_{j,k}$ related) terms. Next, those coefficients are converted into the physical values (via P_j), which are then multiplied by $g(x)$ (at the finest resolution only, using the matrix G). Finally, the approximation for $g(x)u$ is converted back to the wavelet arrangement ($M_{FWT} \times (P_j)^{-1}$). The main issue then becomes the accuracy of that multiplication. Recall that, as usual, we will be taking the approximation of the product back to the original resolution, which includes a



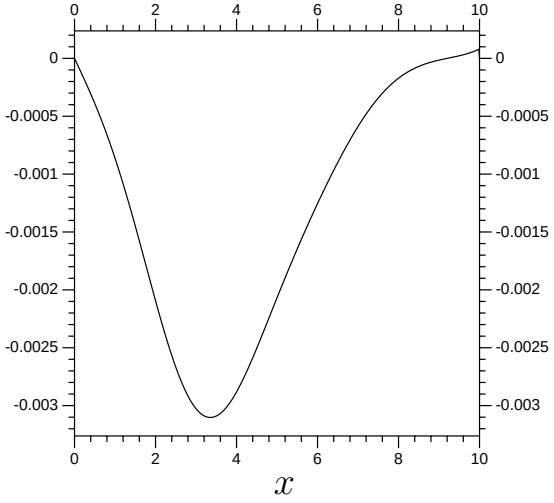
(a) Error for a V_{-3} numerical solution



(b) Error for a V_{-4} numerical solution



(c) Error for a V_{-5} numerical solution

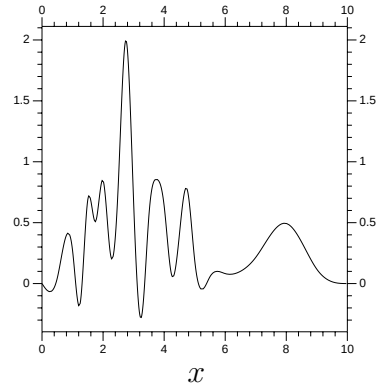


(d) Error for a V_{-6} numerical solution

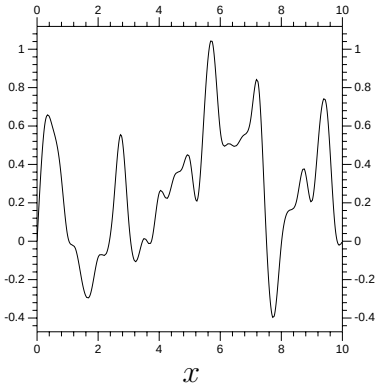
Figure 13: Error for wavelet based approximations of our heat equation problem, so the difference between the wavelet based approximation and the control values from Figure 10. The time step is $h = 0.001$, and these errors are all from $t = 1$. Table 3 has the norms of these errors.

certain amount of inherent error. If we calculate the matrix T_g using the method above and then approximate the multiplication by another function f, we get the result in Figure 14.

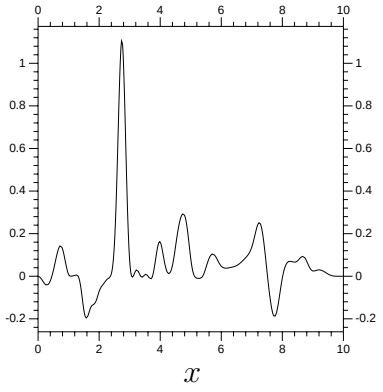
Next we take a look at a time dependent problem using this multiplication, a modified advection problem. In a normal advection problem, the values of the initial conditions (f, in this case) move over time, with a $\frac{\partial x}{\partial t}$ slope equal to the multiplier



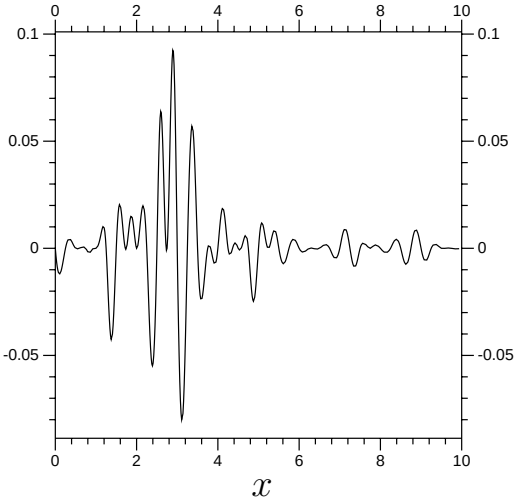
(a) $f(x)$



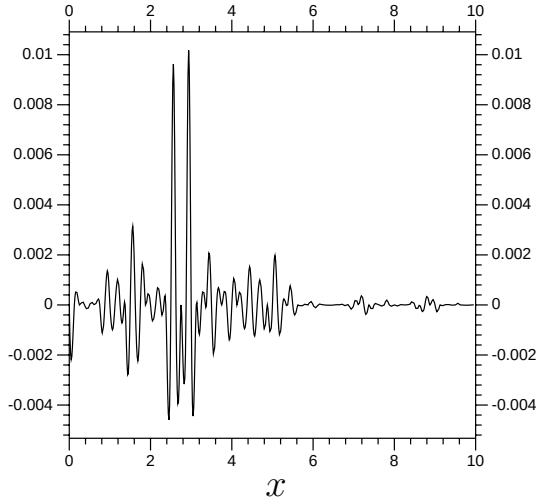
(b) $g(x)$



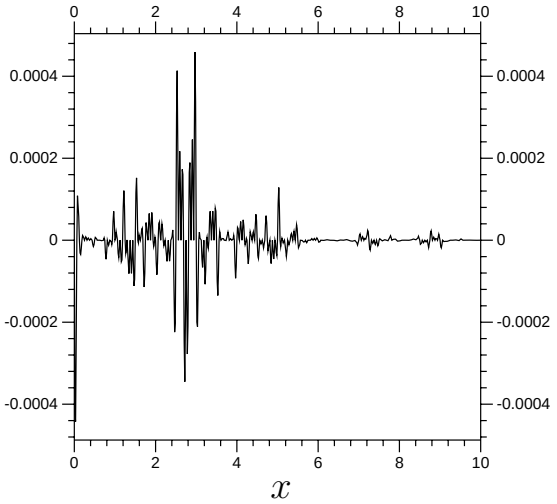
(c) $(fg)(x)$ in 2^6 resolution



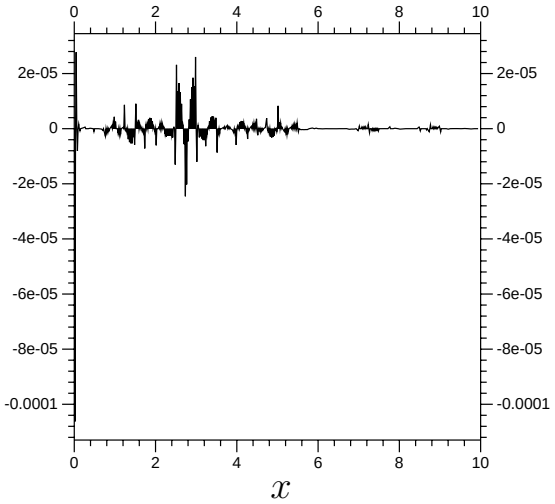
(d) Error in V_{-2}



(e) Error in V_{-3}



(f) Error in V_{-4}



(g) Error in V_{-5}

Figure 14: A look into the effectiveness of collocation based multiplication. Two functions, f and g, in V_{-2} are used, and tested at 2^6 per unit resolution in V_{-2} to V_{-5}. The errors are just $\big((fg)_{V_j} - (fg)_{V_{-6}}\big)$, from $j = -2$ to $j = -5$. Notice that the error is effectively zero at each $\frac{1}{2^j}$ point for each V_j space.

for the u_x term. The result is the solution $u(x,t) = f(x-ct)$, with c a constant. The initial values $f(x)$ move along the characteristics, which have $\frac{\partial x}{\partial t}$ slopes equal to c. Our problem is significantly more difficult:

$$\begin{aligned} &u_t(x,t) = -g(x)u_x(x,t), \qquad u(x,0) = f(x), \quad \text{with} \\ &f(x) = \frac{x(x-2)(x-8)(x-10)}{200}, \qquad g(x) = \frac{x(x-5)(x-10)}{40}. \end{aligned} \tag{6.4.2}$$

Solving (6.4.2) uses the fact that the values of $f(x)$ move along the characteristics with $\frac{\partial x}{\partial t}$ slopes equal to $g(x)$. To solve the path of a characteristic, call it $c(t)$, over time we have to solve

$$c'(t) = \frac{1}{40}\, c(t)\big(c(t)-5\big)\big(c(t)-10\big), \tag{6.4.3}$$

a fairly basic separable differential equation. One integration later, we have

$$\big(c(t)-5\big)^2 = \frac{25}{1+e^{\frac{5}{4}t+K}}, \tag{6.4.4}$$

notice that for any $c(0)$ value in $(0,10)$ the distance between $c(t)$ and $x=5$ will go to zero as $t \to \infty$. Equation (6.4.4) can be used to find exact values for $u(x,t)$ (the code can be found in Example C.4.1). Some exact solutions plotted at 2^6 resolution can be found in Figure 15.

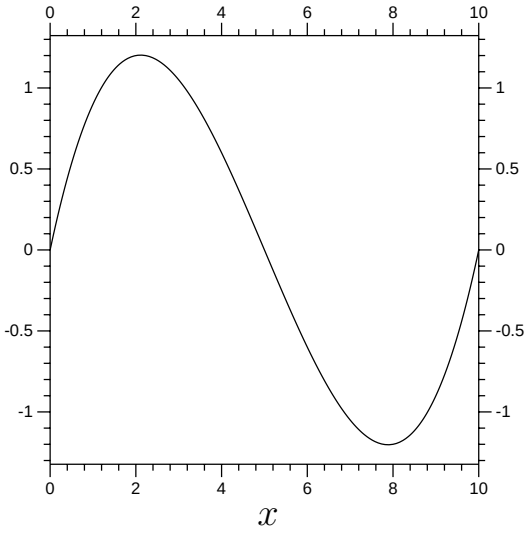
Using these exact results, we can check the accuracy of our numerical results. The time step is calculated using

$$u^{n+1} = u^n + \frac{h}{2}\left(-T_g M_x\, u^{n+1} - T_g M_x\, u^n\right), \tag{6.4.5}$$

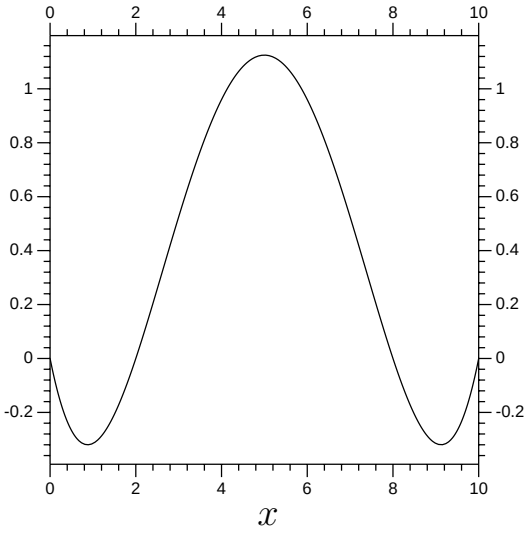
with $h = 0.01$, further simplified to

$$u^{n+1} = \left(I + \frac{h}{2}\, T_g M_x\right)^{-1} \left(I - \frac{h}{2}\, T_g M_x\right) u^n. \tag{6.4.6}$$

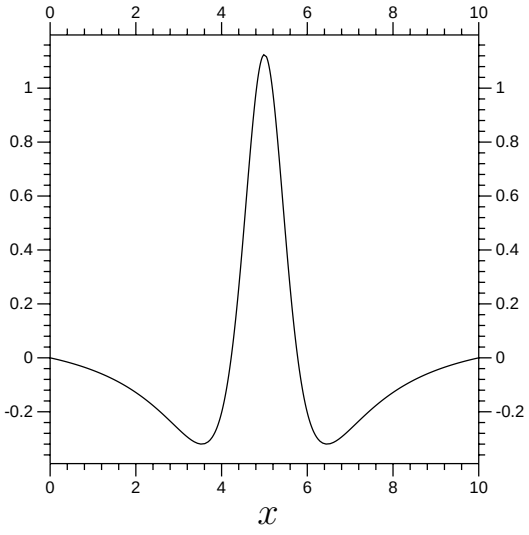
The key matrices are T_g and M_x, calculated as in (6.4.1) and (5.3.9). The code used to calculate the numerical results is Example C.4.2. Notice from Figure 15 that the migration of the $f(x)$ values towards $x=5$ results in very steep, small-scale, values in the center. So, any finite resolution will prove unable to model the



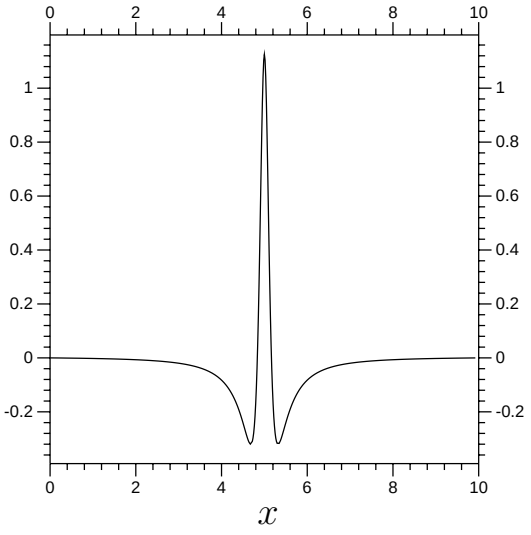
(a) $g(x)$



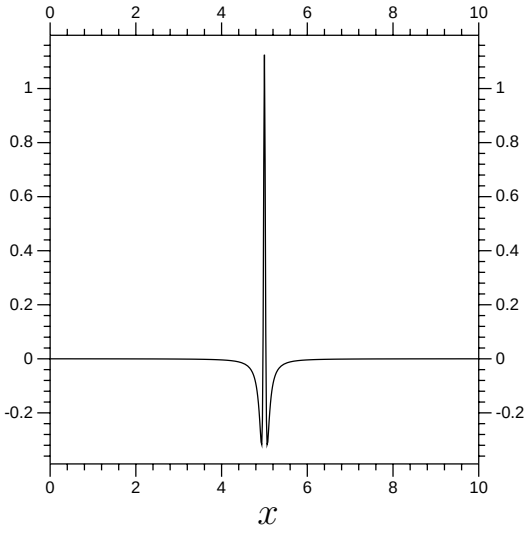
(b) $f(x)$



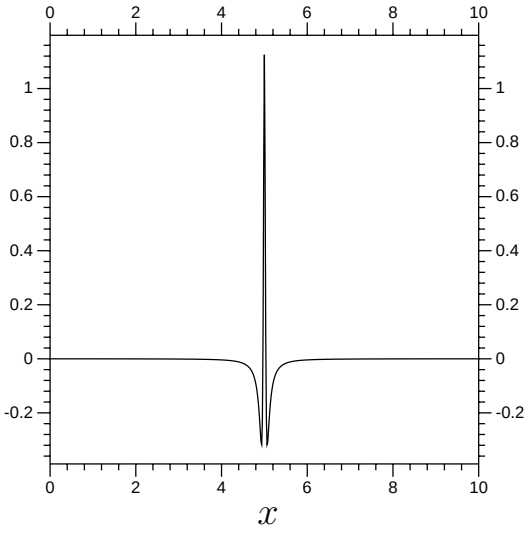
(c) Solution at $t = 2.5$



(d) Solution at $t = 5.0$

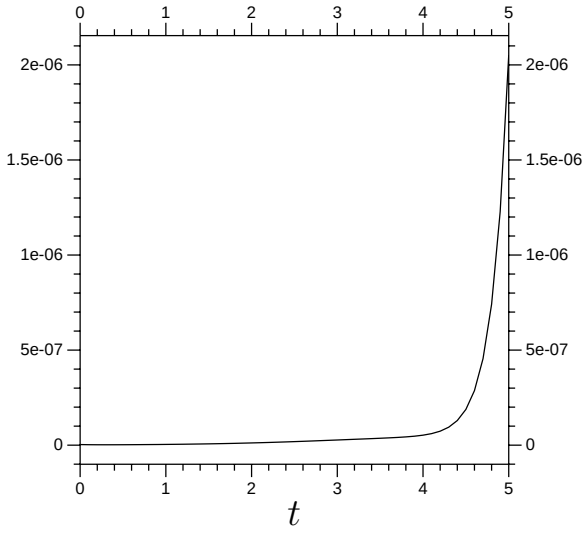


(e) Solution at $t = 7.5$

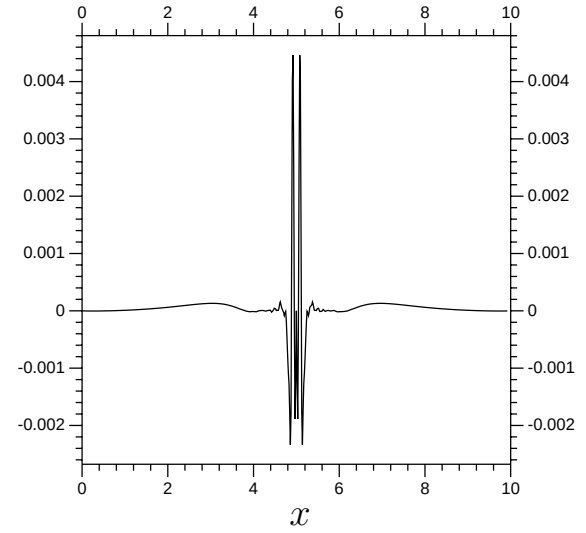


(f) Solution at $t = 10.0$

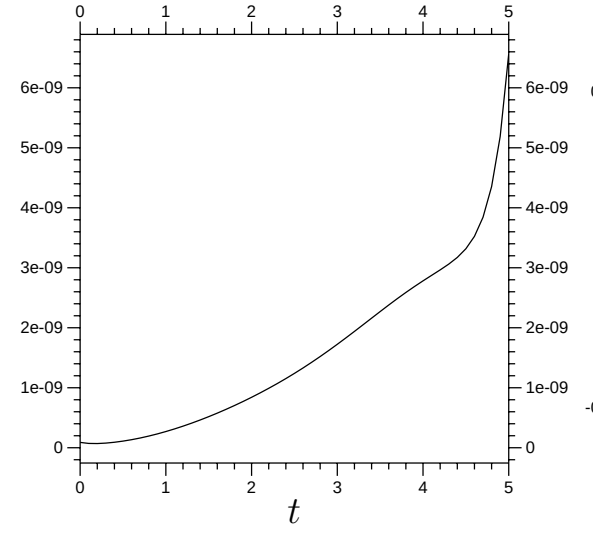
Figure 15: These are the exact solution of our advection diffusion problem, from (6.4.2), found via the method of characteristics. Notice that the shapes move towards the center of the domain, becoming more and more steep as the time increases.



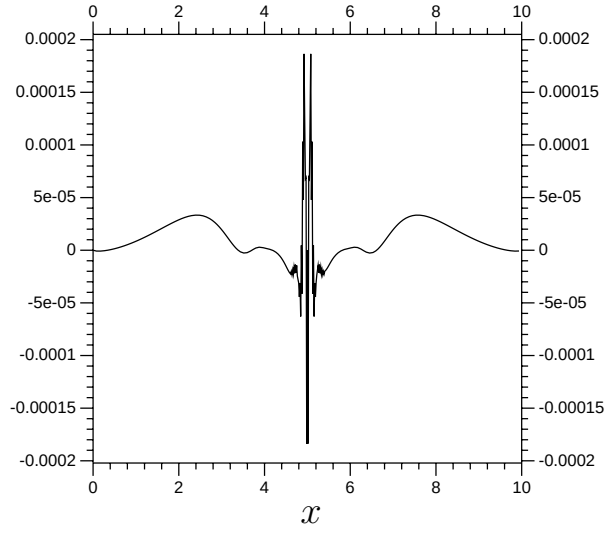
(a) L_2 error from the V_{-4} results



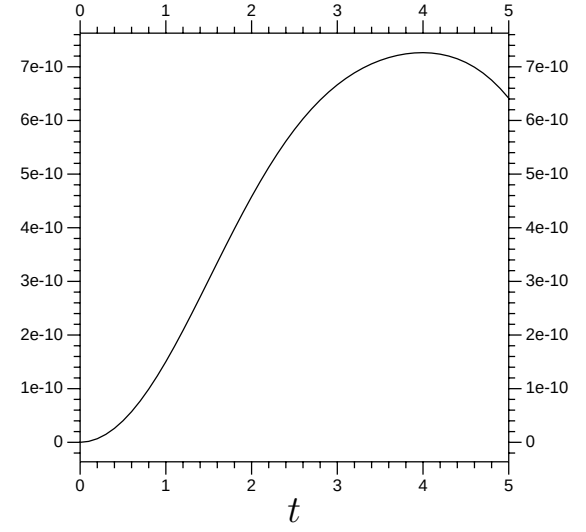
(b) Error from the V_{-4} results at $t = 5.0$



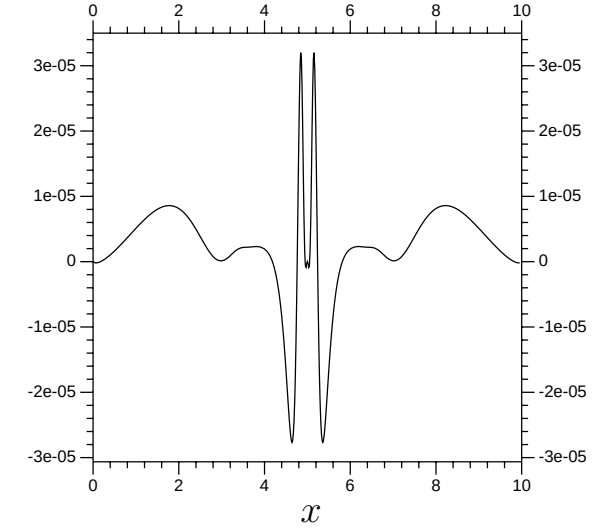
(c) L_2 error from the V_{-5} results



(d) Error from the V_{-5} results at $t = 5.0$



(e) L_2 error from the V_{-6} results



(f) Error from the V_{-6} results at $t = 5.0$

Figure 16: Errors from the numerical approximations of the modified advection diffusion problem, from (6.4.2). A time step of $h = 0.01$ was used.

problem, eventually. Over the time frame $t \in [0, 5]$, the results in V_{-4}, V_{-5} and V_{-6} are reasonable, see Figure 16. The results are clearly improving with greater resolution. So, the (6.4.1) model for the approximation of $g(x)u(x,t)$ multiplication is reliable, given sufficient resolution.

6.5 Non-Zero Boundary Conditions

We will eventually need two more elements related to boundary condition. The first is periodic boundary conditions, which will be handled via Fourier decomposition in the relevant direction (see Section A.2). Using periodic boundary conditions with wavelets is discussed in the appendices (Section B.3). This section will be about fixed, non-zero boundary conditions. This can be difficult when working with basis functions.

The basic idea is to divide the function into two separate functions. Take a fairly general setup

$$v(x,t), \quad x \in [0,10], \quad v(x,0) = f(x), \quad v(0,t) = a, \quad v(10,t) = 0, \quad v_t = Lv,$$

with L a linear differential operator. What we will do is separate $v(x,t)$ into two components: $v(x,t) = u(x,t) + B(x)$. The first, $u(x,t)$, is a V_j based function with zero boundary conditions:

$$u(0,t) = u(10,t) = 0.$$

There is one additional restriction we will place on u. It is constant at the boundary, so we will keep its time derivative to zero at the boundary, so

$$Lu(0,t) = Lu(10,t) = 0,$$

if u is to be consistent with the problem (this can also be viewed as $Lu(x,t) \to 0$ as $x \to 0$ or 10). This is not an additional set of boundary conditions, it is just an implication of the boundary conditions. Sections 6.2 and B.4 discuss a way to arrange these restrictions on u. The function $u(x,t)$ is the time dependent part of v. Next, we have $B(x)$, which we will usually call the boundary function of v. It is

time independent, and will have the values $B(0) = a$ and $B(10) = 0$. One further requirement of B is that it have $LB(0) = LB(10) = 0$. This arrangement means that $v(x,t) = u(x,t) + B(x)$ for every t, satisfying

$$v(0,t) = u(0,t) + B(0) = a, \qquad v(10,t) = u(10,t) + B(10) = 0$$

and

$$Lv(0,t) = Lu(0,t) + LB(0) = 0 = Lu(10,t) + LB(10) = Lv(10,t).$$

We insist that $Lv = 0$ on the boundary so that the constant boundary conditions are consistent with the time derivative.

A time step of size h using the second order implicit Adam's method would be

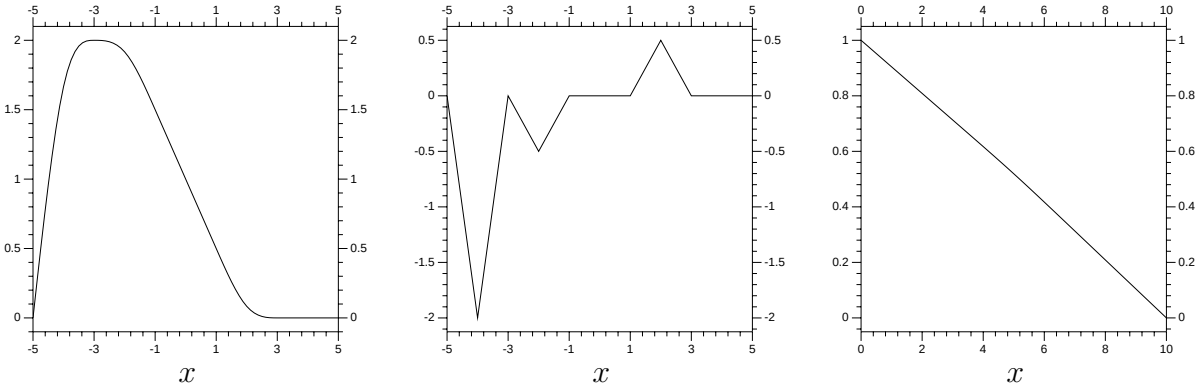
$$u(x,t+h) = u(x,t) + \frac{h}{2}\big[Lu(x,t) + Lu(x,t+h)\big] + h\big[LB(x)\big].$$

An example will prove helpful to clear up any confusion. We will return to the one-dimensional heat equation. Here is the problem:

$$v_t = v_{xx}, \quad x \in [0,10], \quad v(0,t) = 1, \quad v(10,t) = 0. \tag{6.5.1}$$

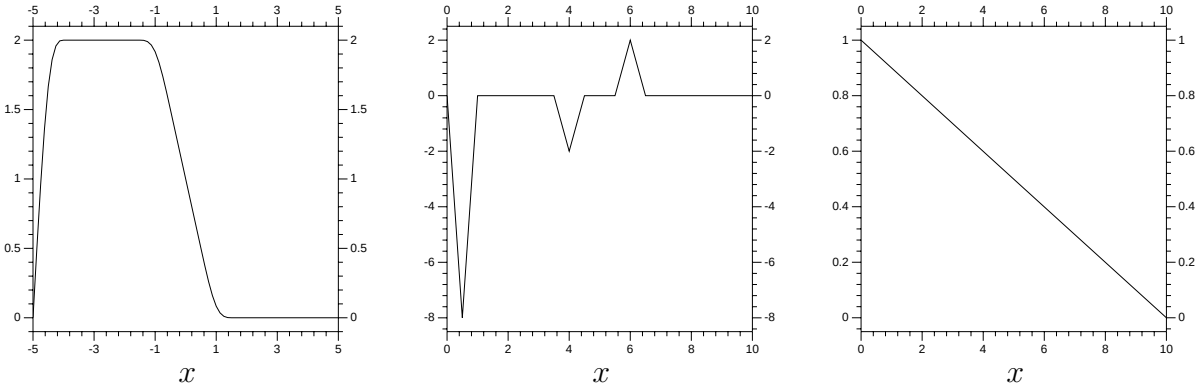
The exact solution converges to the linear function $\left(1 - \frac{x}{10}\right)$ as $t \to \infty$. We are mostly curious about the effect of the boundary function, so we will ignore initial conditions and time related matters and focus only upon the steady state that results. There are many options when it comes to setting up the boundary function. Figure 17 presents one that works, created with a simple sum of $\phi_{0,k}$ terms. The coefficients used are $a_0 = 1$, $a_1 = \frac{1}{2}$, $a_{-4} = \cdots = a_{-2} = 2$ and $a_{-1} = \frac{3}{2}$. Since the values of ϕ at integers all sum exactly to 1, this gives the required value at the central term. The resulting function also has a constant slope around $x = 0$, so it has a second derivative of zero at $x = 0$. The only possible issue is that this boundary function's support extends a bit much into the domain. We may want the boundary function to be limited to a small region near the boundary. Thankfully, the process can be done using the V_{-1} functions, $\phi_{-1,k}$, instead (Figure 18).

So, why did we go through all the trouble of making sure the second derivative is zero at the boundary? Take a look at what happens when we only use a multiple of



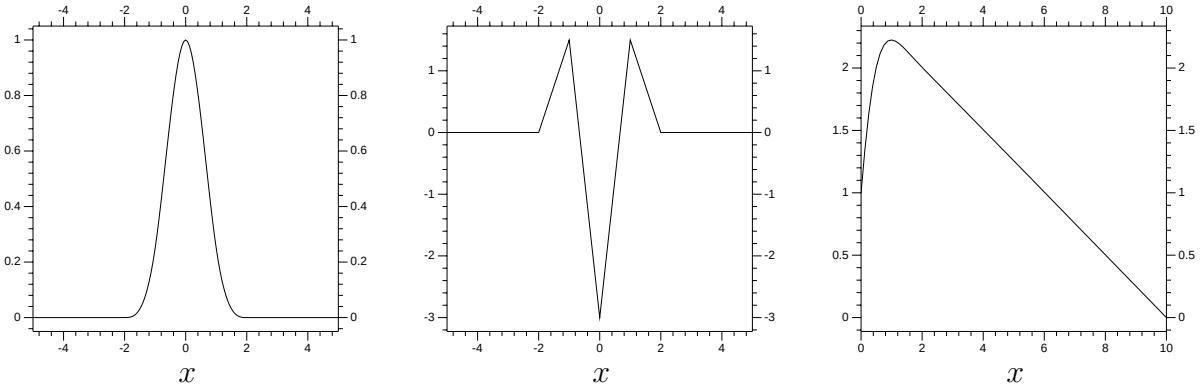
(a) Boundary Function (b) Second Derivative (c) Steady State

Figure 17: Non-zero boundary condition function for the problem in (6.5.1), its second derivative and the resulting steady state. The horizontal axis is the x variable. Notice that the boundary function has a value of 1 at $x = 0$ and is linear around $x = 0$ as well, hence a second spatial derivative of zero and a time derivative of zero, and consistency with the problem. The second derivative is angular because the scaling function and wavelet are only differentiable to that level.



(a) Boundary Function (b) Second Derivative (c) Steady State

Figure 18: Another non-zero boundary setup for the same heat equation problem, this one using higher spatial resolution. This is the same as before, just extending less into the domain. Notice that the function has a derivative value that is four times as high as the previous example.



(a) Boundary Function (b) Second Derivative (c) Steady State

Figure 19: What can happen when the boundary function is not consistent. The lack of consistency is, as one would expect, at the boundary itself (so $x = 0$). What we need is a time derivative of zero. If we do not have this, the numerical results will usually end up inconsistent with the problem.

$\phi_{0,0}$ to get $v(0,t) = 1$ (Figure 19). Clearly, this did not work at all. What was left out in using that boundary function was taking into account of its time derivative (equal to the second x derivative). The boundary function has a non-zero time derivative at the boundary, which makes the model inconsistent with the problem. The function $v(x,t)$ should be constant at $x = 0$, equal to the boundary function only. However, keeping in mind the restrictions we have placed on u,

$$v_{xx}(0,t) = u_{xx}(0,t) + B_{xx}(0) = B_{xx}(0) \neq 0,$$

resulting in a contradiction. Making boundary functions that have a particular height at the boundary and a time derivative of zero at the boundary is not that difficult. We could simply take two basis functions, say, $\phi_{0,0}$ and $\phi_{0,-1}$, for a boundary function $b(x) = a_0\phi_{0,0}(x) + a_1\phi_{0,1}(x)$. In the current, heat equation, example the coefficients a_0 and a_1 would be the solution to the equations

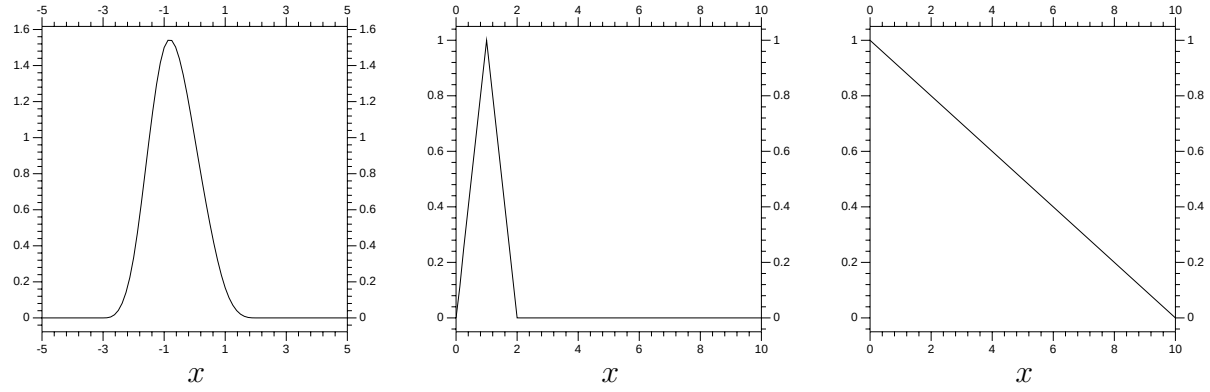
$$a_0\phi_{0,0}(0) + a_1\phi_{0,1}(0) = 1 \qquad \frac{\partial^2}{\partial x^2}\left(a_0\phi_{0,0} + a_1\phi_{0,1}\right)(0) = 0. \tag{6.5.2}$$

This does work, and keeps the resulting functions quite small (in terms of number of coefficients, just two). Some examples are given in Figure 20, for the heat equation example. The V_{-1} and V_{-2} examples work well, but using V_{-3}, so $b(x)$ based on functions $\phi_{-3,k}$, results in a distortion of the steady state.

The principle is the same in two dimensions, just with more calculations. Keeping the resolution low helps maintain stability, as before. The next element that will be required is a non-linear component, specifically a quadratic one. The simplest example to start with is Burgers' equation.

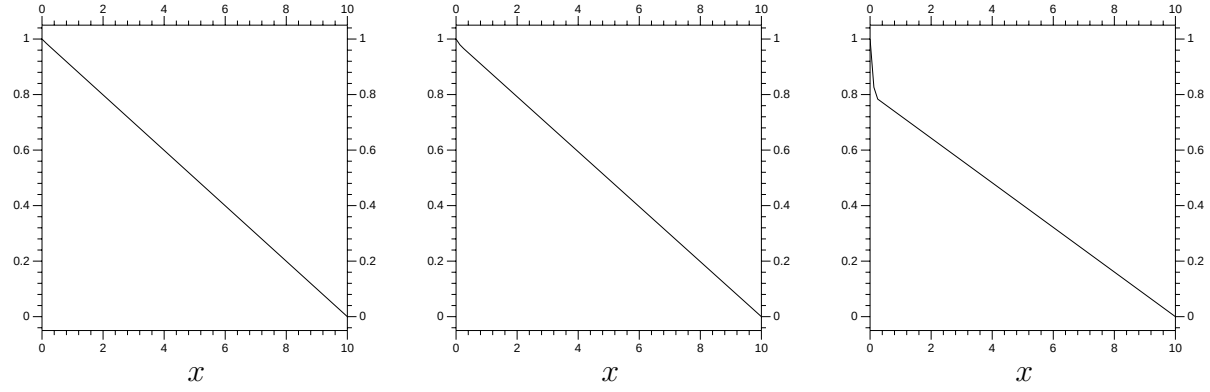
6.6 A Look at Burgers' Equation

Burgers' equation is one of the most basic non-linear partial differential equations, with a simple expression and intuitive solutions. Due to the equation's simplicity, and to some properties of its solutions we will shortly discuss, Burgers' equation is a popular choice for testing wavelet related methods. Our particular problem is actually very similar to that used in [28], where compatible solutions can be found.



(a) Boundary Function (b) Second Derivative (c) Steady State for V_0.

This uses a low resolution: the boundary condition function is in V_0.



(d) V_{-1} (e) V_{-2} (f) V_{-3}

Figure 20: The shape of a basic $a_0\phi_{j,0}(x) + a_1\phi_{j,-1}(x)$ boundary function for the heat equation, its second derivative, and resulting steady states for the resulting system. The first plot is the boundary function for V_0, the higher resolution examples will be narrower, but have pretty much the same shape. This example illustrates why it is generally better to use a low resolution boundary condition function. Note that these steady states were all calculated in V_{-3}.

The paper also discusses at length the advantages of using wavelet decompositions on this particular problem.

We consider the following Burgers' equation problem: find $u(x,t)$ on $x \in [0,10]$, $t \geqslant 0$, with

$$u_t = -uu_x + \nu u_{xx}, \quad u(x,0) = f(x), \quad u(0,t) = u(10,t) = 0. \tag{6.6.1}$$

First, a quick note about the problem. The initial condition $f(x)$, shown in Figure 21, is a function in V_0 using our usual ϕ (Figure 7). So that $f(x)$ is expressed accurately

no matter what V_j space is used, we will make $f(x) \in V_0$ with coefficients

$$a_1 = 0,\ a_2 = -1,\ a_3 = -2,\ a_4 = -1,\ a_5 = 0,\ a_6 = 1,\ a_7 = 2,\ a_8 = 1,\ a_9 = 0.$$

Since the functions ϕ are cubic splines (Section A.3) we can get a closed form of the initial condition if and when necessary. On to Burgers' equation. Technically classified as a 'quasilinear' equation, it is most convenient for us to discuss Burgers' equation in terms of the challenges it poses to our numerical approximation. We can create matrices, fixed matrices useable at every time step, to approximate spatial derivatives applied to u (operations $u(x,t) \to \frac{\partial^m}{\partial x^m}u(x,t)$), as discussed in Section 5.3. This is also true of $u(x,t) \to g(x)u(x,t)$, as seen with the modified advection diffusion problem (Section 6.4). We cannot do the same for $u \to u\frac{\partial}{\partial x}u(x,t)$, as it is non-linear.

The results of Burgers' equation are similar to those of an advection-diffusion equation, except for the influence of the non-linear term. Anything multiplied with the advection term (u_x) determines the speed of the advection. Burgers' equation has a u term multiplied with u_x, so u has speed of propagation dependent upon its magnitude. Points that have $u > 0$ move in the positive direction, those with $u < 0$ move in the negative direction. The greater the value of $|u(x)|$ at a point, the faster that point propagates, so curves can get steeper as they propagate. There is a simple solution for the initial value problems using the inviscid Burgers' equation ($\nu = 0$) using the method of characteristics, though it is an implicit one. If we have $u(x,0) = f(x)$ on $\mathbb{R}$, using $\nu = 0$, then

$$u(x,t) = f\big(x - u(x,t)t\big), \tag{6.6.2}$$

which can be verified relatively easily. Notice that each value in $u(x,t)$ can be found in the initial conditions, and that we can follow a particular value from f like so:

$$u\left(x + u(x,0)t, t\right) = u(x,0). \tag{6.6.3}$$

All this combines to make it trivial to follow a particular value of $f(x)$ over time, even though it can be exceedingly difficult to get an explicit function for $u(x,t)$. However, linear initial conditions work quite easily. For instance,

$$u(x,0) = x \quad \Longrightarrow \quad u(x,t) = \frac{x}{1+t}. \tag{6.6.4}$$

That particular initial condition had the positive values on the right, negative on the left, and they move away from each other, reducing the slope of the function. In contrast,

$$u(x,0) = -x \quad \Longrightarrow \quad u(x,t) = \frac{x}{t-1}. \tag{6.6.5}$$

Notice that even this simple example has $u(x,t)$ become undefined (a vertical line) as t approaches 1.

Our particular example is on the domain $\Omega = (0,10)$, with $u(0,t) = u(10,t) = 0$ boundary conditions. We will implement them in the manner described in Section 6.2. Since this will make $u_{xx} = u = 0$ on the boundary, we will get $u_t = 0$ on the boundary as well. We will include a viscosity of 0.01. Our time step is h, and the scheme will be the second order implicit Adam's method (Equation 6.3.5). This results in the scheme

$$u^{n+1} = u^n + \frac{h}{2}\left(\nu u_{xx}^{n+1} + \nu u_{xx}^n + u^{n+1}u_x^{n+1} + u^n u_x^n\right), \tag{6.6.6}$$

which is not easily solved (being non-linear and implicit). There are non-linear solvers for this type of system, but we will keep things simpler. We will use the approximation

$$u^{n+1}u_x^{n+1} + u^n u_x^n \approx u^{n+1}u_x^n + u^n u_x^{n+1}, \tag{6.6.7}$$

so it is now linear. This approximation will require more detail. Consider a function $u(x,t)$, with $u^n(x) = u(x,t)$ and $u^{n+1}(x) = u(x,t+h)$. What we want is to approximate

$$Au^n Bu^n + Au^{n+1}Bu^{n+1}, \tag{6.6.8}$$

with A and B linear operators. This would be consistent with using the second order implicit Adam's method on a problem with $u_t(x,t) = Au(x,t)\,Bu(x,t)$. So,

$$\begin{aligned}
&Au^n Bu^n + Au^{n+1}Bu^{n+1} \\
&= Au^n B\left(u^{n+1} - hu_t^{n+1} + \frac{h^2}{2}u_{tt}^{n+1}\right) + Au^{n+1}B\left(u^n + hu_t^n + \frac{h^2}{2}u_{tt}^n\right) + O(h^3) \\
&= Au^n Bu^{n+1} + Au^{n+1}Bu^n - hAu^n Bu_t^{n+1} + hAu^{n+1}Bu_t^n \\
&\qquad + \frac{h^2}{2}\left(Au^n Bu_{tt}^{n+1} + Au^{n+1}Bu_{tt}^n\right) + O(h^3).
\end{aligned} \tag{6.6.9}$$

This gets re-ordered for

$$\begin{aligned} & Au^nBu^{n+1} + Au^{n+1}Bu^n - hAu^nB\left(u_t^n + hu_{tt}^n\right) + hA\left(u^n - hu_t^n\right)Bu_t^n \\ & \qquad + \frac{h^2}{2}\left(Au^nBu_{tt}^{n+1} + Au^{n+1}Bu_{tt}^n\right) + O(h^3) \\ =& Au^nBu^{n+1} + Au^{n+1}Bu^n + h^2\left[\frac{1}{2}\left(Au^nBu_{tt}^{n+1} + Au^{n+1}Bu_{tt}^n\right)\right. \\ & \qquad \left. - Au^nBu_{tt}^n - Au_t^nBu_t^n\right] + O(h^3). \end{aligned} \tag{6.6.10}$$

So, the error is $O(h^2)$, and when included in a time stepping scheme will be multiplied by h, making the error $O(h^3)$. As a result, this approximation's error has the same order as the second order Adam's Moulton method.

Next we need the linear operators

$$T_u f \approx uf, \qquad \text{and} \qquad T_{u_x} f \approx u_x f, \tag{6.6.11}$$

created as in Equation (6.4.1), as well as the operators M_1 and M_2 that approximate the derivatives (see the end of Section 5.3, around Equation (5.3.9)). Finally, we can rewrite the time stepping scheme as

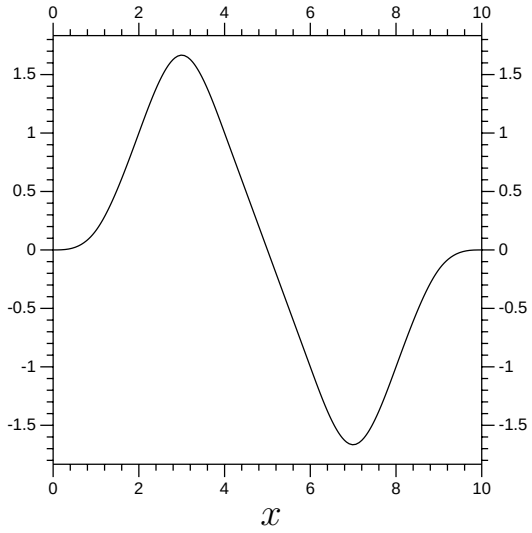
$$u^{n+1} = u^n + \frac{h}{2}\left(\nu M_2 u^{n+1} + \nu M u^n - T_u M_1 u_x^{n+1} - T_{u_x} u^{n+1}\right). \tag{6.6.12}$$

This can be rearranged for a linear implicit system

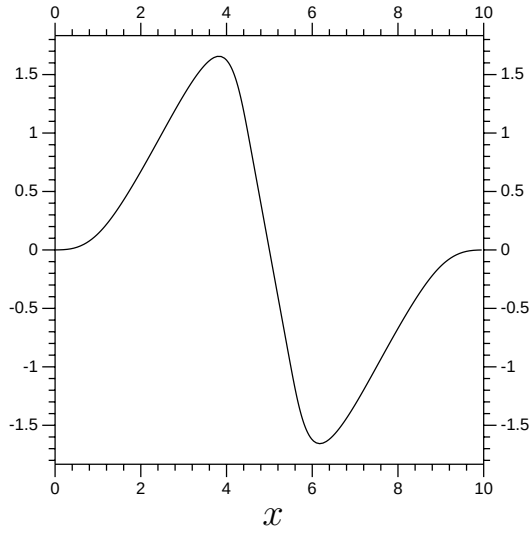
$$u^{n+1} = \left(I - \frac{h}{2}\nu M_2 + \frac{h}{2}T_{u_x} + \frac{h}{2}T_u M_1\right)^{-1}\left(I + \frac{h}{2}\nu M_2\right)u^n. \tag{6.6.13}$$

Our main approximation will use a resolution of 2^6 elements per unit, so V_{-6}. The results are quite reasonable, showing the propagation of the waves, their collision, then dissipation under the influence of the viscosity (See Figure 21). Using a coarser resolution gives clearly imperfect results. Figure 22 has plots from a V_{-5} model. Note the odd peaks near $x = 5$ once the waves collide. If a resolution coarser than V_{-5} is used, the results are actually unstable after the collision (see Figure 23). The basic code used for these calculations is in Example C.5.1.

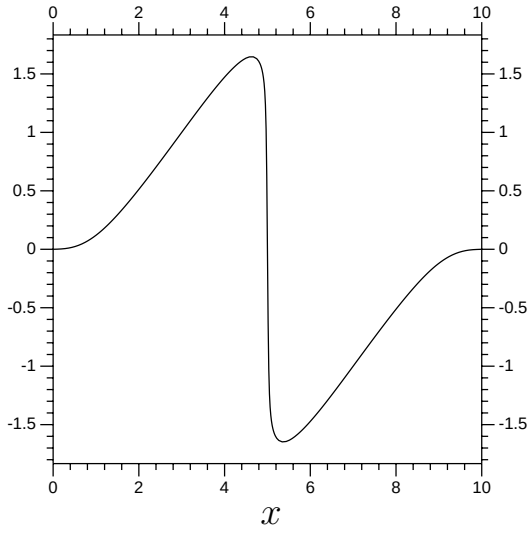
The nature of the problem results in the two peaks approaching each other, getting steeper as they go, and colliding. To be more precise, the characteristics intersect near



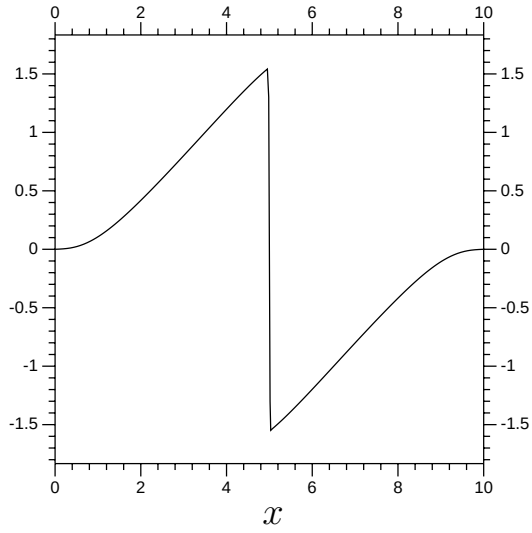
(a) $t = 0.0$



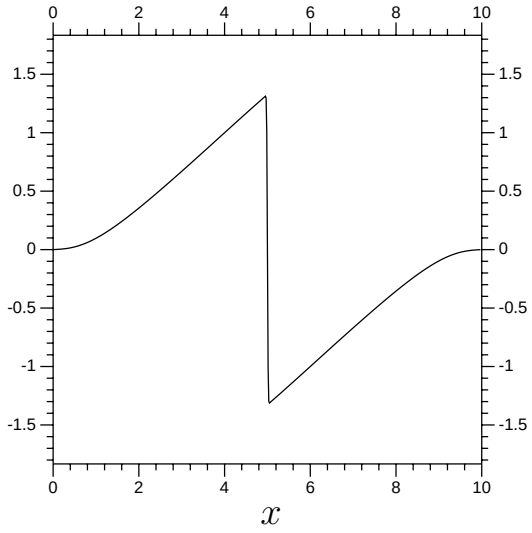
(b) $t = 0.5$



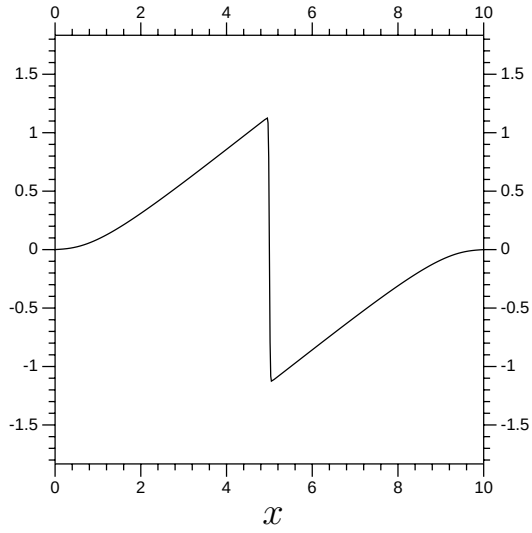
(c) $t = 1.0$



(d) $t = 1.5$

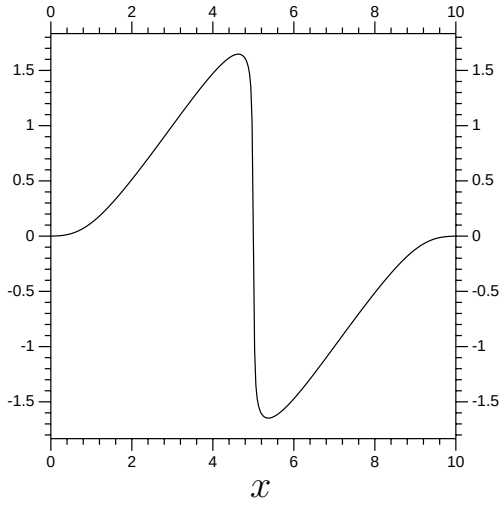


(e) $t = 2.0$

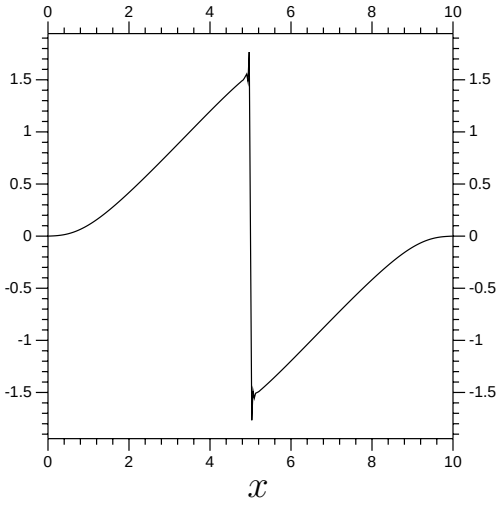


(f) $t = 2.5$

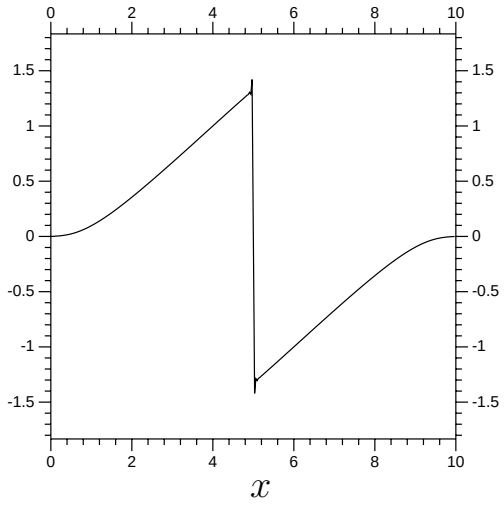
Figure 21: Burgers' equation results using a resolution of V_{-6} and a time step of $h = 0.01$.



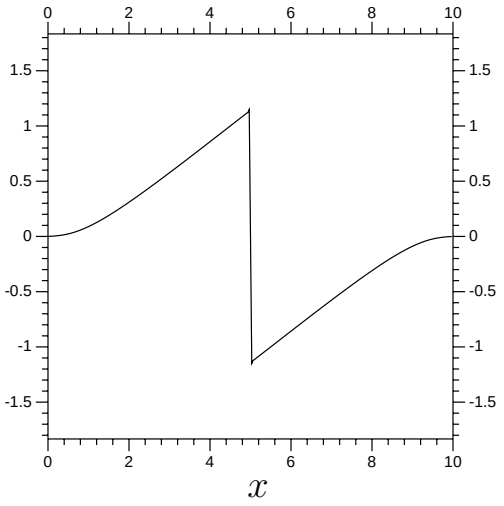
(a) $t = 1.0$



(b) $t = 1.5$

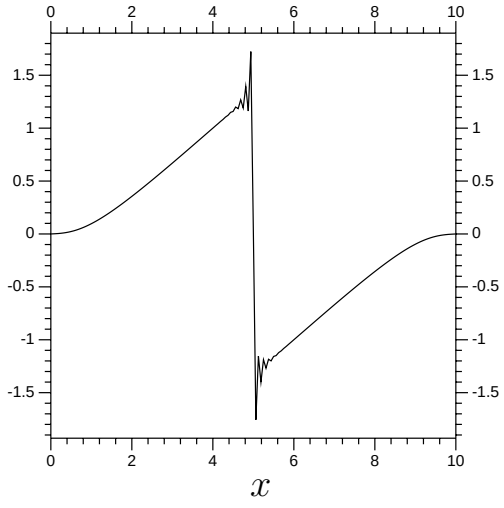


(c) $t = 2.0$

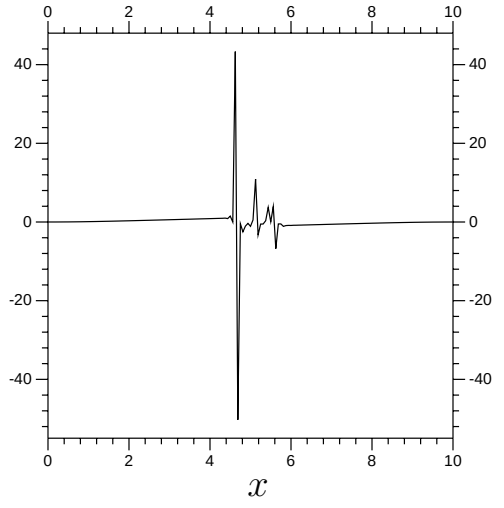


(d) $t = 2.5$

Figure 22: Burgers' equation results using a resolution of V_{-5}. We start at $t = 1.0$ since that is where the waves collide. Notice the odd spiked shapes near $x = 5$ for the later times.



(a) $t = 2.0$



(b) $t = 2.5$

Figure 23: Burgers' equation results using a resolution of V_{-4}. Notice they are not stable far past $t = 2.0$.

the center, resulting in a shock. Without viscosity (with $\nu = 0$) the solution becomes vertical near $t = 1$. Even with viscosity, the solution becomes very steep near $x = 5$, causing serious issues for the calculations. The resolution used is very important for this model. A resolution of V_{-6} gives good results. Reducing the resolution down to V_{-5} causes clear issues: shapes that should not be present. A further reduction to V_{-4} makes the results unstable. Increasing the resolution comes at a serious cost, however. Doubling the size of the system more than doubles the time required to solve the implicit system created by the time discretization. But, notice that the resolution looks to be fine for the regions nearer the boundary. We need only increase the resolution in the center. That is the next component we will acquire: localized higher resolution.

Chapter 7

Our Multi-Scale Method

7.1 Introduction and Motivation

Now we begin to get serious about taking advantage of the properties of wavelets. Everything up to this point is standard material. As far as can be told, the concept introduced in this chapter is an original idea.

While systems for approximating non-linear equations with wavelets are not rare, they are nearly all based upon explicit time stepping schemes. As has been stated, they take advantage of the multi-resolution decomposition purely to efficiently express the functions involved. Fewer coefficients means fewer calculations, means lower cost, and so forth. However, we are looking towards solving 'stiff' problems, which are best solved with implicit schemes. Our method is designed to make implicit calculation of time steps more efficient. Our multi-scale method takes advantage of the multi-resolution decomposition directly, making it possible to calculate a single implicit time step using multiple, smaller, systems.

Previously, in Section 6.6, we were working with a Burgers' equation problem

$$u_t = \nu\, u_{xx} + u\, u_x, \quad u(0,t) = u(10,t) = 0, \quad u(x,0) = f(x), \quad \nu = 0.01, \qquad (7.1.1)$$

$f(x)$ given as the $t = 0$ plot in Figure 21, and explained in Section 6.6 after Equation (6.6.1). The results for the model with two oppositely signed waves hitting each other became rather steep. The results were good using a resolution of V_{-6} (Figure 21),

flawed but stable for a resolution of V_{-5} (Figure 22) and unstable for V_{-4} (Figure 23). It was stated at the end of Section 6.6 that a useful means to express the resulting system would be to include greater resolution around $x = 5$, where the two curves collide. Wavelet based systems are ideal for localized resolution. The simplest arrangement is to construct a system with unimportant coefficients removed. This would involve, for instance, using V_{-4} everywhere and only calculating W_{-4} and W_{-5} in the region between 4 and 6. The small-scale interactions are to be found near $x = 5$, so we would expect the solution's W_{-4} and W_{-5} related coefficients to have very small values outside $(4, 6)$. There are 159 V_{-4} coefficients over $\Omega = (0, 10)$. There are $2 \times (2^4 + 2^5) = 96$ W_{-4} and W_{-5} coefficients on $\Lambda = (4, 6)$. So, having V_{-6} only on $\Lambda = (4, 6)$ leads to a total of 255 coefficients. A full 2^6 resolution system, V_{-6} over all $\Omega = (0, 10)$, has 639 coefficients, and is unlikely to provide much in the way of increased accuracy.

A better approach is to take advantage of the wavelet/scaling function arrangement even further. One can, if certain requirements are taken into account, solve implicit time steps at multiple spatial resolutions. Most importantly, these resolutions are solved sequentially. This is possibly equivalent to doubling the size of the system at the price equivalent to doubling the number of time steps calculated. For non-linear systems, where even a linearized scheme (see Section 6.6 for an example) requires solving a new system every time step, this can be a serious advantage. While this arrangement is surprisingly easy to set up, the concept is difficult to explain. The key idea is to break the system into two components. First,

$$V_j = V_0 \oplus W_0 \oplus \cdots \oplus W_{j+1}, \tag{7.1.2}$$

which we will call the large-scale system, a coarse resolution system that covers the whole domain Ω. Next we have a version of

$$V_{j+m} = V_j \oplus W_j \oplus \cdots \oplus W_{j+m+1}, \tag{7.1.3}$$

only covering the subset $\Lambda \subset \Omega$, which we will call the small-scale system. Recall that $j, m < 0$. The small-scale system uses, as its lowest level of resolution, a localized V_j. This arrangement allows the two systems to be solved sequentially. First, the

new values of the large-scale system, V_j over Ω, are solved, and some are converted to be writable in terms of the small-scale system. Those values are included in the calculations for the new values of the small-scale system. Next, a few of the small scale system's values are converted and added to the large scale system, and the time step is done. The two systems are solved separately.

Depending how everything is set up, the two systems can have exactly the same size. In fact, this arrangement can be done with any number of systems. As long as the small-scale systems are disjoint, they can be solved separately. That the different systems can be solved separately is the advantage here. A pair of systems sized $N \times N$ are almost always quicker to solve than a single system sized $2N \times 2N$, though the effect is much greater for large N. See Table 4, Section 7.5, for the relevant numerical results from Burgers' equation.

7.2 Implementation

To explain in detail requires a certain level of notation. We will look at a one dimensional example, though the concept can be expanded to any number of dimensions. As long as the functions in question are decomposed via wavelets in a particular dimension, then this concept can be applied in that dimension.

First, our biorthogonal decomposition. The actual functions we will be dealing with are those of the form

$$f = \sum_k \phi_{0,k}(x) + \sum_j \sum_k b_{j,k}\psi_{j,k}(x). \tag{7.2.1}$$

Recall that wavelet decompositions are in spaces

$$V_j = \text{Span}\left\{2^{-j/2}\,\phi\left(2^{-j}x - k\right), k \in \mathbb{Z}\right\} \tag{7.2.2}$$

and

$$W_j = \text{Span}\left\{2^{-j/2}\,\psi\left(2^{-j}x - k\right), k \in \mathbb{Z}\right\}. \tag{7.2.3}$$

The setup for orthogonal wavelets has $W_j = V_{j-1} \bigcap V_j^{\perp}$, so

$$V_j = V_0 \oplus W_0 \oplus \cdots \oplus W_{j+1} = \bigoplus_{m>j} W_m. \tag{7.2.4}$$

The main point here is that there are multiple ways of writing any given function in V_j, all fully equivalent. Since the specific form of the decomposition is relevant to any wavelet related method, a means of expressing these differences becomes helpful.

All functions in the V_j and W_j spaces can be characterized in terms of their wavelet and scaling function related coefficients. The wavelet (ψ) coefficients are written $b_{j,k}$, the scaling function (ϕ) coefficients are written $a_{j,k}$. As usual, the j relates to the resolution, k relates to location. The $b_{j,k}$ are best organized into vectors written $\mathbf{b}_j$. The same can be done with the $a_{0,k}$, creating a vector $\mathbf{a}_0$. Any decomposition of $f \in V_j$ can be fully characterized by the vector $[\mathbf{b}_{j+1}, \mathbf{b}_{j+2}, \ldots, \mathbf{b}_0, \mathbf{a}_0]$. Any function $f \in V_j$ can also be characterized by the, equally sized, vector $\mathbf{a}_j$, and so forth. The setup discussed around Equation (7.1.3) focused on two different levels of resolution, V_j and V_{j+m}, with $j, m < 0$. One helpful way of writing this is to have the vector $\mathbf{a}$, with no subscript, contain all necessary coefficients for V_j and have $\mathbf{b}$ contain all the necessary coefficients for W_j to W_{j+m+1}. This means that any function in V_j can be expressed as a vector of the form $\begin{bmatrix} \mathbf{0} \\ \mathbf{a} \end{bmatrix}$, and any function in V_{j+m} can be expressed as a vector $\begin{bmatrix} \mathbf{b} \\ \mathbf{a} \end{bmatrix}$.

Next, we have two different domains to use, Ω and $\Lambda \subset \Omega$. The vector $\mathbf{a}$ will have all the coefficients from the V_j space over the entire domain Ω. Putting Λ with it, for $\mathbf{a}_\Lambda$, restricts the coefficients to those in the smaller domain Λ. This will be standard practice. If a subscript Λ is put under a vector or matrix, that represents a version of the vector or matrix with coefficients relating only to the smaller domain Λ.

Example 7.2.1 *Conversion of the vector $\boldsymbol{a}$ into its Λ equivalent, $\boldsymbol{a}_\Lambda$, with $V_j = V_{-3}$, $\Omega = (0, 10)$, $\Lambda = (3.75, 6.25)$, and the biorthogonal scaling function from Figure 7.*

The vector $\mathbf{a}$, if written in the V_{-3} form (only with $\phi_{-3,k}$ terms), is

$$\mathbf{a} = \left[a_1, a_2, a_3, \ldots, a_{78}, a_{79}\right]^T,$$

representing a function in V_{-3} on the domain $\Omega = (0, 10)$. The number of coefficients relates to the resolution, the size of the domain, and to the boundary conditions (as

well as the shape of our chosen ϕ). The biorthogonal scaling functions $\phi_{j,k}$ we are using are symmetric around integer multiples of 2^j. As a result, the coefficient a_k in $\mathbf{a}$ will correspond to a $\phi_{j,k}(x)$, centered at the point $x = \frac{k}{8}$. Our zero boundary conditions mean that the coefficients for the $\phi_{j,k}$ centered at $x = 0$ and $x = 10$ will be fixed at zero (see Section 6.2), so the first a_k is a_1 (with a function centered at $x = \frac{1}{8}$) and the last is a_{79} (with a function centered at $x = \frac{79}{8} = 10 - \frac{1}{8}$).

The vector $\mathbf{a}_\Lambda$ has fewer components than $\mathbf{a}$, only those corresponding to functions $\phi_{j,k}$ that are centered in $\Lambda = (3.75, 6.25)$. This results in

$$\mathbf{a}_\Lambda = \left[a_{31}, a_{32}, a_{33}, \ldots, a_{47}, a_{48}, a_{49}\right]^T.$$

The function $\phi_{-3,30}$ is centered at 3.75 exactly, so we do not include a_{30}, the same is true of $\phi_{-3,50}$ and a_{50}. The vector $\mathbf{a}$ has 79 components, the vector $\mathbf{a}_\Lambda$ has 19, approximately a quarter as many. This is appropriate, since Λ is a quarter the size of Ω.

The vector $\mathbf{b}_\Lambda$ is more complicated to write. Recall that the vector $\mathbf{b}$ includes all the $b_{j,k}$ coefficients used to extend the coarse resolution to the fine resolution. As a result, there are usually multiple W_j spaces used, and the procedure in Example 7.2.1 has to be used on each $\mathbf{b}_j$ vector. However, the relationship between $\mathbf{b}_\Lambda$ and $\mathbf{b}$ is the same as between $\mathbf{a}_\Lambda$ and $\mathbf{a}$. The component $b_{j,k}$ in $\mathbf{b}$ will be in $\mathbf{b}_\Lambda$ if $\psi_{j+1,k}$ is centered in Λ (remember that the $\psi_{j,k}$ we are using are symmetric, see Figure 8). This will be true for all $b_{j-1,k}$, $b_{j-2,k}$ and so forth, up to $b_{j+m+1,k}$. So, the vector $\mathbf{b}_\Lambda$ has the components of $\mathbf{b}$ which are the coefficients related to functions $\psi_{j,k}$ that are centered within Λ. This means that $\begin{bmatrix} \mathbf{b}_\Lambda \\ \mathbf{a}_\Lambda \end{bmatrix}$ characterizes functions in V_{m+j} (the finest resolution) on Λ the same way $\begin{bmatrix} \mathbf{b} \\ \mathbf{a} \end{bmatrix}$ characterizes functions in V_{m+j} over Ω.

Now we take a linear partial differential equation of the form $u_t(x,t) = Lu(x,t)$, $x \in \Omega$. As L is linear, we approximate it using a matrix (see Section 5.3), which we will call M. If the function $u(x,t)$ is approximated by a V_{j+m} decomposition equivalent to $\begin{bmatrix} \mathbf{b} \\ \mathbf{a} \end{bmatrix}$, then $Lu(x,t)$ restricted to $V_{j,m}$ is approximately $M \begin{bmatrix} \mathbf{b} \\ \mathbf{a} \end{bmatrix}$. The matrix M, along with the second order implicit Adam's method, gives us the time

step

$$\begin{bmatrix} \mathbf{b}^{n+1} \\ \mathbf{a}^{n+1} \end{bmatrix} = \begin{bmatrix} \mathbf{b}^{n} \\ \mathbf{a}^{n} \end{bmatrix} + \frac{h}{2} M \left(\begin{bmatrix} \mathbf{b}^{n+1} \\ \mathbf{a}^{n+1} \end{bmatrix} + \begin{bmatrix} \mathbf{b}^{n} \\ \mathbf{a}^{n} \end{bmatrix} \right). \tag{7.2.5}$$

The elements of M will have to be broken up, since we will not be using all of M, into a block decomposition of the form

$$M = \begin{bmatrix} A & B \\ C & D \end{bmatrix}, \qquad M \begin{bmatrix} \mathbf{b} \\ \mathbf{a} \end{bmatrix} = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} \mathbf{b} \\ \mathbf{a} \end{bmatrix} = \begin{bmatrix} A\,\mathbf{b} + B\,\mathbf{a} \\ C\,\mathbf{b} + D\,\mathbf{a} \end{bmatrix}. \tag{7.2.6}$$

The matrices A and B will have the same number of rows as $\mathbf{b}$, C and D will have the same number of rows as $\mathbf{a}$. Also, A and C will have the same number of columns as $\mathbf{b}$ has rows and B and D will have the same number of columns as $\mathbf{a}$ has rows. Note that A and D are square matrices, but do not have to be equal in size. In fact, our main example in this chapter has A contain about four times as many rows as D. If this block decomposition looks familiar, it is because it was brought up briefly in Section 5.3.

Example 7.2.2 *Using the same domain and resolution as in Example 7.2.1, we will take a look at getting D_Λ from D.*

This is basically the same as converting $\mathbf{a}$ to $\mathbf{a}_\Lambda$, except we will have to look to multiple columns, as well as the rows.

The matrix D, which approximates the differential operator L for V_{-3}, is a square matrix. Write it as

$$D = \begin{bmatrix} d_{1,1} & d_{1,2} & \cdots & d_{1,78} & d_{1,79} \\ d_{2,1} & d_{2,2} & \cdots & d_{2,78} & d_{2,79} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ d_{78,1} & d_{78,2} & \cdots & d_{78,78} & d_{78,79} \\ d_{79,1} & d_{79,2} & \ldots & d_{79,78} & d_{79,79} \end{bmatrix}.$$

Multiplication by this matrix will take the $a_{-3,k}$ coefficients from a function $f \in V_{-3}$, arranged into a vector $\mathbf{a}$, and create values $D\mathbf{a}$, the V_{-3} approximation of Lf. As a result, its height and width both have to be equal to the number of $a_{-3,k}$ coefficients in Ω, 79 in this case.

The matrix D_Λ only needs to deal with functions in V_{-3} restricted to Λ. As a result, the height and width of D_Λ has to be equal to the number of $a_{-3,k}$ coefficients in Λ. As a result, it is much smaller:

$$D_\Lambda = \begin{bmatrix} d_{31,31} & d_{31,32} & \cdots & d_{31,48} & d_{31,49} \\ d_{32,31} & d_{32,32} & \cdots & d_{32,48} & d_{32,49} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ d_{48,31} & d_{48,32} & \cdots & d_{48,48} & d_{48,49} \\ d_{49,31} & d_{49,32} & \cdots & d_{49,48} & d_{49,49} \end{bmatrix}.$$

Now to divide the time step in Equation (7.2.5) into two systems. First, we have the large-scale system, with functions in the space V_j over all Ω, characterized by vectors of the form $\mathbf{a}$. We also have the small-scale system with functions in V_{j+m} over Λ, characterized by vectors of the form $\begin{bmatrix} \mathbf{b}_\Lambda \\ \mathbf{a}_\Lambda \end{bmatrix}$. Recall that $\mathbf{a}_\Lambda$ has only the components of $\mathbf{a}$ corresponding to the functions $\phi_{j,k}$ in Λ. The large-scale system uses the derivative matrix D. The small-scale system uses M_Λ, where

$$M_\Lambda = \begin{bmatrix} A_\Lambda & B_\Lambda \\ C_\Lambda & D_\Lambda \end{bmatrix}. \tag{7.2.7}$$

Again, A_Λ, B_Λ, C_Λ and D_Λ are composed of the elements from A, B, C and D that are related to Λ (like in Example 7.2.2).

The first step is to use whatever time stepping scheme you wish to find a temporary approximation of $\mathbf{a}^{n+1}$, which we will call $\mathbf{a}^{T_m}$. We will keep using the second order implicit Adam's method, making the large-scale system

$$\mathbf{a}^{T_m} = \mathbf{a}^n + \frac{h}{2} D\left(\mathbf{a}^n + \mathbf{a}^{T_m}\right). \tag{7.2.8}$$

We have the time step solved at the large-scale resolution V_j on all Ω. What we want now is to solve the system on Λ at the small scale resolution V_{j+m}. However, $\mathbf{a}^n$ and the newly calculated $\mathbf{a}^{T_m}$ have to be included. So, we take the components of $\mathbf{a}^n$ and $\mathbf{a}^{T_m}$ that are in Λ, $\mathbf{a}^n_\Lambda$ and $\mathbf{a}^{T_m}_\Lambda$. These are used in the system

$$\begin{bmatrix} \mathbf{b}^{n+1}_\Lambda \\ \mathbf{a}^{C_r}_\Lambda \end{bmatrix} = \begin{bmatrix} \mathbf{b}^n_\Lambda \\ \mathbf{0} \end{bmatrix} + \frac{h}{2} \begin{bmatrix} A_\Lambda & B_\Lambda \\ C_\Lambda & D_\Lambda \end{bmatrix} \begin{bmatrix} \mathbf{b}^n_\Lambda + \mathbf{b}^{n+1}_\Lambda \\ \mathbf{a}^{C_r}_\Lambda \end{bmatrix} + \frac{h}{2} \begin{bmatrix} B_\Lambda \\ O \end{bmatrix} \left(\mathbf{a}^{T_m}_\Lambda + \mathbf{a}^n_\Lambda\right). \tag{7.2.9}$$

Note the presence of $\mathbf{a}_\Lambda^{C_r}$, a corrector term for the temporary approximation $\mathbf{a}^{T_m}$. The final step is to take $\mathbf{a}_\Lambda^{C_r}$, rewrite it in Ω as $\mathbf{a}^{C_r}$, and add it to $\mathbf{a}^{T_m}$ for

$$\mathbf{a}^{n+1} = \mathbf{a}^{T_m} + \mathbf{a}^{C_r}. \tag{7.2.10}$$

This setup is consistent with the time stepping scheme. Since $\mathbf{a}_\Lambda^{n+1} = a_\Lambda^{T_m} + a_\Lambda^{C_r}$, the restriction of $\begin{bmatrix} \mathbf{b}^{n+1} \\ \mathbf{a}^{n+1} \end{bmatrix}$ to Λ will be

$$\begin{bmatrix} \mathbf{b}_\Lambda^{n+1} \\ \mathbf{a}_\Lambda^{C_r} + \mathbf{a}_\Lambda^{T_m} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_\Lambda^{n} \\ \mathbf{0} \end{bmatrix} + \frac{h}{2} \begin{bmatrix} A_\Lambda & B_\Lambda \\ C_\Lambda & D_\Lambda \end{bmatrix} \begin{bmatrix} \mathbf{b}_\Lambda^{n} + \mathbf{b}_\Lambda^{n+1} \\ \mathbf{a}_\Lambda^{C_r} \end{bmatrix} + \frac{h}{2} \begin{bmatrix} B_\Lambda \left(\mathbf{a}_\Lambda^{n} + \mathbf{a}_\Lambda^{T_m} \right) \\ \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \left(\mathbf{a}^n + \frac{h}{2} D \left(\mathbf{a}^n + \mathbf{a}^{T_m} \right) \right)\Big|_\Lambda \end{bmatrix}. \tag{7.2.11}$$

Next, getting that set of terms restricted to Λ written in our usual manner leaves

$$\begin{aligned} \begin{bmatrix} \mathbf{b}_\Lambda^{n+1} \\ \mathbf{a}_\Lambda^{C_r} + \mathbf{a}_\Lambda^{T_m} \end{bmatrix} &= \begin{bmatrix} \mathbf{b}_\Lambda^{n} \\ \mathbf{0} \end{bmatrix} + \frac{h}{2} \begin{bmatrix} A_\Lambda & B_\Lambda \\ C_\Lambda & D_\Lambda \end{bmatrix} \begin{bmatrix} \mathbf{b}_\Lambda^{n} + \mathbf{b}_\Lambda^{n+1} \\ \mathbf{a}_\Lambda^{C_r} \end{bmatrix} + \frac{h}{2} \begin{bmatrix} B_\Lambda \left(\mathbf{a}_\Lambda^{n} + \mathbf{a}_\Lambda^{T_m} \right) \\ \mathbf{0} \end{bmatrix} \\ &\quad + \begin{bmatrix} \mathbf{0} \\ \mathbf{a}_\Lambda^n + \frac{h}{2} D_\Lambda \left(\mathbf{a}_\Lambda^n + \mathbf{a}_\Lambda^{T_m} \right) \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{b}_\Lambda^{n} \\ \mathbf{a}_\Lambda^{n} \end{bmatrix} + \frac{h}{2} \begin{bmatrix} A_\Lambda & B_\Lambda \\ C_\Lambda & D_\Lambda \end{bmatrix} \begin{bmatrix} \mathbf{b}_\Lambda^{n} + \mathbf{b}_\Lambda^{n+1} \\ \mathbf{a}_\Lambda^{C_r} \end{bmatrix} + \frac{h}{2} \begin{bmatrix} B_\Lambda \\ D_\Lambda \end{bmatrix} \left(\mathbf{a}_\Lambda^{n} + \mathbf{a}_\Lambda^{T_m} \right) \\ \begin{bmatrix} \mathbf{b}_\Lambda^{n+1} \\ \mathbf{a}_\Lambda^{C_r} + \mathbf{a}_\Lambda^{T_m} \end{bmatrix} &= \begin{bmatrix} \mathbf{b}_\Lambda^{n} \\ \mathbf{a}_\Lambda^{n} \end{bmatrix} + \frac{h}{2} \begin{bmatrix} A_\Lambda & B_\Lambda \\ C_\Lambda & D_\Lambda \end{bmatrix} \begin{bmatrix} \mathbf{b}_\Lambda^{n} + \mathbf{b}_\Lambda^{n+1} \\ \mathbf{a}_\Lambda^{C_r} + \mathbf{a}_\Lambda^{T_m} + \mathbf{a}_\Lambda^{n} \end{bmatrix}. \end{aligned} \tag{7.2.12}$$

This time step matches the method used (2nd order Adam's Moulton).

Next, a note about 'boundary conditions' for the small-scale system. By this we mean the components of $\mathbf{a}$ near Λ that not included in the small-scale system but still have some influence on those elements within Λ. Experimentation has strongly suggested that this is not always a serious issue. If the method is appropriate to the problem, and if the Λ sub-domain is well chosen, then the interactions across the interfaces of Λ will be sufficiently accurately modeled by the coarse resolution terms.

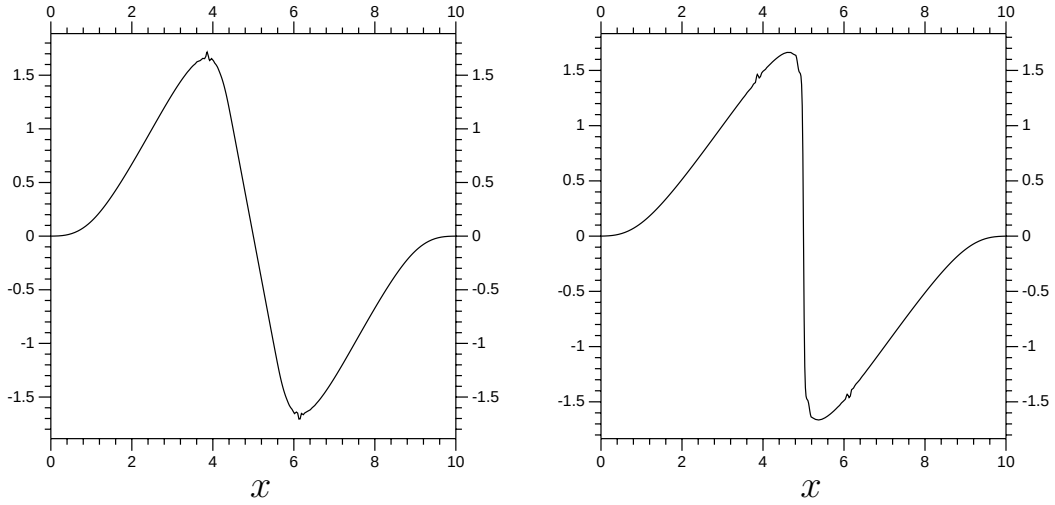
Recall that the large scale system is calculated over all Ω at the coarse resolution. Recall also that the coarse resolution derivative terms from the large-scale system, $D\mathbf{a}_n$ and $D\mathbf{a}_{T_m}$, are deliberately kept out of the small scale-system. As a result, the small-scale system does not include the coarse resolution terms, the only terms necessary to model the problem near the interfaces of Λ. The small-scale system will therefore not include any significant interactions near the interfaces of Λ. Recall that this only works when the problem is appropriate and Λ is well chosen.

7.3 Results

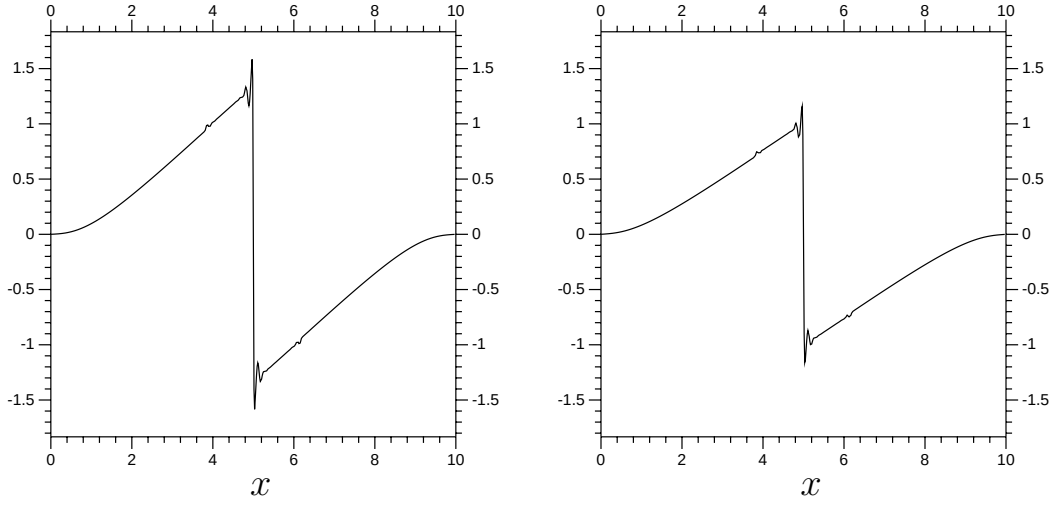
One statement needs to be made before we continue with the Burgers' equation example, and definitely before we introduce the more substantial Rossby problem. The two problems are intended as examples and tests of the method introduced in this chapter. We need only discuss the particular problems as much as is necessary to confirm that our results are reasonable. What we need to focus on is the difference between the calculations done at different resolutions and the advantages/disadvantages of the multi-scale method.

The main results we are interested in are those made using the multi-scale arrangement introduced in this chapter. So, these will involve a large-scale system on Ω and a small-scale system on Λ. There are two basic types of results we will discuss for comparison with the multi-scale results. The first are those calculated using only the large-scale system, the results you get without including the small-scale system. The second type are those calculated with the small-scale resolution over all Ω. These are the results we are attempting to duplicate using the multi-scale method. We will stick to double systems, almost always those where the large and small-scale systems have approximately the same size (number of coefficients).

Recall the results from a single, uniform, resolution of V_{-5} for Burgers' equation with viscosity $\nu = 0.01$ (Figure 22), and those from a resolution of V_{-4} (Figure 23). Those are our large-scale resolution results. Our small-scale resolution results use V_{-6}, and can be found in Figure 21. So, our multi-scale method will use V_{-6} for the small-scale system, and either V_{-4} or V_{-5} for the large-scale system. The full domain



(a) $u(x, 0.5)$ (b) $u(x, 1.0)$



(c) $u(x, 2.0)$ (d) $u(x, 3.0)$

Figure 24: Double system Burgers' equation results with V_{-4} on Ω and V_{-6} on $\Lambda = (3.75, 6.25)$. The function is called $u(x, t)$. The viscosity used is $\nu = 0.01$, the time step is $h = 0.01$. These results are calculated via the method outlined in Section 7.2 adapted for the linearized form of Burgers' equation (Section 6.6).

is $\Omega = (0, 10)$ (as before). The fine resolution terms will need to be concentrated near the shape at $x = 5$. It is helpful for both systems to have the same number of terms, so the cost of including the small-scale system is (approximately) equivalent to doubling the number of time steps. This requirement means that Λ should be equal to $(3.75, 6.25)$ when V_{-4} is the large-scale, and $\Lambda = (2.5, 7.5)$ when V_{-5} is the large-scale. The matrices used for approximating the derivatives are created based upon a collocation method (discussed around the end of Section 5.3). Results using

a localized V_{-6} on $\Lambda = (3.75, 6.25)$, with V_{-4} on Ω can be found in Figure 24. Notice that they are a marked improvement to the original V_{-4} results, being fairly stable, but are still very flawed. We will immediately start improving on them by introducing some additional features to the method.

7.4 Further Improvements

The method from Section 7.2 was, from the beginning, set up using a single M matrix, one created at V_{j+m} resolution and over all Ω. This is the best way to deal with linear elements from the L operator in $u_t = Lu$. There is no reason to use any other approach with linear terms from L. However, it is not a good option for dealing with any non-linear terms.

Recall that we use a linearization of the non-linear term in Burgers' equation,

$$u^{n+1}\frac{\partial}{\partial x}u^n + u^n\frac{\partial}{\partial x}u^{n+1} \quad \approx \quad u^{n+1}\frac{\partial}{\partial x}u^{n+1} + u^n\frac{\partial}{\partial x}u^n \tag{7.4.1}$$

(see Section 6.6). Using this approximation involves creating a linear operator that calculates the multiplication by $\left(u^n\frac{\partial}{\partial x} + \frac{\partial}{\partial x}u^n\right)$, then applying that operator to the values u^{n+1} in the time step. So the linearization requires creating a new, linear, operator at every time step. To do so at the V_{j+m} resolution over all Ω at every single time step would be time consuming. Instead, we will create an operator at V_j resolution on Ω, call it T_Ω, and an operator at V_{j+m} resolution on Λ, S_Λ. These operators are created separately. The first one, T_Ω, is used in the large-scale system

$$\mathbf{a}^{T_m} = \mathbf{a}^n + \frac{h}{2}\left[D\left(\mathbf{a}^{T_m} + \mathbf{a}^n\right) + T_\Omega\mathbf{a}^{T_m}\right], \tag{7.4.2}$$

note that we are keeping the linear terms, written the same as in Section (7.2).

The second operator, S_Λ, created independently of T_Ω, is used in the small-scale system

$$\begin{bmatrix} \mathbf{b}_\Lambda^{n+1} \\ \mathbf{a}_\Lambda^{C_r} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_\Lambda^{n} \\ \mathbf{0} \end{bmatrix} + \frac{h}{2}\left(\begin{bmatrix} A_\Lambda & B_\Lambda \\ C_\Lambda & D_\Lambda \end{bmatrix}\begin{bmatrix} \mathbf{b}_\Lambda^{n+1} + \mathbf{b}_\Lambda^{n} \\ \mathbf{a}_\Lambda^{C_r} \end{bmatrix} + \begin{bmatrix} B_\Lambda \\ O \end{bmatrix}\left(\mathbf{a}_\Lambda^{T_m} + \mathbf{a}_\Lambda^{n}\right)\right.$$
$$\left. + S_\Lambda\begin{bmatrix} \mathbf{b}_\Lambda^{n+1} \\ \mathbf{a}_\Lambda^{C_r} \end{bmatrix} + S_\Lambda\begin{bmatrix} \mathbf{0} \\ \mathbf{a}_\Lambda^{T_m} \end{bmatrix} - \begin{bmatrix} \mathbf{0} \\ \left(T_\Omega\mathbf{a}^{T_m}\right)_\Lambda \end{bmatrix}\right), \tag{7.4.3}$$

(note that O here represents a zero matrix and $\mathbf{0}$ is a zero vector). As usual, the last step is to get $\mathbf{a}^{n+1} = \mathbf{a}^{T_m} + \mathbf{a}^{C_r}$.

One set of terms from these systems really needs to be discussed in more detail:

$$\left(S_\Lambda \begin{bmatrix} \mathbf{0} \\ \mathbf{a}_\Lambda^{T_m} \end{bmatrix} - \begin{bmatrix} \mathbf{0} \\ \left(T_\Omega \mathbf{a}^{T_m}\right)_\Lambda \end{bmatrix} \right), \tag{7.4.4}$$

from (7.4.3). Notice that S_Λ is a matrix the same size as M_Λ (see (7.2.7)), so (7.4.4) will usually have components in the W_j to W_{j+m+1} spaces. These components are not included in Equation (7.4.2), since that calculation is restricted to the resolution V_j, but these terms have to be included in the small-scale system. In this way, (7.4.4) is the equivalent of the $\begin{bmatrix} B_\Lambda \\ O \end{bmatrix} \left(\mathbf{a}_\Lambda^{T_m} + \mathbf{a}_\Lambda^n\right)$ term in Equation (7.2.9), applied to the linearized components of the problem. Both involve vectors used, and calculated, in the large-scale system, and their influence on the small-scale system. The linear version looks simpler, but it actually is not. We could actually write $\begin{bmatrix} B_\Lambda \\ O \end{bmatrix} \left(\mathbf{a}_\Lambda^{T_m} + \mathbf{a}_\Lambda^n\right)$ in the form

$$\begin{bmatrix} B_\Lambda \\ D_\Lambda \end{bmatrix} \left(\mathbf{a}_\Lambda^{T_m} + \mathbf{a}_\Lambda^n\right) - \begin{bmatrix} O \\ \left[D(\mathbf{a}^{T_m} + \mathbf{a}^n)\right]_\Lambda \end{bmatrix}, \tag{7.4.5}$$

which looks much more like (7.4.4). The term on the left in (7.4.5) is equal to $M_\Lambda \begin{bmatrix} \mathbf{0} \\ \mathbf{a}_\Lambda^{T_m} + \mathbf{a}_\Lambda^n \end{bmatrix}$, a normal part of the small-scale system. The second term in (7.4.5) is the D related term from (7.2.8), restricted to Λ. Since they have already been included in the calculations, they need to be subtracted here in order to avoid being included twice. Equation (7.2.9) uses a simplified form of (7.4.5). If D_Λ is composed of elements of D, then (7.4.5) is equal to

$$\begin{bmatrix} B_\Lambda \\ D_\Lambda \end{bmatrix} \left(\mathbf{a}_\Lambda^{T_m} + \mathbf{a}_\Lambda^n\right) - \begin{bmatrix} O \\ D_\Lambda(\mathbf{a}_\Lambda^{T_m} + \mathbf{a}_\Lambda^n) \end{bmatrix} = \begin{bmatrix} B_\Lambda \\ O \end{bmatrix} \left(\mathbf{a}_\Lambda^{T_m} + \mathbf{a}_\Lambda^n\right), \tag{7.4.6}$$

exactly what is used in (7.2.9). The reason it does not cancel quite so well in the non-linear case is that the S_Λ and T_Ω operators are created separately, so the restriction of T_Ω to Λ is unlikely to be completely equal to the large-scale related elements of S_Λ (i.e., S_Λ restricted to V_j). As a result, (7.4.4) is necessary for non-linear problems.

Note that the setup beginning around Equation (7.2.6) is not the only option for calculating the A to D matrices and A_Λ to D_Λ matrices, it is just the best option. One could, for instance, calculate the D matrix and the A_Λ to D_Λ matrices separately (this actually saves time, smaller domains and coarser resolutions make things easier). If E is this new matrix, which approximates the derivative at the coarse resolution over all Ω, then the the large-scale system becomes

$$\mathbf{a}^{T_m} = \mathbf{a}^n + \frac{h}{2} E \left(\mathbf{a}^n + \mathbf{a}^{T_m} \right). \tag{7.4.7}$$

Next, we get the small-scale system

$$\begin{bmatrix} \mathbf{b}_\Lambda^{n+1} \\ \mathbf{a}_\Lambda^{C_r} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_\Lambda^{n} \\ \mathbf{0} \end{bmatrix} + \frac{h}{2} \left(\begin{bmatrix} A_\Lambda & B_\Lambda \\ C_\Lambda & D_\Lambda \end{bmatrix} \begin{bmatrix} \mathbf{b}_\Lambda^{n+1} + \mathbf{b}_\Lambda^{n} \\ \mathbf{a}_\Lambda^{C_r} \end{bmatrix} + \begin{bmatrix} B_\Lambda \\ D_\Lambda - E_\Lambda \end{bmatrix} \left(\mathbf{a}_\Lambda^{T_m} + \mathbf{a}_\Lambda^{n} \right) \right), \tag{7.4.8}$$

following the pattern from (7.4.5). If we, instead, assume that $D_\Lambda - E = O$ then the small-scale system becomes

$$\begin{bmatrix} \mathbf{b}_\Lambda^{n+1} \\ \mathbf{a}_\Lambda^{C_r} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_\Lambda^{n} \\ \mathbf{0} \end{bmatrix} + \frac{h}{2} \left(\begin{bmatrix} A_\Lambda & B_\Lambda \\ C_\Lambda & D_\Lambda \end{bmatrix} \begin{bmatrix} \mathbf{b}_\Lambda^{n+1} + \mathbf{b}_\Lambda^{n} \\ \mathbf{a}_\Lambda^{C_r} \end{bmatrix} + \begin{bmatrix} B_\Lambda \\ O \end{bmatrix} \left(\mathbf{a}_\Lambda^{T_m} + \mathbf{a}_\Lambda^{n} \right) \right). \tag{7.4.9}$$

Using (7.4.8) results in errors around the boundaries of Λ, and when it is applied to linear problems (or components of problems) the errors can manifest as instability. This is particularly common, and galling, when caused by viscosity terms. Using (7.4.9) is another option, which clears up the error (and instability) at the boundary. The downside to (7.4.9) is persistent error in the center of Λ. The solution is to use both approaches simultaneously. We need (7.4.8) at the boundary and (7.4.9) in the center, and it is surprisingly easy to do just that. What we currently use is a diagonal matrix, call it J, to convert smoothly between the two vectors

$$\begin{bmatrix} B_\Lambda \\ D_\Lambda - E_\Lambda \end{bmatrix} \left(\mathbf{a}_\Lambda^{T_m} + \mathbf{a}_\Lambda^{n} \right), \quad \text{and} \quad \begin{bmatrix} B_\Lambda \\ O \end{bmatrix} \left(\mathbf{a}_\Lambda^{T_m} + \mathbf{a}_\Lambda^{n} \right). \tag{7.4.10}$$

What is needed is J to be equal to an identity matrix in the center, with zeros at the corners. So, a diagonal matrix J with diagonal values

$$J_{i,i} = 0, \frac{1}{3}, \frac{2}{3}, 1, 1, \ldots, 1, 1, \frac{2}{3}, \frac{1}{3}, 0, \tag{7.4.11}$$

or something similar. In place of either terms in (7.4.10) we use

$$\begin{bmatrix} I & O \\ O & J \end{bmatrix} \left(\begin{bmatrix} B_\Lambda \\ D_\Lambda - E_\Lambda \end{bmatrix} \left(\mathbf{a}_\Lambda^{T_m} + \mathbf{a}_\Lambda^n \right) \right) = \begin{bmatrix} B_\Lambda \\ J(D_\Lambda - E_\Lambda) \end{bmatrix} \left(\mathbf{a}^{T_m} + \mathbf{a}_\Lambda^n \right). \quad (7.4.12)$$

Consider the $\mathbf{a}$ (as in, V_j resolution) components from (7.4.12). The components nearest the top and bottom (those nearest the interface of Λ) will be equal to zero. Those in the center will be equal to $(D_\Lambda - E_\Lambda)\left(\mathbf{a}^{T_m} + \mathbf{a}_\Lambda^n\right)$. There is also a somewhat smooth transition between the central and outer components. So, we get the small-scale system

$$\begin{bmatrix} \mathbf{b}_\Lambda^{n+1} \\ \mathbf{a}_\Lambda^{C_r} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_\Lambda^n \\ \mathbf{0} \end{bmatrix} + \frac{h}{2} \left(\begin{bmatrix} A_\Lambda & B_\Lambda \\ C_\Lambda & D_\Lambda \end{bmatrix} \begin{bmatrix} \mathbf{b}_\Lambda^{n+1} + \mathbf{b}_\Lambda^n \\ \mathbf{a}_\Lambda^{C_r} \end{bmatrix} + \begin{bmatrix} B_\Lambda \\ J(D_\Lambda - E_\Lambda) \end{bmatrix} \left(\mathbf{a}_\Lambda^{T_m} + \mathbf{a}_\Lambda^n \right) \right), \quad (7.4.13)$$

which uses (7.4.8) in the center of Λ, for accuracy, and (7.4.9) near the interface of Λ, for stability.

This will primarily be applied to nonlinear terms, using a similar diagonal matrix J_{NL}. The related term becomes

$$\begin{bmatrix} I & O \\ O & J_{NL} \end{bmatrix} \left(S_\Lambda \begin{bmatrix} \mathbf{0} \\ \mathbf{a}_\Lambda^{T_m} \end{bmatrix} - \begin{bmatrix} 0 \\ \left(T_\Omega \mathbf{a}^{T_m}\right)\big|_\Lambda \end{bmatrix} \right), \quad (7.4.14)$$

(recall that the S_Λ matrix is the same size as M_Λ). The full small-scale system is

$$\begin{bmatrix} \mathbf{b}_\Lambda^{n+1} \\ \mathbf{a}_\Lambda^{C_r} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_\Lambda^n \\ \mathbf{0} \end{bmatrix} + \frac{h}{2} \left(\begin{bmatrix} A_\Lambda & B_\Lambda \\ C_\Lambda & D_\Lambda \end{bmatrix} \begin{bmatrix} \mathbf{b}_\Lambda^{n+1} + \mathbf{b}_\Lambda^n \\ \mathbf{a}_\Lambda^{C_r} \end{bmatrix} + \begin{bmatrix} B_\Lambda \\ O \end{bmatrix} \left(\mathbf{a}_\Lambda^{T_m} + \mathbf{a}_\Lambda^n \right) + S_\Lambda \begin{bmatrix} \mathbf{b}_\Lambda^{n+1} \\ \mathbf{a}_\Lambda^{C_r} \end{bmatrix} + \begin{bmatrix} I & O \\ O & J_{NL} \end{bmatrix} \left(S_\Lambda \begin{bmatrix} \mathbf{0} \\ \mathbf{a}_\Lambda^{T_m} \end{bmatrix} - \begin{bmatrix} \mathbf{0} \\ \left(T_\Omega \mathbf{a}^{T_m}\right)_\Lambda \end{bmatrix} \right) \right), \quad (7.4.15)$$

assuming there is no reason to use a J matrix on the linear terms from the problem.

Our next modification to the systems is more complicated to implement than a matrix multiplication. Recall that our large-scale system uses the V_j space on Ω and the small-scale system is in V_{j+m} on the smaller domain Λ. By necessity, a lot of our focus is on the section they have in common, a set coefficients of functions

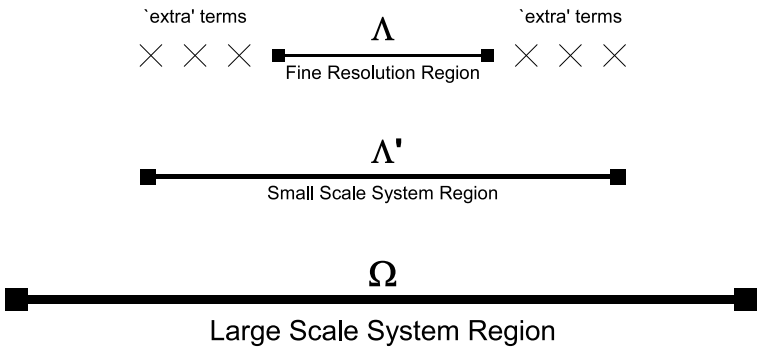


Figure 25: A simple diagram showing Ω, Λ, Λ' and the location of the 'extra' terms.

that are within both the large-scale system's limited resolution and the small-scale system's limited domain (so V_j functions within Λ). These values are where the bulk of the interactions between the two systems occur, and getting accurate interactions between the large and small-scale systems is key to getting accurate results with the method. This particular modification is about expanding that region of interaction, increasing the number of coefficients $a_{j,k}$ included in the small-scale system. These extra coefficients (and their associated functions $\phi_{j,k}$) will be treated exactly the same as those in Λ. However, these extra terms (we will frequently refer to them as simply 'extra' coefficients $a_{j,k}$ or 'extra' $\phi_{j,k}$) will not actually be in Λ itself. We will not include any extra $b_{j,k}$ to $b_{j+m+1,k}$, keeping the small-scale resolution V_{j+m} restricted to the domain Λ. As a result, the increase in the size of the small-scale system (in terms of number of coefficients) will be fairly modest. This new feature creates a new domain Λ', with $\Lambda \subset \Lambda' \subset \Omega$. The small-scale system will calculate V_j resolution coefficients within Λ', but will keep all V_{j+m} resolution within Λ itself. See Figure 25.

Example 7.4.1 *Including three 'extra' terms on each side of* Λ*, when we use* V_{-3} *on* $\Omega = (0, 10)$*,* V_{-5} *on* $\Lambda = (3.75, 6.25)$ *multi-scale system from before.*

Having $\Lambda = (3.75, 6.25)$ means that the small-scale system uses coefficients $a_{-3,k}$, starting at $k = 31$ and ending at $k = 49$. Those coefficients $a_{-3,k}$ correspond to functions $\phi_{-3,k}$ centered at 3.875 and 6.125, respectively. The three 'extra' $\phi_{-3,k}$ that will be included on the left are the three that are closest to the boundary without being inside it. They are $\phi_{-3,30}$, centered at $x = 3.75$; $\phi_{-3,29}$, centered at $x = 3.625$; and $\phi_{-3,28}$, centered at $x = 3.5$. The three 'extra' $\phi_{-3,k}$ that will be included on

the right are $\phi_{-3,50}$, centered at $x = 6.25$; $\phi_{-3,51}$, centered at $x = 6.375$; and $\phi_{-3,51}$, centered at 6.5. The basic setup, without the 'extra' terms, has a small-scale system size of 79 (as in, there are 79 coefficients calculated in the small-scale system). The setup including the 'extra' terms has 85. The new domain Λ' is $(3.375, 6.625)$, the boundaries moved three $\frac{1}{8} = 2^{-3}$ sized translations further apart than those of Λ. In V_{-5}, the resolution of the small-scale system, Λ' has $2^5(6.625 - 3.375) = 104$ coefficients.

Including the 'extra' terms in the small-scale system's time step calculation, (7.2.9), results in some changes to the vectors and matrices in the equation, but no changes to how the equation itself written. The inclusion of more $a_{j,k}$ terms in the small-scale system results in larger vectors $\mathbf{a}$, and a larger matrix D_Λ. The matrix C_Λ will become larger as well, namely taller (additional rows on top and bottom). The matrix B_Λ will have extra columns on the left and right. The matrix A_Λ will be unchanged.

Expanding the number of coefficients $a_{j,k}$ has a cost. First, there are those six extra terms to solve in the time step calculation. Also, calculating the operator approximating matrices, like S_Λ, is, now best done using the Λ' domain. This would require including all the functions ψ within Λ'. Thankfully, in our current, linearized, scheme (see Section 6.6), those terms are not included in the actual solving of the values for the time step, which is the most time consuming phase. They only come into play when setting up the operators themselves.

The best results use linear components derived from a full M matrix (see Section 7.2). Doing so results in the $D_\Lambda - (E)|_\Lambda$ term from (7.4.8) being equal to a zero matrix, so no need for a J type multiplier for the linear components of L. A J_{NL} multiplier is, however, required for the non-linear derivative terms (again, see Section 7.2). Our multi-scale results use the standard V_{-4} on $\Omega = (0, 10)$ and V_{-6} on $\Lambda = (3.75, 6.25)$. They also use 3 'extra' terms and

$$(J_{NL})_{i,i} = 0\,,\ 0\,,\ 0\,,\ 0\,,\ \frac{1}{2}\,,\ 1\,,1\,,\ \ldots\ ,1\,,1\,,\frac{1}{2},0\,,\ 0\,,\ 0\,,\ 0,$$

meaning that the non-linear effect on the 'extra' terms in $\mathbf{a}_\Lambda^{T_m}$ is removed entirely. The results are in Figure 26. They show little difference from the full resolution V_{-6}

results. Remember that these results use half as many terms, total, and the method solves them in approximately half sized sections. As a result, the average time step for the V_{-6} system takes about 0.988 seconds, while the average time step for the V_{-4}/V_{-6} multi-scale system takes about 0.0337 seconds. So, the multi-scale system gives nearly identical results and takes about one 29th as long.

Coding can be found in Example C.5.2.

7.5 Error, Convergence and Stability

Now we take a closer look at the main sources of error. Recall, from (7.2.5), that the full resolution, full domain, time stepping scheme for a linear problem is

$$\begin{bmatrix} \mathbf{b}^{n+1} \\ \mathbf{a}^{n+1} \end{bmatrix} = \begin{bmatrix} \mathbf{b}^{n} \\ \mathbf{a}^{n} \end{bmatrix} + \frac{h}{2} \begin{bmatrix} A & B \\ C & D \end{bmatrix} \left(\begin{bmatrix} \mathbf{b}^{n+1} \\ \mathbf{a}^{n+1} \end{bmatrix} + \begin{bmatrix} \mathbf{b}^{n} \\ \mathbf{a}^{n} \end{bmatrix} \right). \tag{7.5.1}$$

The multi-scale method makes use of the components of the vectors $\mathbf{a}$ and $\mathbf{b}$, as well as the elements of the matrices A to D, that relate to the sub-domain Λ. Now we will have to discuss the sub-domain $\Omega \bigcap \Lambda^C$, and related vectors/matrices. We will now subdivide the vectors like so:

$$\mathbf{a} = \begin{bmatrix} \mathbf{a}_1 \\ \mathbf{a}_\Lambda \end{bmatrix} \quad \text{and} \quad \mathbf{b} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_\Lambda \end{bmatrix}. \tag{7.5.2}$$

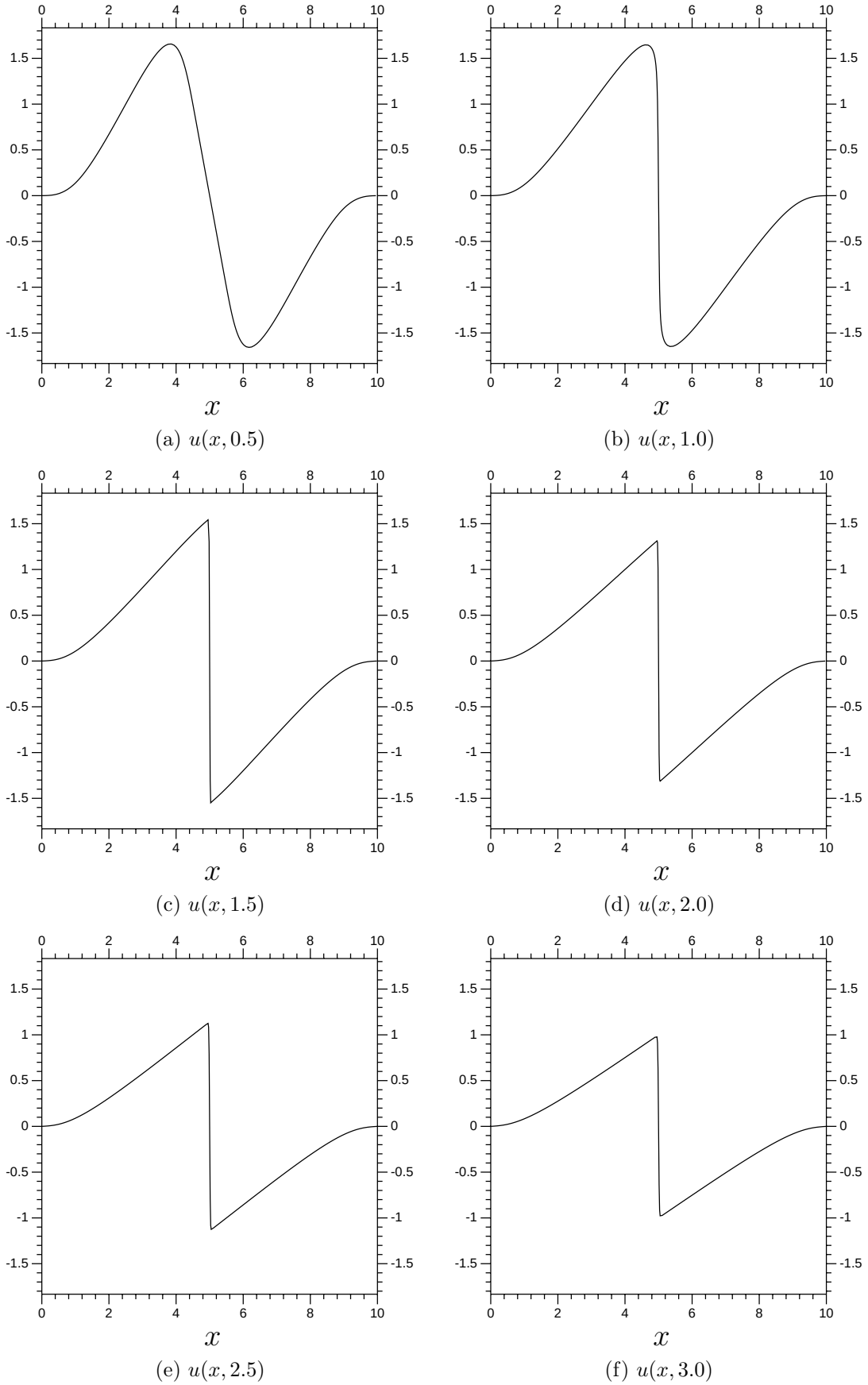
The related matrix block decomposition is

$$A = \begin{bmatrix} A_1 & A_2 \\ A_3 & A_\Lambda \end{bmatrix}, \quad B = \begin{bmatrix} B_1 & B_2 \\ B_3 & B_\Lambda \end{bmatrix}, \tag{7.5.3}$$

and the same for C and D. The idea is to have

$$A\mathbf{b} = \begin{bmatrix} A_1 & A_2 \\ A_3 & A_\Lambda \end{bmatrix} \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_\Lambda \end{bmatrix} = \begin{bmatrix} A_1\mathbf{b}_1 + A_2\mathbf{b}_\Lambda \\ A_3\mathbf{b}_1 + A_\Lambda\mathbf{b}_\Lambda \end{bmatrix}, \tag{7.5.4}$$

$$D\mathbf{a} = \begin{bmatrix} D_1 & D_2 \\ D_3 & D_\Lambda \end{bmatrix} \begin{bmatrix} \mathbf{a}_1 \\ \mathbf{a}_\Lambda \end{bmatrix} = \begin{bmatrix} D_1\mathbf{a}_1 + D_2\mathbf{a}_\Lambda \\ D_3\mathbf{a}_1 + D_\Lambda\mathbf{a}_\Lambda \end{bmatrix}, \tag{7.5.5}$$

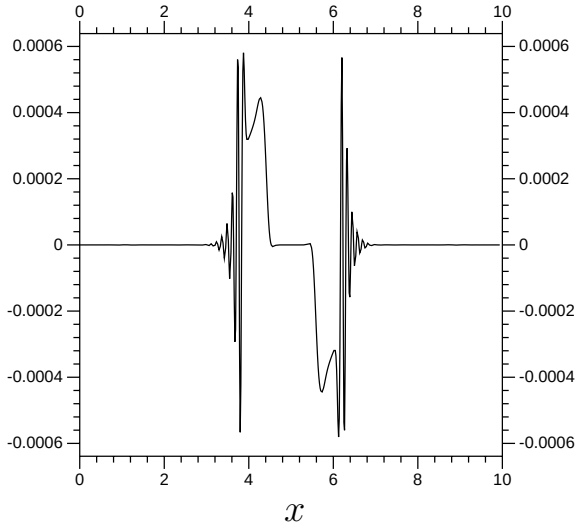


(a) $u(x, 0.5)$ (b) $u(x, 1.0)$

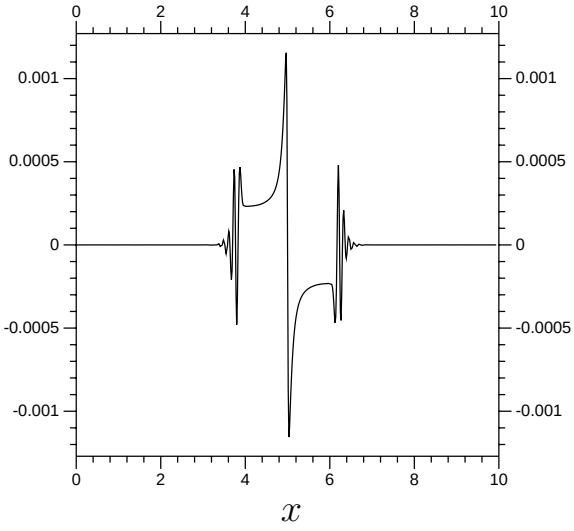
(c) $u(x, 1.5)$ (d) $u(x, 2.0)$

(e) $u(x, 2.5)$ (f) $u(x, 3.0)$

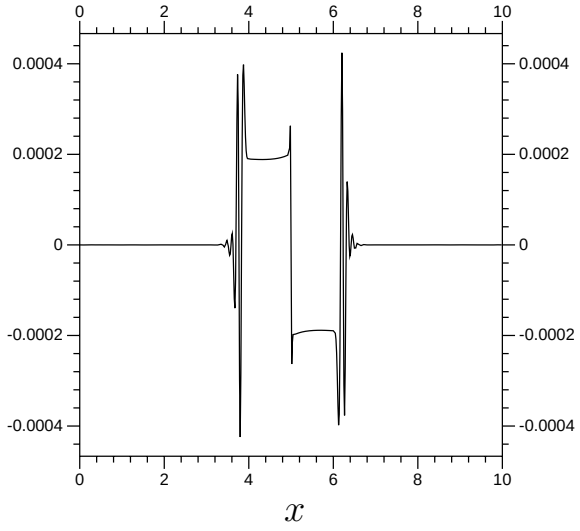
Figure 26: Burgers' equation results using the new components in Section 7.4. This one uses the same setup as Figure 24, but with three 'extra' coefficients $a_{j,k}$ around Λ. It also uses $(J_{NL})_{i,i}$ values of 0, 0, 0, 0 $\frac{1}{2}$, 1, 1 etc. The same time step of 0.01 was used.



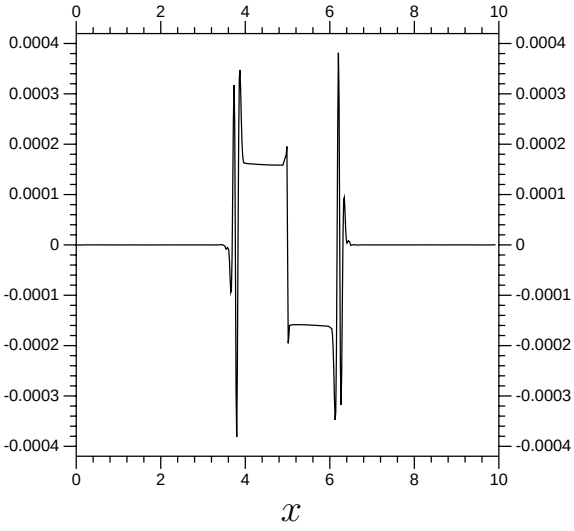
(a) 'Error' at $t = 0.5$



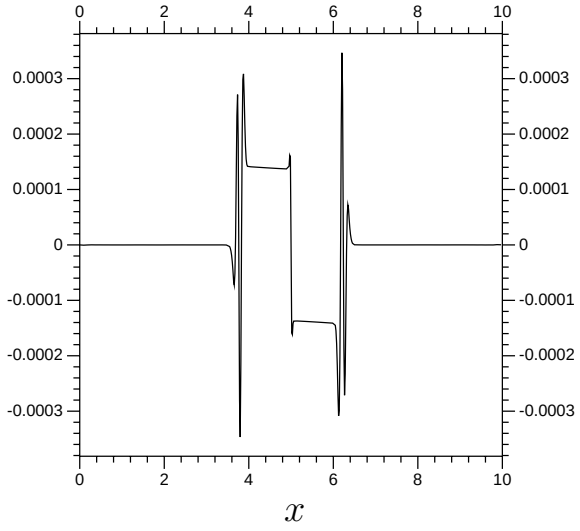
(b) 'Error' at $t = 1.0$



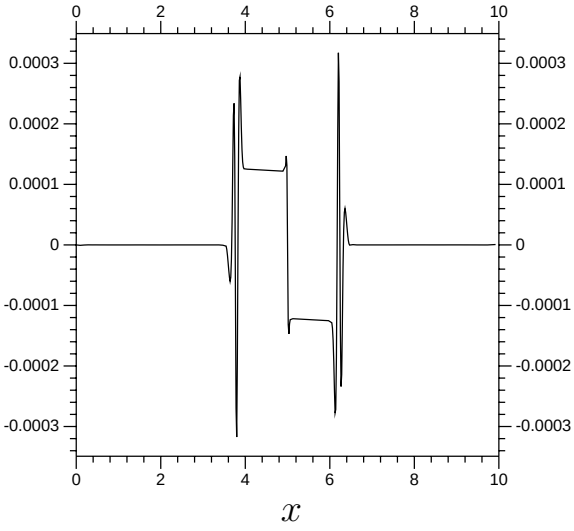
(c) 'Error' at $t = 1.5$



(d) 'Error' at $t = 2.0$



(e) 'Error' at $t = 2.5$



(f) 'Error' at $t = 3.0$

Figure 27: Burgers' equation multi-scale system results compared with a full resolution system. Plots are of the difference between the V_{-4}/V_{-6} results from Figure 26 and the full V_{-6} resolution results from Figure 21.

and so on. Written this way, the time step system becomes

$$\begin{bmatrix} \mathbf{b}_1^{n+1} \\ \mathbf{b}_\Lambda^{n+1} \\ \mathbf{a}_1^{n+1} \\ \mathbf{a}_\Lambda^{n+1} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1^{n} \\ \mathbf{b}_\Lambda^{n} \\ \mathbf{a}_1^{n} \\ \mathbf{a}_\Lambda^{n} \end{bmatrix} + \frac{h}{2} \begin{bmatrix} A_1 & A_2 & B_1 & B_2 \\ A_3 & A_\Lambda & B_3 & B_\Lambda \\ C_1 & C_2 & D_1 & D_2 \\ C_3 & C_\Lambda & D_3 & D_\Lambda \end{bmatrix} \left(\begin{bmatrix} \mathbf{b}_1^{n+1} \\ \mathbf{b}_\Lambda^{n+1} \\ \mathbf{a}_1^{n+1} \\ \mathbf{a}_\Lambda^{n+1} \end{bmatrix} + \begin{bmatrix} \mathbf{b}_1^{n} \\ \mathbf{b}_\Lambda^{n} \\ \mathbf{a}_1^{n} \\ \mathbf{a}_\Lambda^{n} \end{bmatrix} \right). \quad (7.5.6)$$

Now to compare that to the multi-scale method. The large-scale step is

$$\begin{bmatrix} \mathbf{a}_1^{n+1} \\ \mathbf{a}_\Lambda^{T_m} \end{bmatrix} = \begin{bmatrix} \mathbf{a}_1^{n} \\ \mathbf{a}_\Lambda^{T-M} \end{bmatrix} + \frac{h}{2} \begin{bmatrix} D_1 & D_2 \\ D_3 & D_\Lambda \end{bmatrix} \left(\begin{bmatrix} \mathbf{a}_1^{n+1} \\ \mathbf{a}_\Lambda^{T_m} \end{bmatrix} + \begin{bmatrix} \mathbf{a}_1^{n} \\ \mathbf{a}_\Lambda^{n} \end{bmatrix} \right). \quad (7.5.7)$$

Next we look at the small-scale system,

$$\begin{bmatrix} \mathbf{b}_\Lambda^{n+1} \\ \mathbf{a}_\Lambda^{C_r} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_\Lambda^{n} \\ \mathbf{0} \end{bmatrix} + \frac{h}{2} \begin{bmatrix} A_\Lambda & B_\Lambda \\ C_\Lambda & D_\Lambda \end{bmatrix} \begin{bmatrix} \mathbf{b}_\Lambda^{n} + \mathbf{b}_\Lambda^{n+1} \\ \mathbf{a}_\Lambda^{C_r} \end{bmatrix} + \frac{h}{2} \begin{bmatrix} B_\Lambda \\ O \end{bmatrix} \left(\mathbf{a}_\Lambda^{T_m} + \mathbf{a}_\Lambda^{C_r} \right). \quad (7.5.8)$$

Combining those two steps results in the full time stepping scheme

$$\begin{bmatrix} \mathbf{b}_\Lambda^{n+1} \\ \mathbf{a}_1^{n+1} \\ \mathbf{a}_\Lambda^{n+1} \end{bmatrix} = \begin{bmatrix} \mathbf{b}_\Lambda^{n} \\ \mathbf{a}_1^{n} \\ \mathbf{a}_\Lambda^{n} \end{bmatrix} + \frac{h}{2} \left(\begin{bmatrix} A_\Lambda & O & B_\Lambda \\ O & D_1 & D_2 \\ C_\Lambda & D_3 & D_\Lambda \end{bmatrix} \begin{bmatrix} \mathbf{b}_\Lambda^{n+1} + \mathbf{b}_\Lambda^{n} \\ \mathbf{a}_1^{n+1} + \mathbf{a}_1^{n} \\ \mathbf{a}_\Lambda^{n+1} + \mathbf{a}_\Lambda^{n} \end{bmatrix} - \begin{bmatrix} O \\ D_2 \\ O \end{bmatrix} \mathbf{a}_\Lambda^{C_r} \right). \quad (7.5.9)$$

So, the error inherent in the multi-scale system comes from three distinct sources:

1. The vectors $\mathbf{b}_1$ not being included at all. This is inherent in the setup, it is assumed that the fine resolution is only necessary in the Λ region, so $\|\mathbf{b}_1\|$ will be kept below a given threshold. Managing this source of error is relatively simple. An adaptive scheme would require calculating some values $b_{j,k}$ outside of the large-scale system and outside Λ, making sure that they are still below a given threshold. The best candidates are probably the terms $b_{j,k}$ that are closest to Λ, but are not in Λ, for obvious reasons.

2. The matrices C_2 and B_3 from (7.5.6) being missing from (7.5.9). This source of error is distinct from the $\mathbf{b}_1$ related error above. This error does not stem from not including some particular coefficients, it is from missing the interactions

between coefficients we are actually using. This source of error is reduced by using what we have called 'extra' terms (see Section 7.4). The matrices A to D will be diagonal dominant, or close, depending on how centered are the ϕ and ψ. If we look at

$$C = \begin{bmatrix} C_1 & C_2 \\ C_3 & C_\Lambda \end{bmatrix}$$

then the largest elements in C are to be found on the diagonal, and so are (mostly) in C_1 and C_Λ. Adding 'extra' terms widens the square matrix C_Λ, increases the number of elements of C taken by C_Λ. It also reduces the number of elements in C_1 C_2 and C_3. The matrix C_2 will be shrunk by removing rows of elements from the bottom, effectively restricting C_2 to the top right corner of C. Since the elements in C are smaller the further you get from the diagonal, the result is a matrix C_2 that has both fewer elements, and smaller valued elements. As a result, we get a smaller value for $\|C_2\mathbf{b}_\Lambda\|$.

The best way to manage this source of error is to simply check the largest values of $C_2\mathbf{b}_\Lambda$ and $B_3\mathbf{a}_1$. One could simply multiply the bottom row of C_2 by $\mathbf{b}_\Lambda$, and the top row of B_3 by $\mathbf{a}_1$, and increase the number of 'extra' terms if and when these products had magnitudes over a given tolerance.

3. The last type of error is from the last vector term in (7.5.9). This source of error is best managed via the 'extra' terms, as well. Including more 'extra' terms will result in D_2 having fewer elements, as well as smaller elements (like with C_2 and B_3 above). One other thing to notice is that $\mathbf{a}_\Lambda^{C_r}$, being a corrector term, is likely to be much smaller than any $\mathbf{a}_\Lambda^n$. We have

$$-\frac{h}{2}\, D_2\, \mathbf{a}_\Lambda^{C_r} = -\frac{h^2}{4}\, D_2 \left[C_\Lambda(\mathbf{b}_\Lambda^n + \mathbf{b}_\Lambda^{n+1}) + D_\Lambda \mathbf{a}_\Lambda^{C_r} \right],$$

so it is order h^2, and likely to be small if a reasonable time step is used.

So, if we increase the size of Λ and the number of 'extra' terms, we should expect all sources of error to decrease. To confirm this experimentally, several Burgers' equation multi-scale systems were calculated, all at V_{-5} with a localized V_{-6}. As Λ was increased in width, so was the number of 'extra' terms. The J multiplier was set

to have a zero for each 'extra' term, then one more zero followed by $\frac{1}{2}$, 1, and so on. The problem was kept steady, with $\nu = 0.01$, as was the size of the time step. All the results were compared to a control system with a fixed resolution of V_{-6}, with the same time step. One interesting result is that the accuracy in the interior of Λ, the accuracy near $x = 5$, becomes so good that it gets overshadowed by the error at the boundary. See Figure 28, and notice that the error on the boundary stays constant as Λ increases in size.

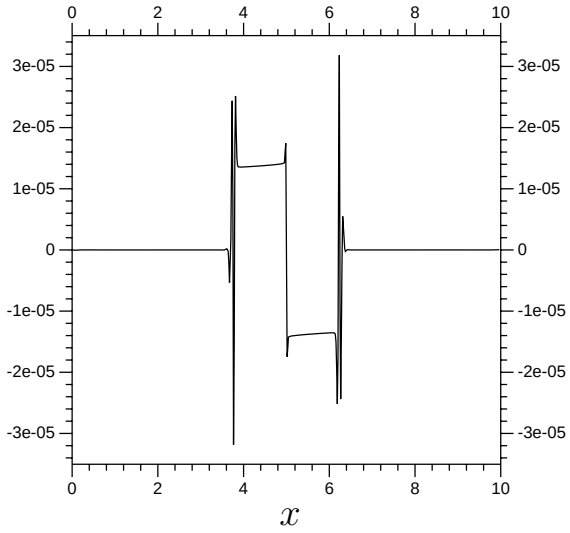
That error near $x = 0$ and $x = 10$ is due to lack of resolution around the boundary. That particular component of the error is constant, no matter how large or close we make Λ. Since we care about the accuracy in the center, around the Λ sub-domains, we will simply ignore $(0, 1.5)$ and $(8.5, 10)$ when calculating the error. In Figure 29 we have a set of plots of the numerically approximated errors of the different multi-scale systems, compared to the full domain V_{-6} system. These are calculated at $t = 2.0$, only on $(1.5, 8.5)$. Table 4 summarizes the results.

Table 4: Burgers' equation error for multi-scale systems.

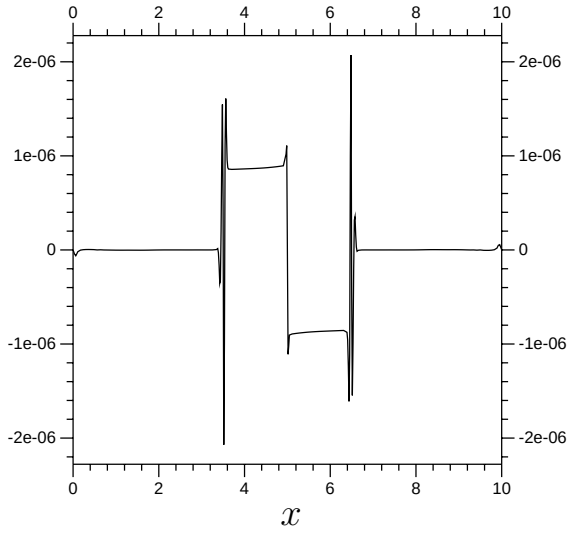
	Λ	'extra'	Average Time	L_2 Error	L_1 Error	L_∞ Error
V_{-5}/V_{-6}	[3.75,6.25]	4	0.0904	$2.4\,10^{-5}$	$4.6\,10^{-5}$	$3.7\,10^{-5}$
V_{-5}/V_{-6}	[3.50,6.50]	5	0.0998	$1.6\,10^{-6}$	$2.8\,10^{-6}$	$2.1\,10^{-6}$
V_{-5}/V_{-6}	[3.00,7.00]	6	0.1241	$1.0\,10^{-7}$	$2.1\,10^{-7}$	$1.2\,10^{-7}$
V_{-5}/V_{-6}	[2.50,7.50]	7	0.1560	$5.8\,10^{-9}$	$1.4\,10^{-8}$	$6.2\,10^{-9}$
V_{-5}/V_{-6}	[2.00,8.00]	7	0.1977	$3.7\,10^{-9}$	$8.6\,10^{-9}$	$4.1\,10^{-9}$
V_{-5}/V_{-6}	[1.50,8.50]	7	0.2541	$2.1\,10^{-9}$	$2.0\,10^{-9}$	$4.6\,10^{-9}$
V_{-6}	NA	NA	0.9888	0	0	0

Next we get to stability. The entire point of implicit methods is stability. If our method does not preserve the stability of the time stepping scheme used, then it is of limited value. However, actual theoretical confirmation of stability preservation is difficult, if not impossible. So, we will attempt to reduce any concerns about stability via some simple numerical experiments.

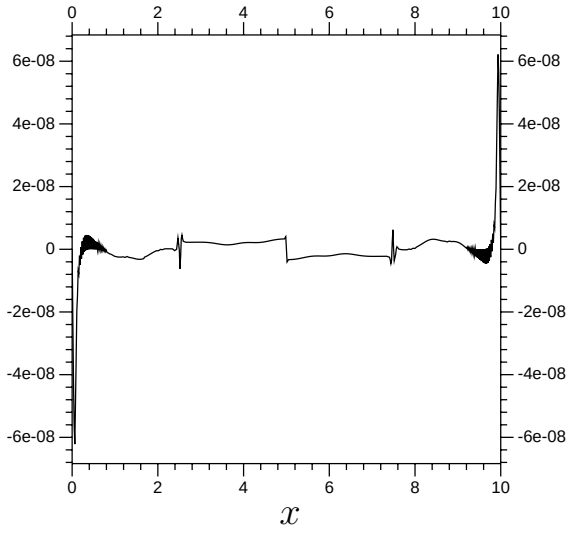
For a proper discussion of stability see [16] or any text on numerical methods for differential equations. Here is the basic idea for eigenvalue based stability. The system of linear differential equations $\frac{d}{dt}\mathbf{y}(t) = M\mathbf{y}(t)$, with $\mathbf{y}$ in $\mathbb{R}^n$, has the solution



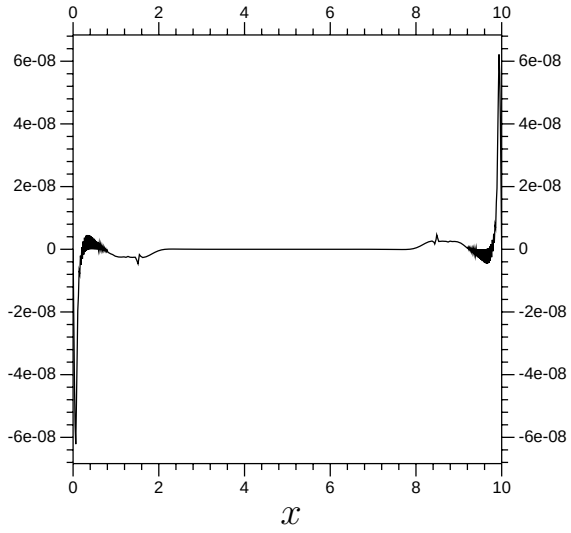
(a) $\Lambda = (3.75, 6.25)$ and 4 'extra' terms



(b) $\Lambda = (3.50, 6.50)$ and 4 'extra' terms



(c) $\Lambda = (2.50, 7.50)$ and 4 'extra' terms

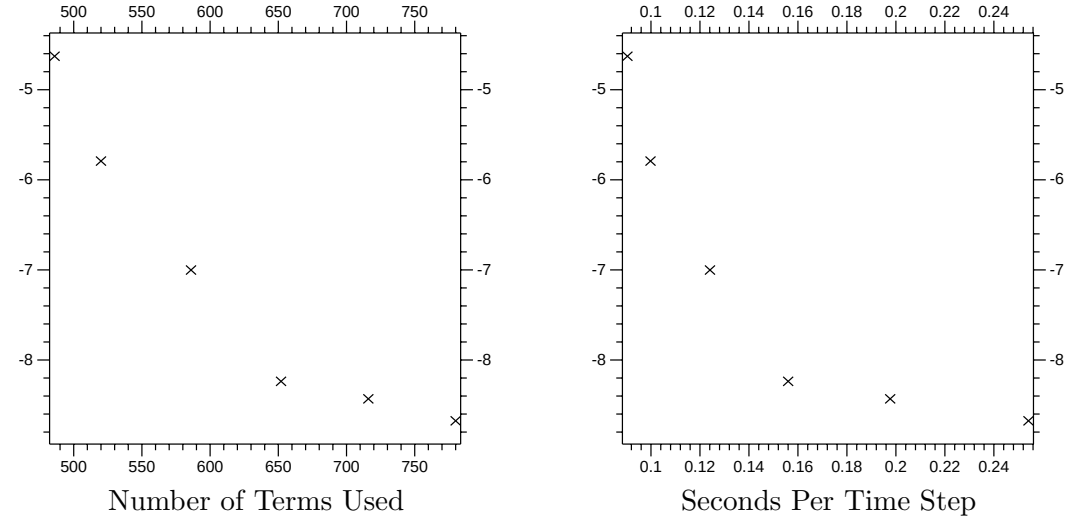


(d) $\Lambda = (1.50, 8.50)$ and 4 'extra' terms

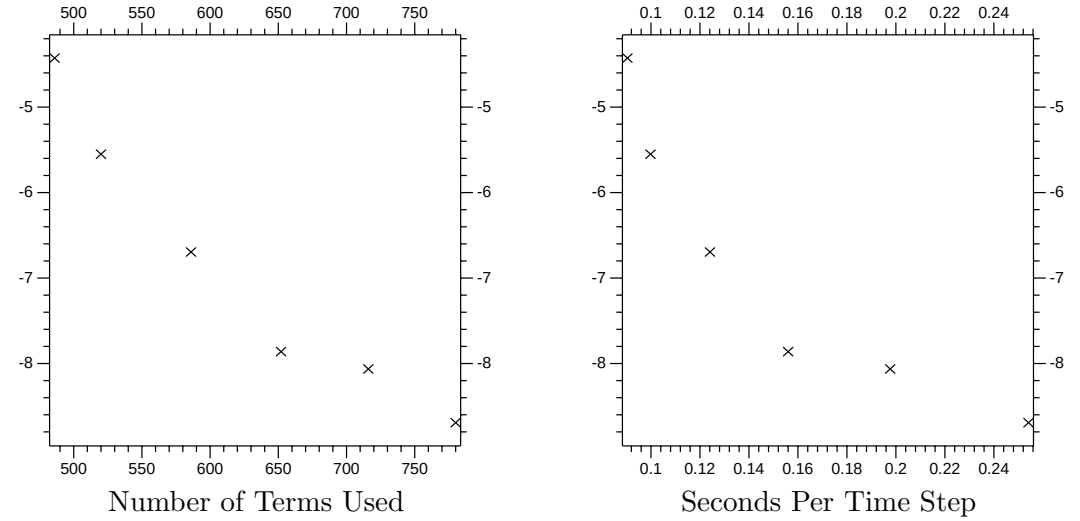
Figure 28: Error from results using V_{-5} with localized V_{-6}. The error is simply the difference between the multi-scale results and the V_{-6} control results, all calculated at $t = 2.0$. The multi-scale results differ primarily in their Λ sub-domains and in the number of 'extra' terms.

$\mathbf{y}(t) = e^M \mathbf{y}(0)$. The solution converges to zero for all initial conditions if and only if the eigenvalues of the matrix M have real elements smaller than zero. The time stepping scheme

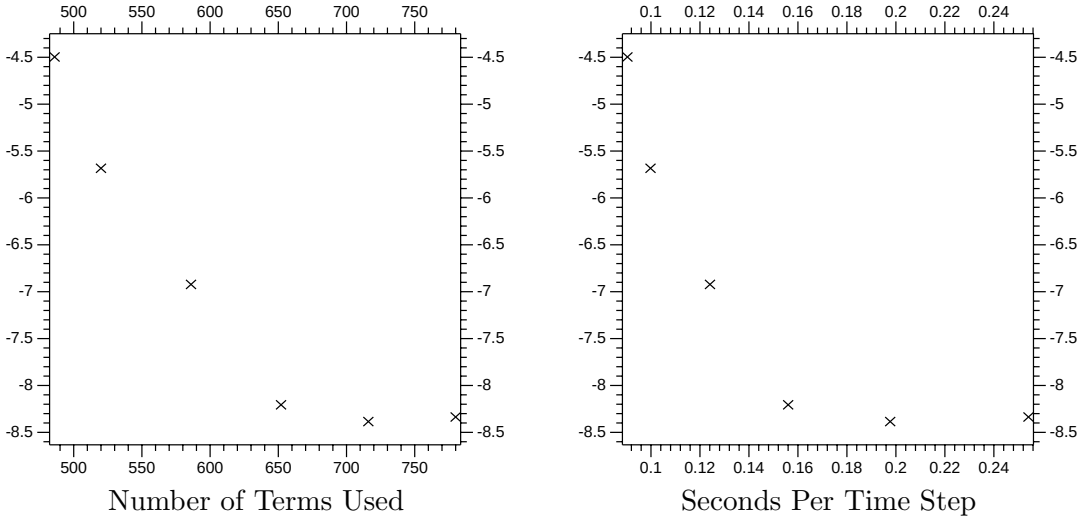
$$\mathbf{y}^{n+1} = \mathbf{y}^n + h \sum_{k=0}^{K} a_k \, M \, \mathbf{y}^{n-k}$$



(a) The $\log_{10}$ of the L^2 error.



(b) The $\log_{10}$ of the L^1 error.



(c) The $\log_{10}$ of the L^∞ error.

Figure 29: Error from multi-scale systems, compared to system size and computational time. The system size is the total number of coefficients $a_{j,k}$ and $b_{j,k}$ used the systems, all together. The computational time is the average time taken to calculate a single time step, in seconds. The vertical axis is the $\log_{10}$ of the error.

can, with quite a lot of effort, be rewritten as a matrix multiplication

$$\begin{bmatrix} \mathbf{y}^{n+1} \\ \vdots \\ \mathbf{y}^{n-K+1} \end{bmatrix} = T \begin{bmatrix} \mathbf{y}^{n} \\ \vdots \\ \mathbf{y}^{n-k} \end{bmatrix}.$$

The scheme is stable if it converges to zero independent of the initial conditions. It will do so if and only if the eigenvalues of the matrix T all have absolute values smaller than one.

The time stepping scheme we have been using, the second order Adam's Moulton method, will converge to zero, using any time step, for any differential equation that converges to zero. As a result, we will test the eigenvalues resulting from what are sometimes called the θ methods:

$$\mathbf{y}^{n+1} = \mathbf{y}^{n} + h\left[\theta M\mathbf{y}^{n} + (1-\theta)M\mathbf{y}^{n+1}\right].$$

This means checking the eigenvalues from the matrices

$$\left[I - h(1-\theta)M\right]^{-1}\left[I + h\theta M\right].$$

Any more general multi-step methods (using $\mathbf{y}^{n-1}$ and so on) will become too large for convenient eigenvalue calculation, but the θ methods provide a wide range of stability.

The partial differential equations we will use for testing purposes are the heat equation (Section 6.3) and the modified advection equation from 6.4.

First, we have the heat equation. This is a very stable problem, with large negative eigenvalues. For instance, the derivative matrix for the V_{-6} resolution has eigenvalues from $-4.10 \cdot 10^{4}$ to $-9.9 \cdot 10^{-2}$, with most in the 10^4 range. This makes it difficult to control the eigenvalues from the time stepping scheme itself. If we use the derivative matrix from V_{-6}, a time step of $h = 5$ and the second order implicit Adams' Moulton method, the time step has eigenvalues between 0.604 and $-1 + 1.95 \cdot 10^{-5}$. A multi-scale method equivalent, using resolutions of V_{-4} and V_{-6} on $(3.75, 6.25)$ and 4 'extra' terms, has eigenvalues between 0.604 and $-1 + 1.95 \cdot 10^{-5}$. Table 5 has a summary of absolute largest eigenvalues from multiple θ methods used on the heat equation. Table 6 has absolute largest eigenvalues from multiple time steps using a fixed $\theta = \frac{1}{2}$.

Table 5: Eigenvalues for different heat equation θ methods.

	Maximum Eigenvalue Size			
	$h = 5$		$h = 0.5$	
θ	Single System	Multi-Scale System	Single System	Multi-Scale System
0.00	$6.696 \cdot 10^{-1}$	$6.696 \cdot 10^{-1}$	$9.530 \cdot 10^{-1}$	$9.530 \cdot 10^{-1}$
0.10	$6.583 \cdot 10^{-1}$	$6.583 \cdot 10^{-1}$	$9.528 \cdot 10^{-1}$	$9.528 \cdot 10^{-1}$
0.20	$6.462 \cdot 10^{-1}$	$6.462 \cdot 10^{-1}$	$9.525 \cdot 10^{-1}$	$9.525 \cdot 10^{-1}$
0.30	$6.332 \cdot 10^{-1}$	$6.332 \cdot 10^{-1}$	$9.523 \cdot 10^{-1}$	$9.523 \cdot 10^{-1}$
0.40	$6.667 \cdot 10^{-1}$	$6.667 \cdot 10^{-1}$	$9.521 \cdot 10^{-1}$	$9.521 \cdot 10^{-1}$
0.50	$9.999 \cdot 10^{-1}$	$9.999 \cdot 10^{-1}$	$9.998 \cdot 10^{-1}$	$9.998 \cdot 10^{-1}$
0.60	1.500	1.500	1.500	1.500
0.70	2.333	2.333	2.333	2.333
0.80	4.000	4.000	4.000	4.000
0.90	9.000	9.000	8.996	8.996

A time step of $h = 5$ results in the matrix hM_{Der} having a largest magnitude eigenvalue in the $2.458 \cdot 10^5$ range. The $h = 0.5$ leads to a largest magnitude eigenvalue of $2.458 \cdot 10^4$.

Table 6: Eigenvalues for different heat equation time steps.

	Maximum Eigenvalue Size			
	$\theta = \frac{1}{2}$		$\theta = 0.2$	
h	Single System	Multi-Scale System	Single System	Multi-Scale System
0.00005	$1 - 4.935 \cdot 10^{-6}$	$1 - 4.935 \cdot 10^{-6}$	$1 - 4.935 \cdot 10^{-6}$	$1 - 4.935 \cdot 10^{-6}$
0.0005	$1 - 4.935 \cdot 10^{-5}$	$1 - 4.935 \cdot 10^{-5}$	$1 - 4.935 \cdot 10^{-5}$	$1 - 4.935 \cdot 10^{-5}$
0.005	$1 - 4.934 \cdot 10^{-4}$	$1 - 4.934 \cdot 10^{-4}$	$9.995 \cdot 10^{-1}$	$9.995 \cdot 10^{-1}$
0.05	$1 - 1.626 \cdot 10^{-3}$	$1 - 1.627 \cdot 10^{-3}$	$9.951 \cdot 10^{-1}$	$9.951 \cdot 10^{-1}$
0.5	$1 - 1.628 \cdot 10^{-4}$	$1 - 1.628 \cdot 10^{-4}$	$9.525 \cdot 10^{-1}$	$9.525 \cdot 10^{-1}$
5	$1 - 1.628 \cdot 10^{-5}$	$1 - 1.628 \cdot 10^{-5}$	$6.462 \cdot 10^{-1}$	$6.462 \cdot 10^{-1}$
50	$1 - 1.628 \cdot 10^{-6}$	$1 - 1.628 \cdot 10^{-6}$	$2.500 \cdot 10^{-1}$	$2.500 \cdot 10^{-1}$

Next we get to a more substantial linear problem, the modified advection problem from Section 6.4. One issue with this particular problem is that the time step is going to have, at least, an eigenvalue of one. Recall that the value of the solution at $x = 5$ is constant for all t. As a result, our largest eigenvalue will be one (or one to within machine precision), if and only if the time stepping scheme proves stable. The derivative matrix (using resolution V_{-6}) for this problem has one eigenvalue of exactly zero (relating to the constant value at $x = 0$), but has mostly large imaginary

eigenvalues (the largest has magnitude 159).

The stability for multiple time steps is not quite as good for the multi-scale method. However, an unreasonably large time step is required to get substantial instability. See Tables 7 and 8.

Table 7: Eigenvalues for our modified advection problem.

	Maximum Eigenvalue			
	$h = 5$		$h = 0.5$	
θ	Single System	Multi-Scale System	Single System	Multi-Scale System
0	$1 - 2.176 \cdot 10^{-14}$	$1 + 1.999 \cdot 10^{-14}$	$1 - 3.997 \cdot 10^{-15}$	$1 - 4.552 \cdot 10^{-15}$
0.1	$1 + 5.174 \cdot 10^{-14}$	$1 + 9.104 \cdot 10^{-15}$	$1 + 2.420 \cdot 10^{-14}$	$1 + 2.220 \cdot 10^{-15}$
0.2	$1 + 3.331 \cdot 10^{-14}$	$1 - 4.340 \cdot 10^{-15}$	$1 + 8.860 \cdot 10^{-14}$	$1 - 4.885 \cdot 10^{-15}$
0.3	$1 + 1.910 \cdot 10^{-14}$	$1 + 3.553 \cdot 10^{-15}$	$1 + 2.145 \cdot 10^{-13}$	$1 - 2.109 \cdot 10^{-15}$
0.4	$1 + 2.087 \cdot 10^{-14}$	$1 + 1.177 \cdot 10^{-14}$	$1 + 5.112 \cdot 10^{-13}$	$1 - 5.440 \cdot 10^{-15}$
0.5	$1 + 8.216 \cdot 10^{-15}$	$1 + 4.078 \cdot 10^{-8}$	$1 + 1.023 \cdot 10^{-13}$	$1 + 4.010 \cdot 10^{-7}$
0.6	1.500	1.500	1.499	1.495
0.7	2.333	2.333	2.331	2.320
0.8	4.000	3.998	3.990	3.954
0.9	8.999	8.992	8.901	8.901

The time step of $h = 5$ has a largest eigenvalue with magnitude 663.1 for hM_{Der}. Obviously, $h = 0.5$ has 66.31.

Table 8: More largest eigenvalues for the modified advection problem.

	Maximum Eigenvalue			
	$\theta = 0.5$		$\theta = 0.2$	
h	Single System	Multi-Scale System	Single System	Multi-Scale System
0.00005	$1 + 7.328 \cdot 10^{-15}$	$1 + 2.595 \cdot 10^{-9}$	$1 + 3.397 \cdot 10^{-14}$	$1 + 1.332 \cdot 10^{-15}$
0.0005	$1 + 5.107 \cdot 10^{-15}$	$1 + 2.594 \cdot 10^{-8}$	$1 + 5.122 \cdot 10^{-12}$	$1 + 1.110 \cdot 10^{-15}$
0.005	$1 + 9.326 \cdot 10^{-15}$	$1 + 2.578 \cdot 10^{-7}$	$1 + 3.109 \cdot 10^{-15}$	$1 - 2.443 \cdot 10^{-15}$
0.05	$1 + 6.240 \cdot 10^{-14}$	$1 + 1.574 \cdot 10^{-6}$	$1 + 1.745 \cdot 10^{-12}$	$1 - 1.221 \cdot 10^{-15}$
0.5	$1 + 1.024 \cdot 10^{-13}$	$1 + 4.010 \cdot 10^{-7}$	$1 + 8.860 \cdot 10^{-14}$	$1 - 4.885 \cdot 10^{-15}$
5	$1 + 8.216 \cdot 10^{-15}$	$1 + 4.078 \cdot 10^{-8}$	$1 + 3.331 \cdot 10^{-14}$	$1 - 4.330 \cdot 10^{-15}$
50	$1 + 1.932 \cdot 10^{-14}$	3.418	$1 + 2.487 \cdot 10^{-13}$	$1 - 3.853 \cdot 10^{-14}$

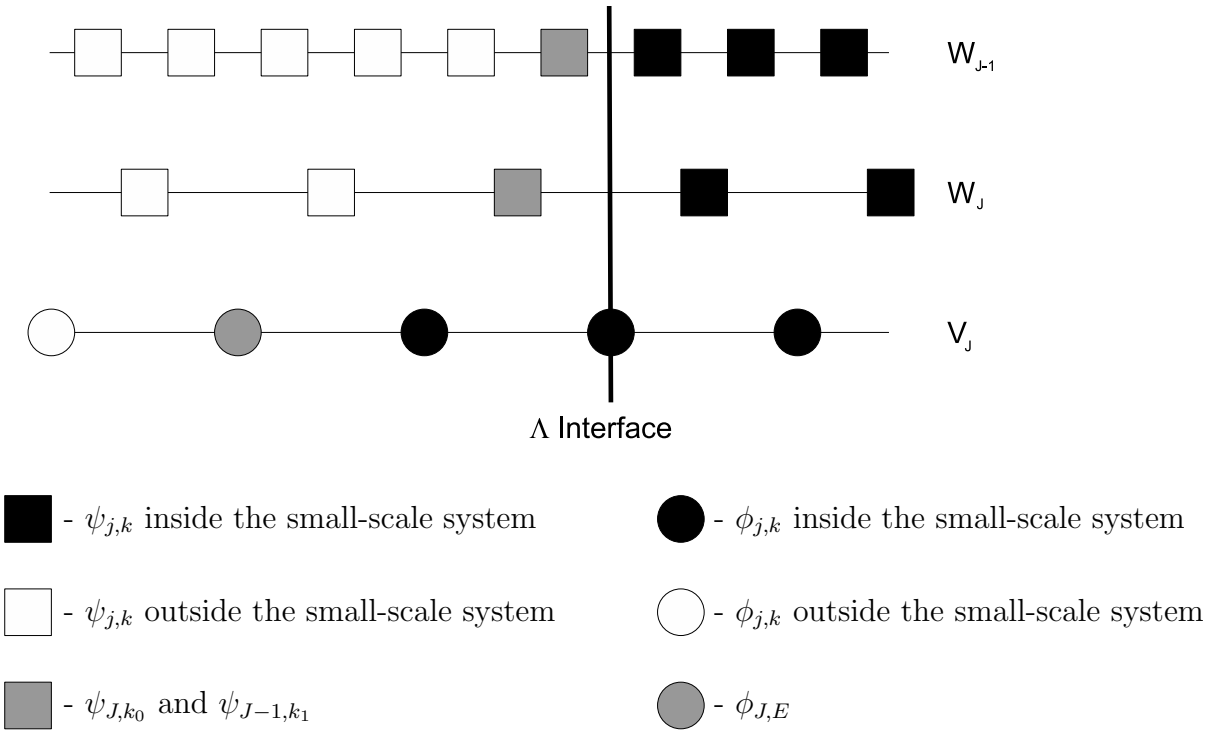


Figure 30: A simple diagram showing functions ψ and ϕ relevant to a possible adaptive scheme. The subdomain Λ is to the right of the interface. Notice that there are two 'extra' terms.

7.6 A Possible Adaptive Scheme

This is a simple outline of a possible scheme for setting the width of Λ, and the 'extra' terms, adaptively. Before even the preliminaries, we need to state clearly that this setup has not yet been tested. This is purely an example of how certain elements of the method could be decided adaptively.

Before we begin with an outline of how the sub-domain Λ and 'extra' values can be decided adaptively, we should first identify a few, key, coefficients and functions that will be discussed. We will consider only a single interface of Λ, and assume that Λ is on the positive side of the interface (so a higher k in $\phi_{j,k}$ will increase the chance of the scaling function being centered in Λ).

- First, we have a single scaling function $\phi_{J,E}$, with J equal to the resolution of the large-scale system, and E such that $\phi_{J,E}$ is just outside the small-scale system.

What this means is that $\phi_{j,E}$ would be included in the small-scale system if the 'extra' width was increased by one. The coefficient $a_{j,E}$ for this function will be found in $\mathbf{a}_1$.

We will assume that the Λ interface under consideration is such that $\phi_{j,E+1}$ is in the small-scale system. This means that $\phi_{j,E+1}$ is the outermost scaling function $\phi_{J,k}$ included in the small-scale system. If we reduce the number of 'extra' terms by one, then $\phi_{j,E+1}$ will no longer be in the small-scale system.

- Next, we have a set of wavelets ψ_{J-n,k_n}, $n = 0, \ldots, N$. These functions are going to be finer resolution than the large-scale system, since $J - n \leqslant J$. They will also be located outside of Λ, but, like $\phi_{J,E}$, not that far outside of Λ. For the interface we are discussing, this means we want ψ_{J-n,k_n+1} to be the outermost function $\psi_{J-n,k}$ included in the small-scale system.

  Due to how $\psi_{j,k}$ are spaced, changing the width of Λ will do more than include the ψ_{J-n,k_n} or remove the ψ_{J-n,k_n+1}. Increasing the width of Λ will result in the inclusion of

$$\psi_{J-n,k_n}, \ldots, \psi_{J-n,k_n-2^n+1}, \qquad \text{for } n = 0, \ldots, N.$$

  Decreasing the width of Λ will result in

$$\psi_{J-n,k_n+1}, \ldots, \psi_{J-n,k_n+2^n}, \qquad \text{for } n = 0, \ldots, N.$$

We will use a set of error tolerances T_1 to T_6 (all > 0, dependent only on required accuracy and the nature of the problem) to determine how many functions/coefficients can be safely ignored. We will, occasionally, use norm values on the vectors. Which norm is not specified, but $\|\cdot\|_\infty$ and $\|\cdot\|_2$ would work. Note, again, that this sequence is present to illustrate the type of testing that could be done to manage a multi-scale method. These procedures have not been tested.

1. Solve the large-scale system, (7.2.8), calculating $\mathbf{a}^{T_m}$.

2. Check error source (1): calculate the components of the vector

$$\mathbf{E}_1 = \begin{bmatrix} B_1 & B_2 \end{bmatrix} \left(\mathbf{a}^n + \mathbf{a}^{T_m} \right),$$

corresponding to b_{J-n,k_n} for $n = 0 \dots N$ (the the components of $\mathbf{E}_1$ corresponding to the ψ_{J-n,k_n} discussed above).

If any of the calculated components of $\mathbf{E}_1$ have absolute value $> T_1$, then widen Λ by 2^J (so Λ will include one more $\phi_{J,k}$, and all of the ψ_{J-n,k_n}.

3. Check error source (2). Take the coefficient $a_{J,E}$ and multiply it by the Eth column of B_3 (the column corresponding to $\phi_{J,E}$) to calculate the vector $\mathbf{E}_2$.

   If $\|\mathbf{E}_2\| > T_2$ then increase the number of 'extra' terms by one.

4. Solve the small-scale system, (7.2.9), calculating $\mathbf{a}_\Lambda^{C_r}$, $\mathbf{a}^{n+1}$ and $\mathbf{b}^{n+1}$.

5. Check the error source (1) again. Calculate the components of the vector

$$\mathbf{E}_1 = A_2 \left(\mathbf{b}_\Lambda^n + \mathbf{b}_\Lambda^{n+1} \right) + B_2 \left(\mathbf{a}_\Lambda^n + \mathbf{a}_\Lambda^{n+1} \right)$$

   corresponding to the coefficients b_{J-n,k_n} (so the components corresponding to ψ_{J-n,k_n}) for $n = 0 \dots N$. If $|\mathbf{E}_1| > T_1$ for any of those components, then widen Λ by 2^J. Having widened Λ, we are done (skip remaining steps).

6. If Λ was not widened in the previous step, start checking error sources (2) and (3). First, define E_3 as the component of the vector

$$\mathbf{E}_3 = C2 \left(\mathbf{b}_\Lambda^{n+1} + \mathbf{b}_\Lambda^n \right) - D_2 \mathbf{a}_\Lambda^{C_r}$$

   corresponding to $a_{J,E}$ (so the component corresponding to $\phi_{J,E}$). If $|E_3| > T_2$ then increase the number of 'extra' terms by one and skip to step 8.

7. If $|E_3| \leqslant T_2$ then check if the current 'extra' is needed (as long as there is at least one 'extra' term).

   (a) First, check the effect of the small-scale system on the outermost 'extra' term. Define $E_4 = \left(\mathbf{a}_\Lambda^{C_r} \right)_{E+1}$, the coefficient $a_{J,E+1}$ (for $\phi_{J,E+1}$, the outermost 'extra' term) in $\mathbf{a}_\Lambda^{C_r}$. If $|E_4| / h > T_4$ then the 'extra' term is necessary, so go on to step 8.

(b) Now check the effect of the outermost 'extra' term on the small-scale system. Define $a_{E+1} = a^{n+1}_{J,E+1} + a^{n}_{J,E+1}$, the coefficients $a_{J,E+1}$ from the vectors $\mathbf{a}^n$ and $\mathbf{a}^{n+1}$. The vector E_5 will be $\mathbf{a}_{E+1}$ multiplied by the $a_{J,E+1}$ ($\phi_{J,E+1}$) related column from B_Λ. If $\|E_5\| < T_5$ then the outermost 'extra' term is not necessary. Remove one 'extra' term (removing $\phi_{J,E+1}$ from the small-scale system).

8. Finally, check if the width of Λ is necessary. Define the vector $\mathbf{b} = \left(\mathbf{b}^{n+1}_\Lambda + \mathbf{b}^{n}_\Lambda\right)$, with most of its components set to zero. Specifically, set to zero all of the coefficients $b_{j,k}$ that are not in

$$b_{J-n,k_n+1}\,,\ \ldots\ ,\ b_{J-n,k_n+2^n} \qquad \text{for} \qquad n = 1 \ldots N.$$

These coefficients relate to the functions ψ in the small-scale system that are within 2^J of the interface of Λ. These are precisely the coefficients (and functions) that would be removed if Λ is shrunk by 2^J.

Next, calculate the vector

$$\mathbf{E}_6 = \begin{bmatrix} A_\Lambda \\ C_\Lambda \end{bmatrix} \mathbf{b}.$$

If $\|\mathbf{E}_6\| < T_6$ then shrink Λ by 2^J and increase the number of 'extra' terms by one.

7.7 Context

It seems appropriate at this point to include some discussion on other work similar to our own. Very little of it can be found. While wavelet methods for partial differential equations are numerous, and the number grows quickly, most are of a different nature from our multi-scale method. They are, as discussed in Section 1.1, mostly about determining the ideal resolution at different locations. Sometimes the coefficients $b_{j,k}$ are approximated from finite difference data to determine if the resolution near $k2^j$ is sufficient, like in [19]. Other times the $b_{j,k}$ data is used directly, and $b_{j,k-1}$, $b_{j,k+1}$, $b_{j-1,2k}$ are included in subsequent time steps to see if they are relevant, like

in [13]. These types of methods, useful as they are, have nothing in common with ours other than the use of wavelets. In fact, most (if not all) adaptive schemes are compatible with ours. For instance, the small-scale system on Λ could use an adaptive decomposition, as could the large-scale system on Ω. In fact, the creation of the small-scale system could be made adaptive: the $b_{j,k}$ data could be analyzed to determine where and when more levels of refinement are needed. So, methods that use wavelets to analyze or efficiently express a problem are not particularly connected to our own. Now we need to discuss the methods that are particularly similar to our own.

The main concept, and advantage, to our approach is the ability to solve a time step at different resolutions, in different domains, sequentially. Solving a problem at multiple resolutions is an obvious way to exploit a multi-scale decomposition. As such, there are similar methods to be found throughout the literature. The most similar are those intended to be used to solve elliptic problems at multiple scales. One that is particularly connected to our own method, due to its use of biorthogonal wavelets, can be found in [27] (also a particularly well written discussion of such methods). Other examples can be found in [28, 21 and 20]. Our method is different in that it is intended to be used on time dependent problems, though this is, in fact, a fairly shallow difference. More importantly, adaptive multi-scale solvers for elliptic problems use repeated iterations at each resolution to maintain consistency and reduce the error. Our method takes great pains to make sure that the different levels of resolution, the different systems, are consistent with each other. As such, each system only needs to be solved once (per time step). This is not an advantage in an elliptic solver, but when several thousand (if not million) time steps have to be solved, it is usually best to avoid having to repeatedly calculate individual times steps.

Another perspective on our method is that it allows, in effect, the problem to be solved in a different manner in Λ than over the remainder of Ω. This difference is, at the moment, restricted to the use of a finer resolution, but it could well be that using a different time step altogether could be an option (note that this is not yet tested thoroughly). This means that our method has a property similar to that of 'Domain Decomposition Methods.' At this point we come to a similar concept separating our

method from others. These methods typically require multiple iterations between the different domains to be accurate. Furthermore, they typically require limited interaction across the interfaces between domains. Our method has a much less stringent requirement. It requires that the problem be adequately modeled around the interfaces by the large-scale resolution. As long as the small-scale interactions are restricted to the interior of Λ, multiple iterations are not necessary. A single solving in Ω and then one more in Λ will suffice.

Chapter 8

The Rossby Wave Problem

In this section we will introduce a more substantial problem than those we have looked at before. Rossby waves are perturbations of predominantly east/west flows in the atmosphere or oceans. They are exceedingly large, their wavelengths have the same order of magnitude as the earth's circumference, so the atmospheric forms are frequently called 'planetary waves.' The behavior of Rossby waves has been studied extensively: analytically, asymptotically, numerically, and empirically. We have solid numerical results to compare with our own (starting with [2], published in 1976), as well as asymptotic results (like from [7], numerical results also included). We will use Rossby waves as the source of our second example problem, so the physical interpretation is of limited importance. For those interested in a fuller understanding, explanation can be found fairly easily, from texts on fluid dynamics ([18, p. 632]) to meteorology ([25, p.236]).

8.1 The Problem

The problem is time dependent on an x, y, domain, with reference lengths L_x and L_y, respectively. There are periodic boundary conditions in the x direction, and fixed boundary conditions in the y direction. First, we will nondimensionalize the problem, converting the domain to $x \in [0, 2\pi)$ and $y \in [0, 10]$. The primary effect of this is that

the Laplacian operator $\left(\frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial x^2}\right)$ will be re-scaled to

$$\Delta = \left(\frac{\partial^2}{\partial y^2} + \delta\frac{\partial^2}{\partial x^2}\right), \quad \text{with} \quad \delta = \left(\frac{L_y}{L_x}\right)^2. \tag{8.1.1}$$

The problem is based on the nondimensionalized barotropic vorticity equation

$$\Delta\Psi_t + \Psi_x\Delta\Psi_y - \Psi_y\Delta\Psi_x + \beta\Psi_x - \nu\Delta^2\Psi = 0, \tag{8.1.2}$$

(see [18, p. 637] for details) with the (real) function $\Psi(x, y, t)$ the streamfunction on the domain, so the x directional flow $u(x, y, t) = -\Psi_y(x, y, t)$ and the y directional flow is $v(x, y, t) = \Psi_x(x, y, t)$ (see the glossary for more on streamfunctions). Always remember that Ψ is a streamfunction, not a wavelet (wavelets are identified by ψ, the lower case of Ψ). The variable x represents longitude (east-west) and y the latitude (north-south). The coefficient β relates to planetary rotation and ν is the viscosity coefficient.

Next we break up the streamfunction Ψ into two components,

$$\Psi(x, y, t) = \overline{\Psi}(y) + \epsilon\, P(x, y, t), \tag{8.1.3}$$

with ϵ small relative to Ψ and $\overline{\Psi}$. The function $\overline{\Psi}(y)$ is the streamfunction for the basic flow. It is a function only of y, meaning that the basic flow is only in the x direction, and constant in time. The basic flow will always be a given (known) function, $\overline{u}(y)$, meaning $\overline{\Psi}'(y) = -\overline{u}(y)$. The Rossby waves are found in $P(x, y, t)$, the streamfunction representing small perturbations of the basic flow. This is the function we will be attempting to solve.

Substituting (8.1.3) into (8.1.2) leads to

$$\epsilon\Delta P_t + \epsilon P_x\left(-\overline{u}''(y) + \epsilon\Delta P_y\right) - \left(-\overline{u}(y) + \epsilon P_y\right)\epsilon\Delta P_x + \epsilon\beta P_x - \nu\left(\overline{u}^{(3)}(y) + \epsilon\Delta^2 P\right) = 0. \tag{8.1.4}$$

Next, we substitute in $Z = \Delta P$, the vorticity of P (see Glossary), and divide by ϵ for

$$Z_t + \overline{u}(y)Z_x + \left(\beta - \overline{u}''(y)\right)P_x - \epsilon\left(P_y Z_x - P_x Z_y\right) - \nu\Delta Z + \frac{\nu}{\epsilon}\overline{u}^{(3)}(y) = 0, \tag{8.1.5}$$

(with $\overline{u}^{(3)}$ the third derivative of $\overline{u}$). We will ignore the y dependent term at the end ($\overline{u}^{(3)}$ is small, and even when $\nu \neq 0$ we will keep ν much smaller than ϵ). The

equation becomes

$$Z_t = -\overline{u}(y)Z_x - (\beta - \overline{u}''(y))\, P_x + \nu\Delta Z + \epsilon\left(P_y Z_x - P_x Z_y\right). \tag{8.1.6}$$

Recall that ν represents viscosity, and that ϵ, now the coefficient for the non-linear term, is assumed to be small. Our examples will use $\overline{u}(y) = \tanh(y-5)$. This has several implications. First, the basic flow will be very nearly constant in the regions near $y = 10$ and $y = 0$, (at $\tanh(y) \approx 1$ and $\tanh(y) \approx -1$, respectively). Second, the basic flow velocity will be zero at $y = 5$. As we will see, these properties of $\overline{u}$ have interesting implications for P.

As for boundary conditions, we have $P(x, 10, t) = \cos(2x)$ and $P(x, 0, t) = 0$ (recall that the x directional boundary conditions are periodic). Notice that the boundary conditions are all composed of $\cos(2x)$, so they can be written in terms of $e^{\pm 2ix}$. If we use the linear form of the problem (set $\epsilon = 0$), then P (and Z) can be written using only functions $e^{\pm 2ix}$ in the x direction (each multiplied by a function of y and t). The initial condition is simply $P(x, y, 0) = 0$ on $[0, 2\pi) \times (0, 10)$.

Now go to the simplest case, set $\nu = 0$ along with ϵ. If we assume that the problem has a steady state, we can find that it will have the form

$$P(x,y) = \Phi_2(y)\, e^{i2x} + \Phi_{-2}(y)e^{-i2x}, \quad \frac{d^2}{dy^2}\,\Phi_\kappa(y) + \left(-\delta\kappa^2 + \frac{\beta - \overline{u}''(y)}{\overline{u}(y)}\right)\Phi_\kappa(y) = 0. \tag{8.1.7}$$

In the area immediately around $y = 10$, $\overline{u}(y)$ is approximately one and $\overline{u}''(y)$ will be approximately zero. The problem becomes $\left(\frac{d^2}{dy^2} + (\beta - \delta\kappa^2)\right)\Phi = 0$, with a solution of the form $e^{\pm i\rho(y-10)}$, $\rho = \sqrt{\beta - \delta k^2}$. As a result, recalling that the $y = 10$ boundary is just $\cos(2x)$, the steady state is similar to

$$P(x,y) = \cos\left(2x + \rho(y-10)\right) \tag{8.1.8}$$

near $y = 10$. The more general observation is that ρ, where

$$\rho^2 = \left(-\delta\kappa^2 + \frac{\beta - \overline{u}''(y)}{\overline{u}(y)}\right), \tag{8.1.9}$$

is the wave number in the y direction (so we get $e^{\pm i\rho}$ terms). This gives us some information on the behavior of P in the center of the y domain. Notice that ρ^2

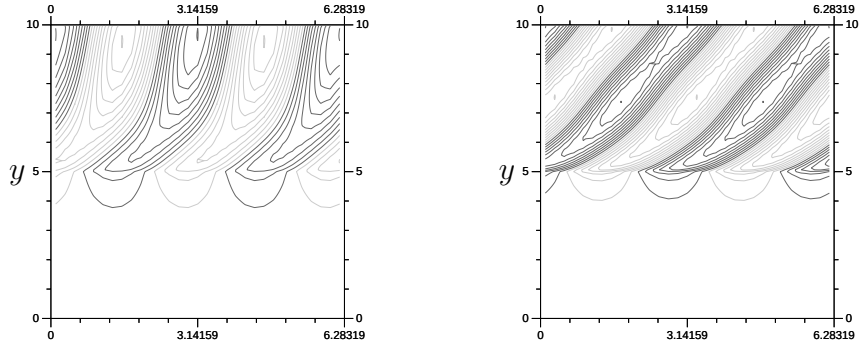
approaches infinity as $y \to 5$, since $\overline{u}(y) \to 0$ as $y \to 5$. The implication is that the waves stop propagating when they hit $y = 5$, so the steady state on $y \in (0, 5)$ will be approximately zero. All of these observations can be seen in Figure 31. The waves curve then stop at $y = 5$, and the different values of β used result in different angles of propagation. These are also consistent with the literature, Figure 31 (a) is consistent with all the P (equivalent) plots in [7], and virtually identical to [7, Figure 2(a)]. The behavior of the function Z is also fairly well known. Figure 32 contains plots of Z at different times, all taken from the same results as Figure 31 (a). The general Z shape expected for the linear problem is the set of peaks around $y = 5$, seen in Figure 32. Notice that the amplitude of Z near $y = 5$ is much higher than that anywhere else, and that the waves in the center get more tightly concentrated as t increases. As we will see, this results in a greater requirement for fine resolution in the center as t increases. The shape of our Z plots in Figure 32 match well with [7, Figure 18(a)] and [8, Figure 4], even though our results use different domains and boundary conditions.

Apart from P and Z, there is one more quantity we will use to check the accuracy of our results. We will be interested in the momentum of the system, specifically the momentum flux of the critical region around $y = 5$. Momentum is mass multiplied by velocity, which in theoretical fluid dynamics means an integral of the product of velocity and density (see [18, p. 88]). Since density is constant in our system, we need only focus on the velocity. Consider the region $y \in [2.5, 7.5]$. We will need to calculate the flux at the boundaries $y = 2.5$ and 7.5. We will focus on the flux of the x directional component of the momentum. This makes the flux over a fixed y simply the x component of the velocity times the y component of the velocity, so $-P_y P_x$. So the total flow over y, at a time t, will be

$$-\int_0^{2\pi} P_x(x, y, t) P_y(x, y, t)\, dx. \tag{8.1.10}$$

The flux for the region $[2.5, 7.5]$ will be the flux at $y = 2.5$ minus the flux at $y = 7.5$, so

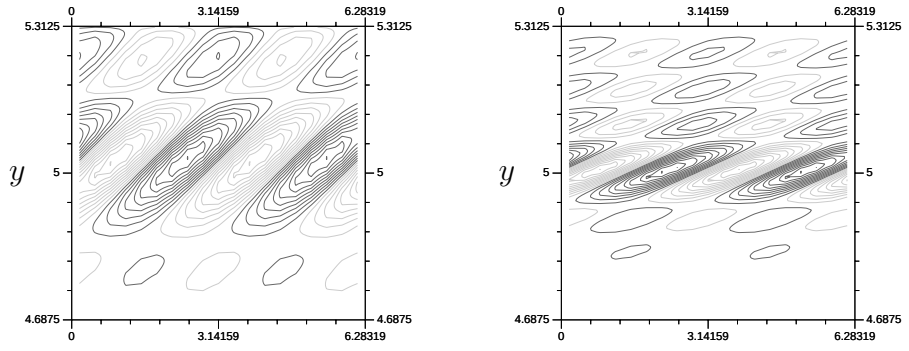
$$F(t) = -\int_0^{2\pi} P_x(x, 2.5, t) P_y(x, 2.5, t)\, dx + \int_0^{2\pi} P_x(x, 7.5, t) P_y(x, 7.5, t)\, dx, \tag{8.1.11}$$



(a) $\beta = 1$ (b) $\beta = 2$

(c) $\beta = 4$ (d) $\beta = 6$

Figure 31: Steady state P plots using differing values of β. These all use $\delta = 0.2$ and $\nu = \epsilon = 0$. Notice the angle changing with β.



(a) Z at $t = 25$ (b) Z at $t = 50$

(c) Z at $t = 75$ (d) Z at $t = 100$

Figure 32: Z plots from the Rossby problem. All of these use the, from now on, standard $\beta = 1$ (with $\delta = 0.2$ and $\nu = \epsilon = 0$). Specifically, these come from the same results as Figure 31 (a).

though the first integral is approximately zero since $P \approx 0$ on $y = 2.5$. In this particular system, there are some very clear expectations. First, the linear problem is expected to result in the critical (central) region absorbing momentum, resulting in a positive flux. In [2, Figure 5, line 1] and [7, Figure 3], the flux for the linear problem can be seen reaching its maximum early, then lowering and stabilizing at a positive value. This is the expected behavior of the linear problem. When the non-linear interactions are included, the critical region is expected to reflect some of the momentum. The flux, as seen in [2, Figure 5, lines 2,3] and [7, Figure 3b], is expected to reach its maximum value early, then reduce down to zero and begin oscillating around zero. Later, we will test our numerically derived results against these expectations.

8.2 Calculation

Recall that our boundary conditions are $P(x, 10, t) = \cos(2x)$ and $P(x, 0, t) = 0$ (with periodic boundary conditions in the x direction). As in Section 6.5, this will be done using a boundary value function, breaking up $P(x, y, t)$ into

$$P(x, y, t) = P_1(x, y, t) + P_2(x, y). \tag{8.2.1}$$

Both functions will have periodic boundary conditions at $x = 0$ and $x = 2\pi$. Also, $P_1(x, 10, t) = P_1(x, 0, t) = 0$, with the same true of its vorticity, $Z_1 = \Delta P_1$. So, P_1 is the time dependent component that we will be calculating at every time step, with zero boundary conditions satisfied in the manner explained in Section 6.2. Since the initial conditions are zero, $P_1(x, y, 0) = Z_1(x, y, 0) = 0$. The function P_2 is the boundary function, satisfying the boundary conditions, but constant in time. With $P_1(x, 10) = 0$, we will require $P_2(x, 10) = \cos(2x)$ and $P_2(x, 0) = 0$. The value of $Z_2 = \Delta P_2$, the vorticity of the boundary function, will be figured out next.

We will keep the boundary conditions constant, meaning $\frac{\partial}{\partial t} Z(x, 10, t) = 0$. Assume that $\epsilon = 0$ and $\nu = 0$. Putting P_1, P_2, Z_1 and Z_2 into (8.1.6) yields

$$\frac{\partial}{\partial t} Z(x, 10, t) = -\overline{u}(10) \frac{\partial}{\partial x} \Delta P_2(x, 10) - [\beta - \overline{u}''(10)] \frac{\partial}{\partial x} P_2(x, 10) = 0, \tag{8.2.2}$$

due to the fact that P_1 and Z_1 are both zero at $y = 10$. So, we get

$$\frac{\partial}{\partial x}\left(\Delta P_2(x,10) + \frac{\beta - \overline{u}''(10)}{\overline{u}(10)}\, P_2(x,10)\right) = 0, \tag{8.2.3}$$

meaning that

$$Z_2(x,10) = \Delta P_2(x,10) = -\left(\frac{\beta - \overline{u}''(10)}{\overline{u}(10)}\right)\cos(2x). \tag{8.2.4}$$

There is a similar requirement for $y = 0$, but it can be easily satisfied by setting $Z_2 = P_2 = 0$ at $y = 0$. So, we will use the boundary values

$$\begin{aligned} P_2(x,10) &= \cos(2x), & P_2(x,0) &= 0, \\ Z_2(x,10) = (\Delta P_2)(x,10) &= -\left(\frac{\beta - \overline{u}''(y)}{\overline{u}(y)}\right)\cos(2x), & \Delta P_2(x,0) &= 0. \end{aligned} \tag{8.2.5}$$

Those boundary conditions, combined with the periodic boundary conditions at $x = 0$ and $x = 2\pi$, make Fourier decomposition ideal for this problem. The Fourier decomposition in the x direction creates functions $\widehat{P}(\kappa, y, t)$ and $\widehat{Z}(\kappa, y, t)$, with

$$P(x,y,t) = \sum_{\kappa\in\mathbb{Z}} \widehat{P}(\kappa,y,t)e^{i\kappa x} \quad \text{and} \quad Z(x,y,t) = \sum_{\kappa\in\mathbb{Z}} \widehat{Z}(\kappa,y,t)e^{i\kappa x}. \tag{8.2.6}$$

Since P and Z are real functions, $\widehat{P}(-\kappa,y,t) = \overline{\widehat{P}(\kappa,y,t)}$, so we only need to work with one of $\pm\kappa$ at a time. For general Fourier mode κ we get

$$\widehat{Z}(\kappa,y,t) = \widehat{P}_{yy}(\kappa,y,t) - \kappa^2\delta\,\widehat{P}(\kappa,y,t) = \left(\frac{\partial^2}{\partial y^2} - \kappa^2\delta\right)\widehat{P}(\kappa,y,t) \tag{8.2.7}$$

and (assuming $\epsilon = 0$)

$$\frac{\partial}{\partial t}\widehat{Z}(\kappa,y,t) = -\kappa i\,\overline{u}(y)\widehat{Z}(\kappa,y,t) - \big(\beta - \overline{u}''(y)\big)\kappa i\,\widehat{P}(\kappa,y,t) + \nu\left(\frac{\partial^2}{\partial y^2} - \kappa^2\delta\right)\widehat{Z}(\kappa,y,t). \tag{8.2.8}$$

As was briefly discussed in Section 8.1, the boundary conditions of $\cos(2x)$ and zero (at $y = 10$ and $y = 0$, respectively) make the linear problem expressible using only $\widehat{P}(2,y,t)$. The Fourier decomposition simplifies a real valued problem with two spatial dimensions into a complex problem with one spatial dimension. Of course, Fourier decomposition has advantages for the non-linear problem as well. Using the given

boundaries, the only non zero $\widehat{Z}(\kappa, y, t)$ have κ equal to an integer multiple of two. So, with the x direction modeled via a Fourier decomposition, we now focus on the y direction. The $\widehat{P}(\kappa, y, t)$ and $\widehat{Z}(\kappa, y, t)$ are approximated using wavelets.

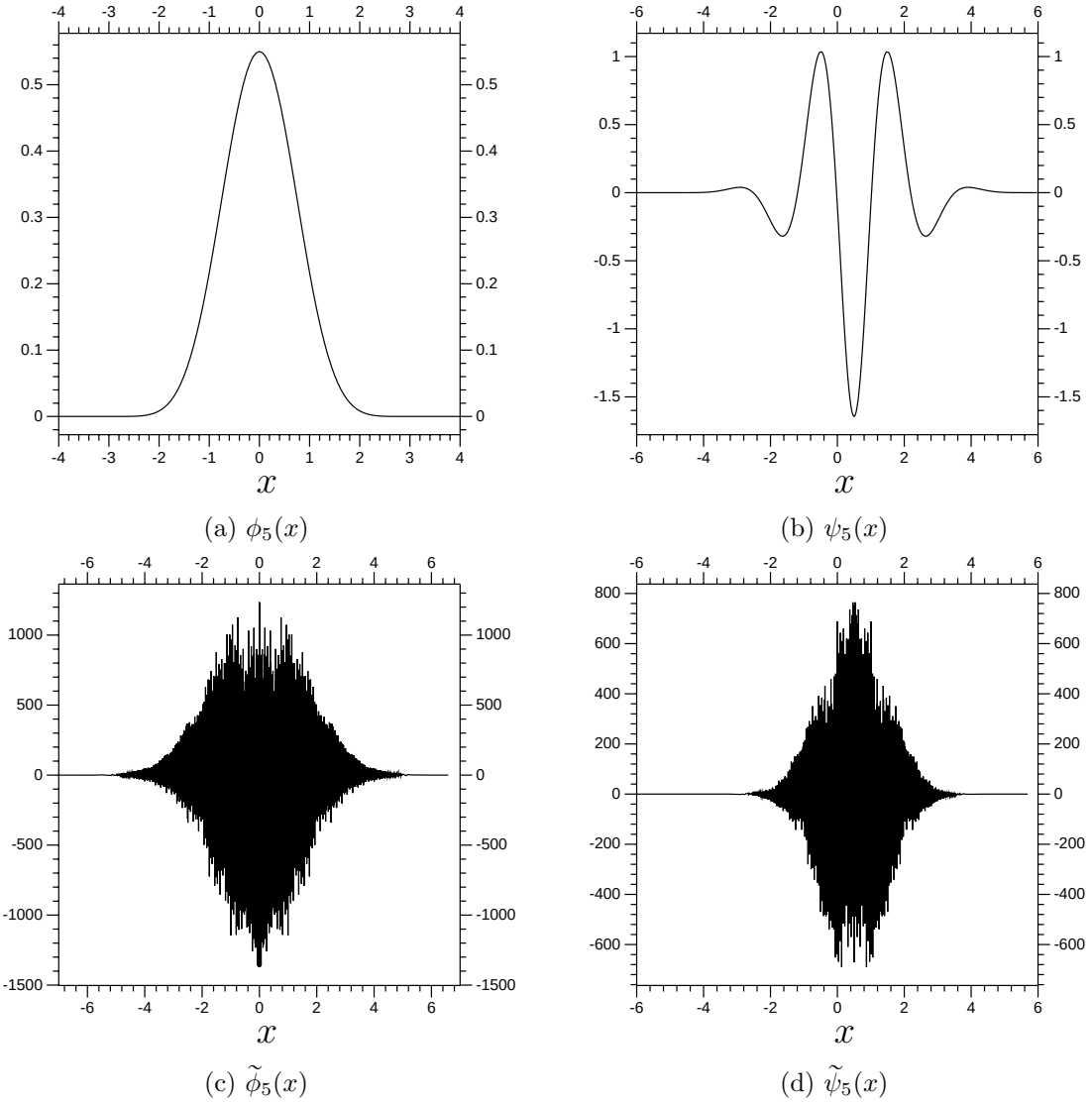
The non-linear Rossby wave equation, (8.1.6), includes a third y derivative of P on the right hand side (in Z_y). The scaling function and wavelet used in Chapter 6 have only a weak third derivative. We will use a higher order pair, based on the spline B_5 (see Section A.3). Figure 33 has our new ϕ and ψ, and one advantage of switching to higher order splines is shown with the V_j decompositions of the second derivative of ϕ (Figure 35).

Now it is time to look towards numerical computation of Equation (8.1.6), with the boundaries given. As discussed earlier, modeling the function $Z(x, y, t)$ will be done via a Fourier decomposition in the x direction, creating functions $\widehat{Z}(\kappa, y, t)$ (with κ the Fourier mode), which, in turn, will be modeled via a wavelet decomposition. Therefore, if we use V_J ($J \leqslant 0$), and Fourier modes $\kappa \in [-K, K]$, we will be dealing with function data of the form

$$Z(x, y, t) = \sum_{\kappa=-K}^{K} e^{i\kappa x} \left(\sum_k a_{\kappa,0,k}\, \phi_{0,k}(y) + \sum_{j=J+1}^{0} \sum_k b_{\kappa,j,k} \psi_{j,k}(y) \right) \tag{8.2.9}$$

at any particular time t. We will work with an approximation of Z at different time steps. We will need the function P to calculate Z_t, calculated using a approximations of $\Delta^{-1} Z$. Writing the functions in terms of Z, rather than P, is an easy decision to make. First, the derivative is written in terms of Z, so it makes things easier. More importantly, in fact, it is necessary because Z has a lot of small-scale values and interactions in the center region (around $y = 5$), while P is more evenly distributed. As a result, there are serious advantages to using wavelets while working with Z. More on that later. For now, we must deal with the boundary conditions.

The periodic boundary conditions on $x = 0$ and $x = 2\pi$ are satisfied by the Fourier decomposition. The boundary conditions at $y = 10$ and $y = 0$ are a little more difficult. First, recall from (8.2.1) that P has been broken into $P_1(x, y, t)$ and $P_2(x, y)$. We decided (at the beginning of Section 8.2) to set $P_1(x, 10, t) = P_1(x, 0, t) = 0$, and the same for Z_1, using the approach outlined in Section 6.2. All that remains are the P_2 (and Z_2) based boundary conditions in (8.2.5).



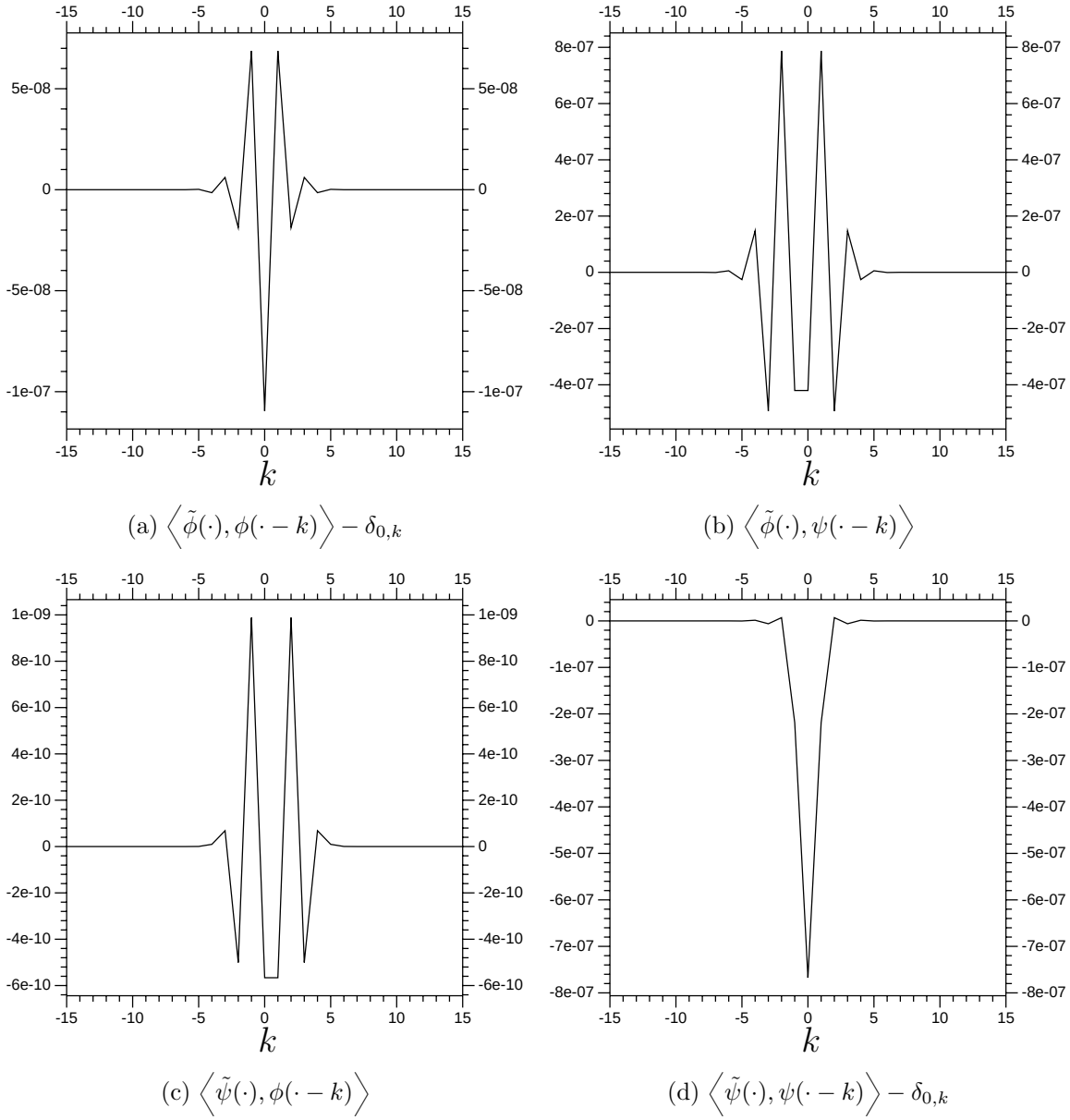
(a) $\phi_5(x)$ (b) $\psi_5(x)$

(c) $\widetilde{\phi}_5(x)$ (d) $\widetilde{\psi}_5(x)$

Figure 33: Our new ϕ and ψ with their duals.

We need to decide some specifics about how we are to create $P_2(x, y)$. We will, unless it is said otherwise, be using a function $P_2(x, y)$ based upon $\phi_{0,k}(y)$ terms (so a V_0 function in the y direction). Switching to the Fourier transform, $\widehat{P}_2(\kappa, y)$, and setting $\epsilon = \nu = 0$ gives us a fairly simple system to solve. We only need to use $\kappa = 2$, and will avoid including $\phi_{j,k}$ centered below $y = 10$, using only

$$\widehat{P}_2(y, 2) = a_{10}\phi(y - 10) + a_{11}\phi(y - 11), \tag{8.2.10}$$



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

(b) $\left\langle \tilde{\phi}(\cdot), \psi(\cdot - k) \right\rangle$

(c) $\left\langle \tilde{\psi}(\cdot), \phi(\cdot - k) \right\rangle$

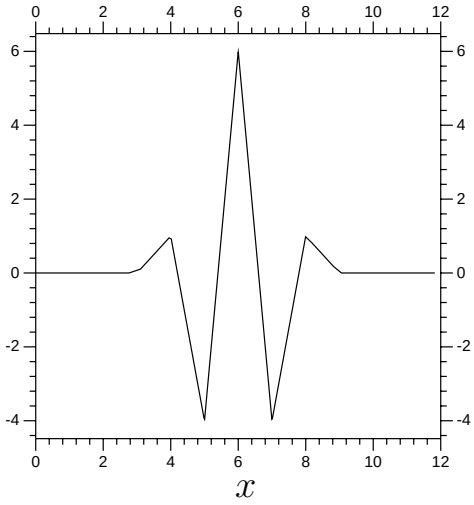
(d) $\left\langle \tilde{\psi}(\cdot), \psi(\cdot - k) \right\rangle - \delta_{0,k}$

Figure 34: Inner product related errors for the functions in Figure 33. Note that the inner products $\left\langle \widetilde{\psi}_0, \phi_k \right\rangle$ and $\left\langle \widetilde{\phi}_0, \psi_k \right\rangle$ are supposed to be zero for all k, while $\left\langle \widetilde{\phi}_0, \phi_k \right\rangle$ and $\left\langle \widetilde{\psi}_0, \psi_k \right\rangle$ are supposed to be equal to $\delta_{0,k}$.

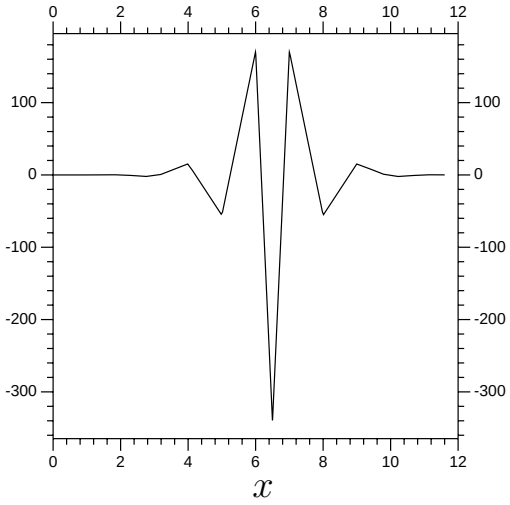
with a_{10} and $a_{11} \in \mathbb{R}$. This is because we only have two equations to satisfy:

$$\widehat{P}_2(10,2) = \frac{1}{2}\,, \quad \text{and} \quad \left(\frac{\partial^2}{\partial y^2} - 4\right) \widehat{P}_2(10,2) = -\frac{1}{2}\left(\frac{\beta - \overline{u}''(y)}{\overline{u}(y)}\right). \tag{8.2.11}$$

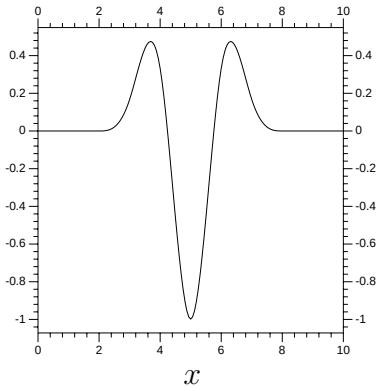
The first makes $P_2(x, 10) = \cos(2x)$, the second keeps P_2 consistent with Equation



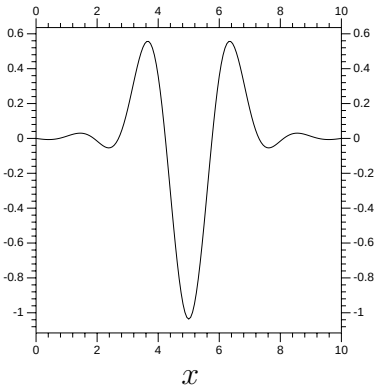
(a) $\frac{\partial^4}{\partial x^4}\,\phi_5(x)$



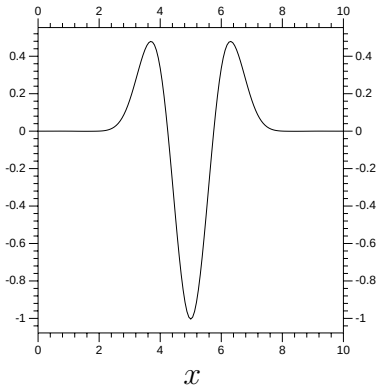
(b) $\frac{\partial^4}{\partial x^4}\,\psi_5(x)$



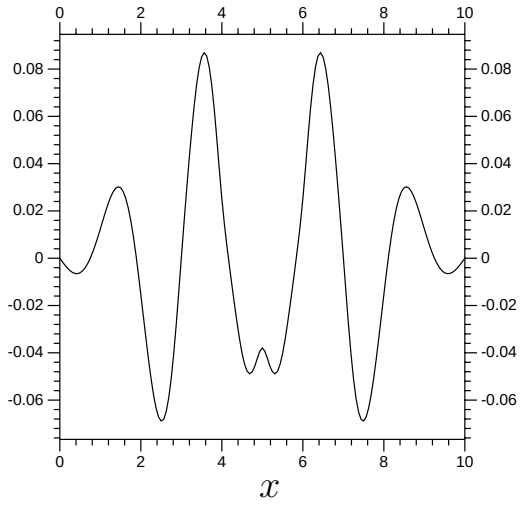
(c) $\frac{\partial^2}{\partial x^2}\,\phi(x)$



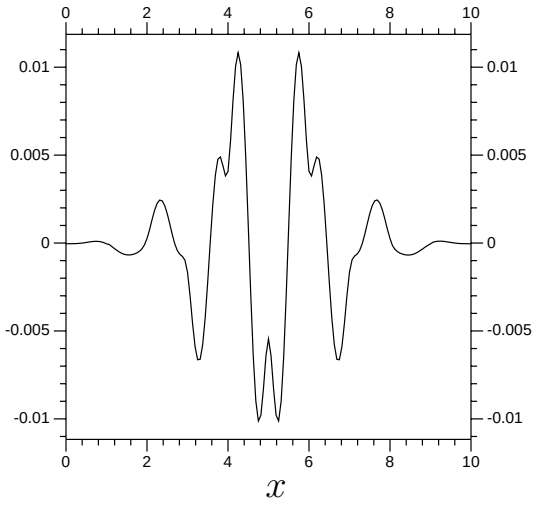
(d) V_0 approximation



(e) V_1 approximation

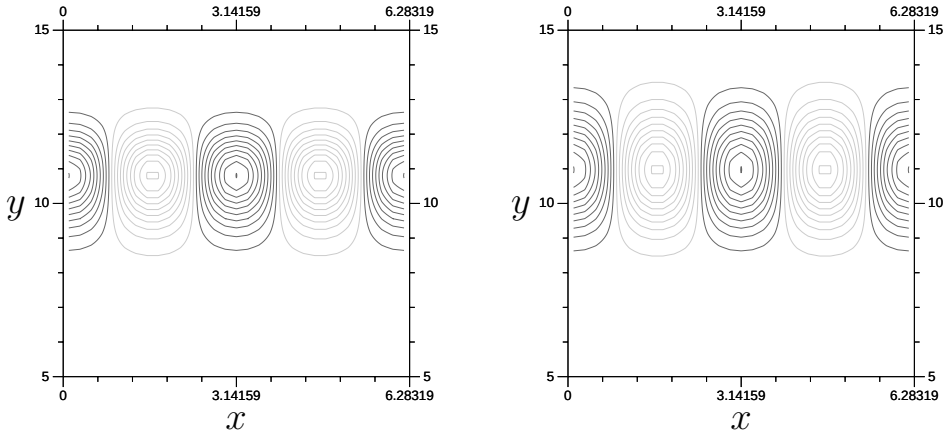


(f) V_0 error



(g) V_1 error

Figure 35: Derivatives and derivative approximations for our new ϕ. These were all plotted at a resolution of 2^8 per unit. The exact derivatives were calculated using the method in (A.3.3). The finite resolution approximations were calculated using matrices of the form from (5.3.9). The L^2 norm of the derivative is ≈ 1. The L^2 error in V_0 is ≈ 0.13, and for V_{-1} is ≈ 0.013. We will usually make use of a resolution V_{-3} or finer.



(a) $B(x, y)$ for $\nu = 0$ (b) $B(x, y)$ for $\nu = 10^{-4}$

Figure 36: Boundary function plots for the Rossby wave problem. It is only the section of the function on $y \in (0, 10)$ that directly affects the time step.

(8.2.3). A plot for $P_2(x, y)$ for $\nu = 0$ can be found in Figure 36 a). When $\nu \neq 0$, P_2 becomes more complicated, the values a have to be complex and a third a is needed, so that Z_t, P and Z all match the necessary values at the boundary $y = 10$. A plot of $P_2(x, y)$ when $\nu = 10^{-4}$ can be found in Figure 36. Example C.6.1 has programs for calculating P_2, for $\nu = 0$ and $\nu \neq 0$.

The whole point of $\widehat{P}_2$ is for calculating the $\widehat{P}_2$ related component of $\frac{\partial}{\partial t}\widehat{P}(y, t, \kappa)$ from Equation (8.1.6), equal to

$$\left[-\kappa i\,\overline{u}(y)\,\left(\frac{\partial^2}{\partial y^2} - \kappa^2\right) - \kappa i\,\left[\beta - \overline{u}''(y)\right] + \nu\,\left(\frac{\partial^2}{\partial y^2} - \kappa^2\right)^2\right]\widehat{P}_2(y, \kappa), \qquad (8.2.12)$$

and taking its V_j decomposition. This vector, which we will call $\mathbf{B}_C$, will then be included in the eventual time step equation, though not initially. Recall that initial conditions are zero, with $P_1(x, y, 0) = Z_1(x, y, 0) = 0$. Including $\mathbf{B}_C$ at the beginning, at full value, creates a certain level of instability, and actually contradicts the initial conditions (recall that P_2 extends into $y \in (0, 10)$. We are primarily concerned with steady states for the linear system, and the non-linear system is unstable enough already. As a result, following the lead of [7], we will multiply the $\mathbf{B}_C$ vector by what we will call a 'switch-on' function, $f(t) = \min\{\frac{t}{t_1}, 1\}$, usually with $t_1 = 5$. As a result, it is fair to say that the initial conditions are zero, but with P_2, plotted in Figure 36, being gradually forced in.

We will keep our basic time stepping scheme, the second order implicit Adam's

method, found in (6.3.5). As before, we will have a set of vectors $\mathbf{Z}^n$, each representing our approximation of $Z(x, y, t)$ at times $t = nh$ ($n \in \mathbb{Z}$). There are a few matrices that are needed.

First we need operators T_{u_1} and T_{u_2} to handle the multiplication by $u(y)$ and $u''(y)$. These are most easily constructed using a pseudo-wavelet method (similar to pseudo-Fourier, see Glossary), by combining matrices that transform ψ and ϕ coefficient data into an equal number of equally spaced points (a decomposition matrix), then multiply by a diagonal matrix of $u(y)$ values, then multiply by the inverse of the decomposition matrix (look near (6.4.1)).

Next, we have derivative matrices, those that map our V_J approximations of the functions P and Z to approximations of the derivatives of those functions. We need a matrix M_Δ to approximate Δ, as well as M_x and, eventually, M_y for the first derivatives by x and y. We will also need M_Δ^{-1}, since our calculations will be in terms of Z and not P. There are quite a few means to create such matrices. The x direction is easy, with the $\frac{\partial^d}{\partial x^d}$ derivative being equal to multiplication by $(i\kappa)^d$ (with κ the Fourier coefficient). For the y direction we will use a collocation method (described near the end of Section 5.3). Basically, pointwise data for the derivative in question is then put in place of the actual values of ϕ in the decomposition matrix. Then, a regular inverse decomposition matrix is multiplied on the left. Again, see Section 5.3 for details.

All of this leads to the following time stepping scheme, if $\epsilon = 0$:

$$\begin{aligned}\mathbf{Z}^{n+1} &= \mathbf{Z}^n + \frac{h}{2}\left[-T_{u_1}M_x - \left(\beta I - T_{u_2}\right)M_x M_\Delta^{-1} + \nu M_\Delta\right]\left(\mathbf{Z}^n + \mathbf{Z}^{n+1}\right) + h\mathbf{B}_C \\ &= \mathbf{Z}^n + \frac{h}{2}M_{Der}\left(\mathbf{Z}^n + \mathbf{Z}^{n+1}\right) + h\mathbf{B}_C, \end{aligned} \tag{8.2.13}$$

by defining the new matrix M_{Der}, covering the entire (linear) component of the derivative expression. This simplifies to

$$\mathbf{Z}^{n+1} = \left(I - \frac{h}{2}M_{Der}\right)^{-1}\left[\left(I + \frac{h}{2}M_{Der}\right)\mathbf{Z}^n + h\,\mathbf{B}_C\right]. \tag{8.2.14}$$

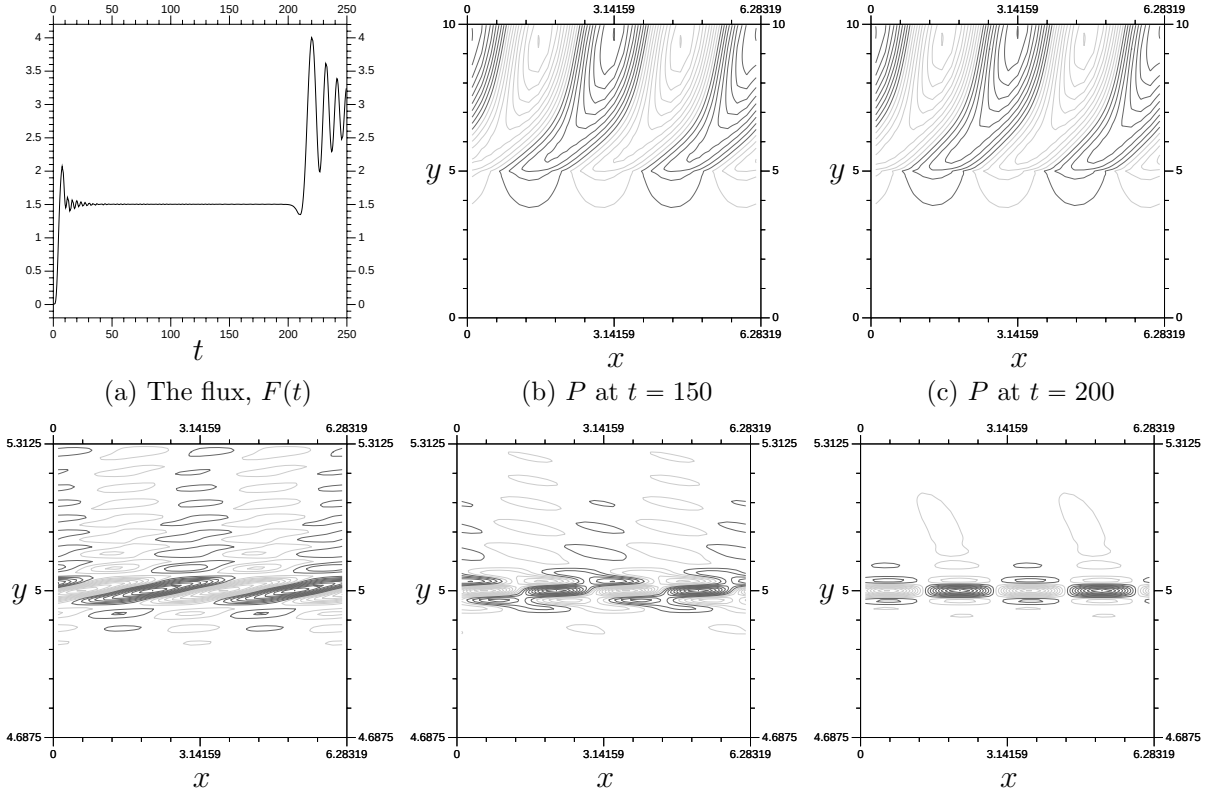
Example C.6.2 is a basic solver for the linear case.

8.3 Results

First we will look at the momentum flux (discussed near the end of Section 8.1). We calculate it for $y \in [2.5, 7.5]$, and the general behavior for the linear problem is well known. The flux is expected to rise from zero to some positive value, then reduce a (relatively) small amount and then stabilize. There will be some oscillation, but a very small amount, expected to reduce in amplitude as t increases. This behavior is shown in [2, Figure 5] and [7, Figure 3]. Notice that our results for V_{-6}, Figure 37(a), match this perfectly up to $t = 200$. Our results for V_{-5} (Figure 38(a)) do so until $t = 100$, then fail, and our results for V_{-4} (Figure 39(a)) until $t = 50$. Insufficient resolution is obviously the cause. If we include some viscosity, this effect is mitigated, as shown in Figure 40(a). Without viscosity, the higher the resolution, the longer the flux remains stable. The approximate time line is from $t = 0$ up to $25 \times 2^{-j-3}$ when using a V_j space for the y direction.

Recall the expected shape of the waves of P described in Section 8.1, and shown in Figure 31. That shape can be seen up to $t = 200$ in Figure 37(b,c). It can also be seen up to $t = 100$ in Figure 38(b,c) and up to $t = 50$ in Figure 39(b,c). So, the P are shaped as expected for the same time frame as the flux. The Z plots, however, show signs of failure earlier. Figures 37(f), 38(f) and 39(f) all show the same, anomalous, reflected pattern around $y = 5$, not the slowly sharpening wave seen in Figure 32.

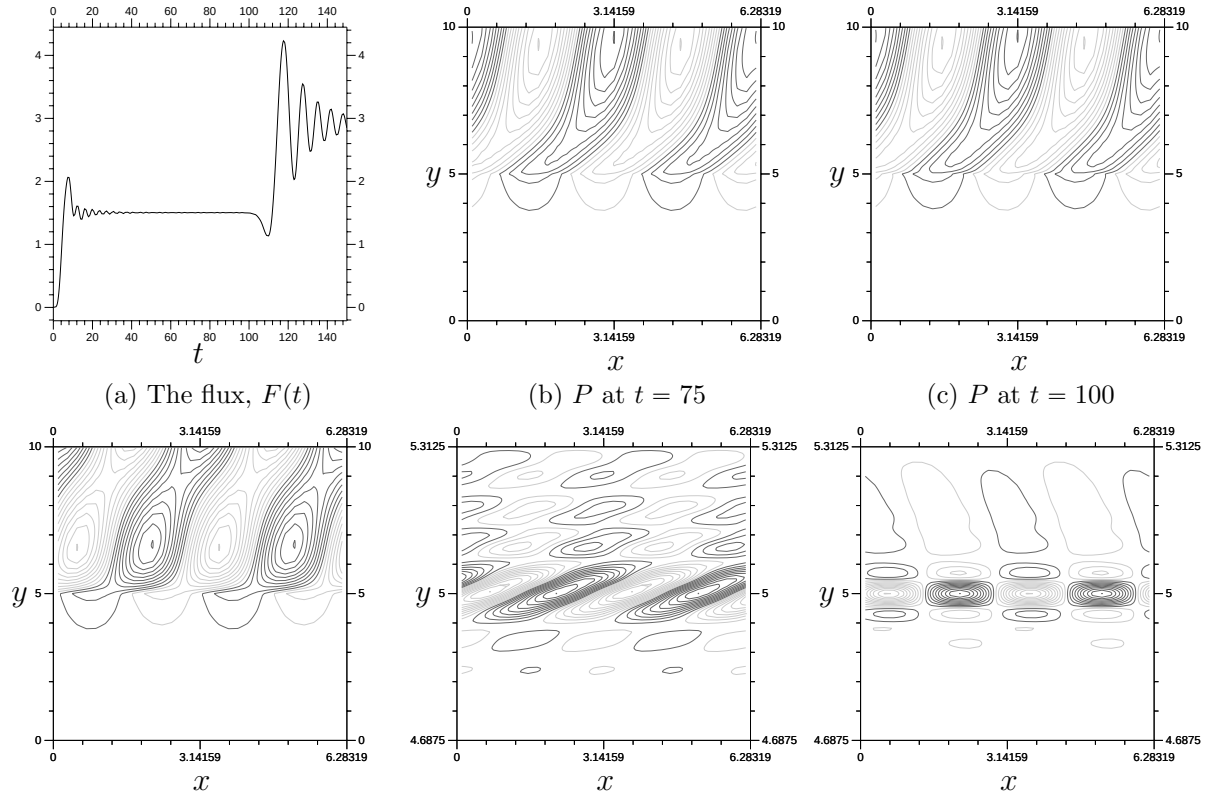
So now we should consider the full size of the systems. We need $10 \times 2^5 - 1$ terms to fully express V_{-5} on this domain. For the linear case, this means we need a total of $(10 \times 2^5 - 1) \times 2$ real elements, since the x direction will be a combination of $\sin(2x)$ and $\cos(2x)$ or a single complex value. However, if we switch to a non-linear system, the number of Fourier modes must increase dramatically, so we could, for instance, need to solve nonlinear (or merely linearized) systems of the size $(10 \times 2^5 - 1) \times 11 = 3509$. The level of raw effort involved is substantial, and some programs may simply balk at solving systems of that size. However, recall Section 8.1. The fact that $\overline{u}(5) = 0$ means that the solution's wavelengths will get very short near $y = 5$, so that is where we are likely to require fine resolution. Also, look at the plots of Z in Figures 37, 38 and 39. We have further confirmation that the $y = 5$ region is where the fine



(a) The flux, $F(t)$ (b) P at $t = 150$ (c) P at $t = 200$

(d) Z at $t = 100$ (e) Z at $t = 150$ (f) Z at $t = 200$

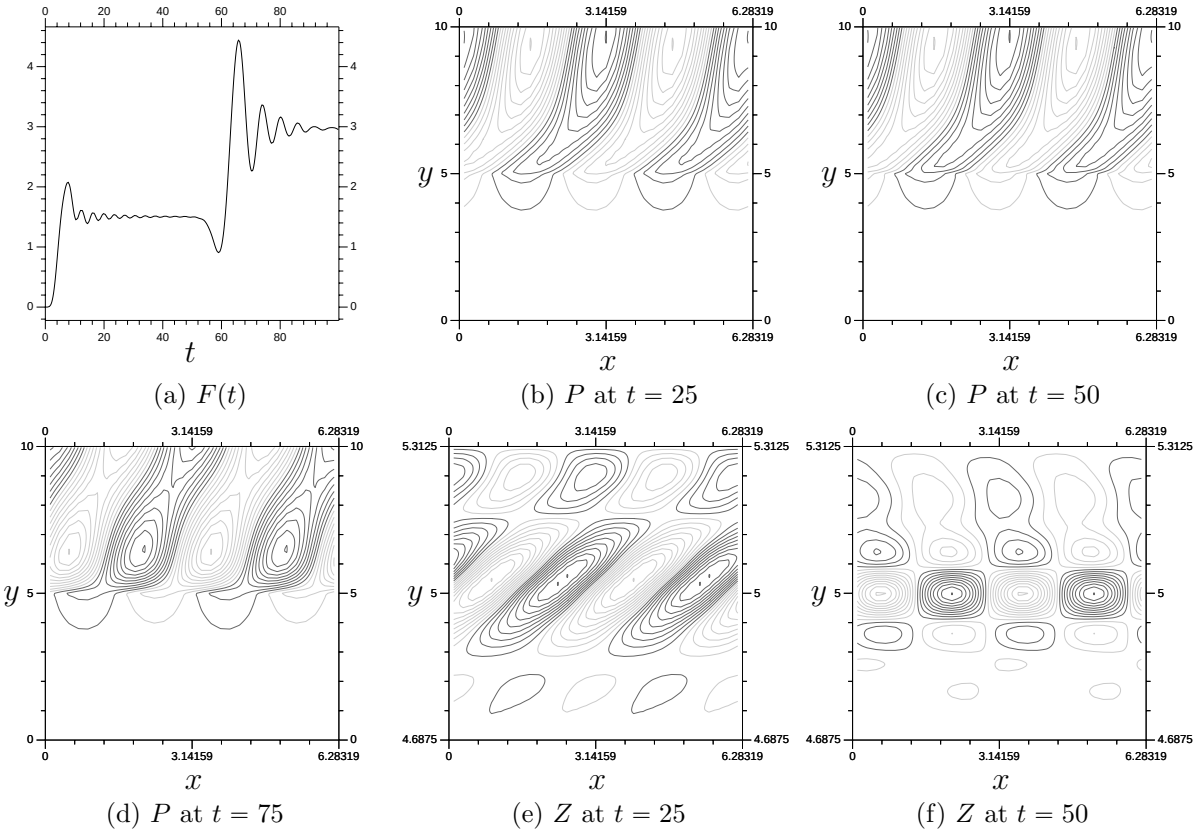
Figure 37: The results from a linear V_{-6} system. Notice that the flux remains stable until $t = 200$. These results us a 'switch-on' function for the boundary condition values equal to $\min\left\{\frac{t}{5}, 1\right\}$, $\beta = 1$, $\delta = 0.2$, and no viscosity.



(a) The flux, $F(t)$ (b) P at $t = 75$ (c) P at $t = 100$

(d) P at $t = 125$ (e) Z at $t = 50$ (f) Z at $t = 100$

Figure 38: Flux, P and Z values for V_{-5}, 2^5 resolution, no viscosity. The function P behaves itself until $t = 100$, where the Z is already showing problems. Other than resolution, these use the same values, etc, as Figure 37.



(a) $F(t)$ (b) P at $t = 25$ (c) P at $t = 50$

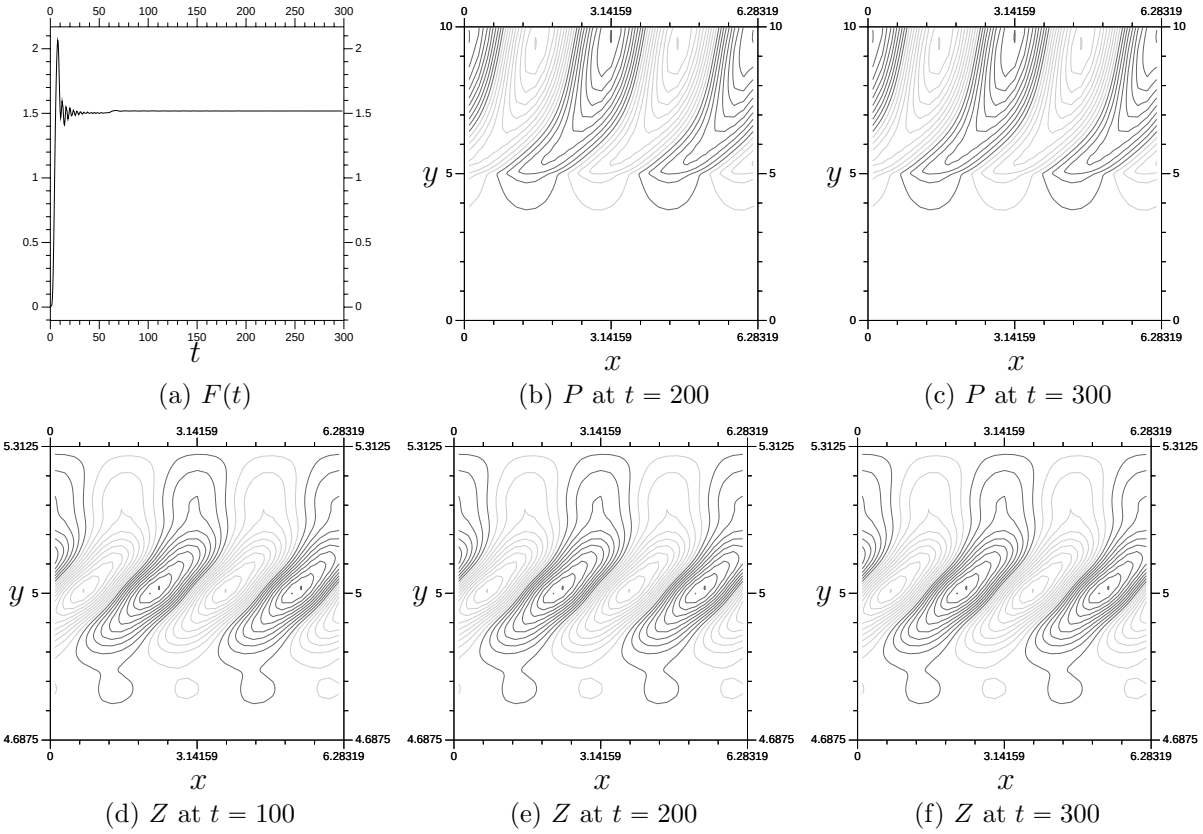
(d) P at $t = 75$ (e) Z at $t = 25$ (f) Z at $t = 50$

Figure 39: The results from a linear V_{-4} system. Contrast with Figure 37 to see the importance of the V_j space used. Everything except the resolution is kept the same as Figure 37 (and Figure 38).

resolution is required. In Figure 41 we have the value $\max_\kappa \log |b_{\kappa,-4,k}|$ against the center of the support of $\psi_{-4,k}(y)$, as well as $\max_\kappa \log |b_{\kappa,-3,k}|$ against the center of the support of $\psi_{-3,k}(y)$. Figure 41 effectively measures how important the W_{-4} and W_{-5} spaces are on different regions of $y \in (0, 10)$. Both spaces are most significant near $y = 5$, and W_{-5} in particular is small near the boundaries. So, theory and numerics agree that the problem will require finer resolution in the center. What we will do is apply the multi-scale method from Chapter 7.

8.4 Rossby Wave Multi-Scale Systems

Implementing a multi-scale system for the linear Rossby wave problem (Equation (8.1.6)) is actually less complicated than that for Burgers' equation (results in Section 7.3). The non-zero boundary conditions and additional physical dimension are more



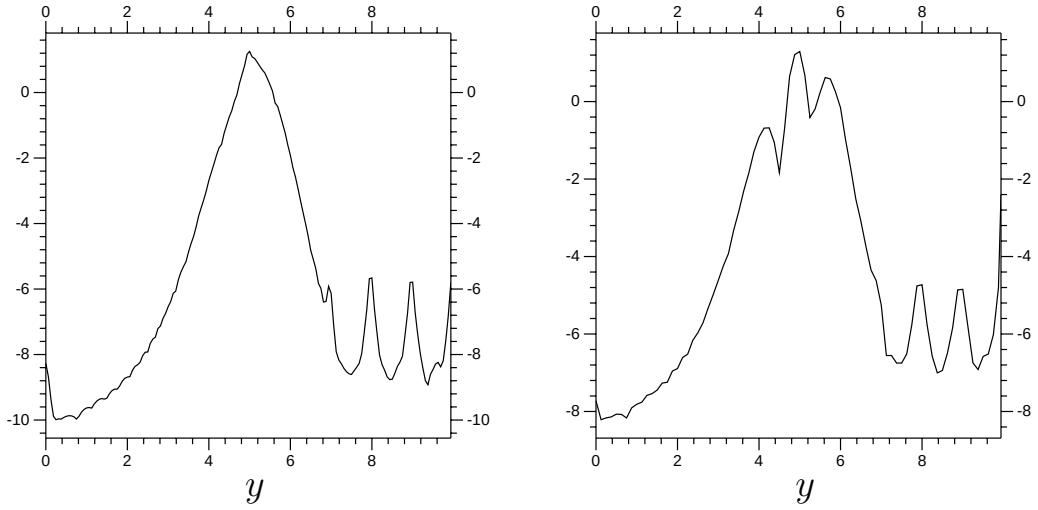
(a) $F(t)$ (b) P at $t = 200$ (c) P at $t = 300$

(d) Z at $t = 100$ (e) Z at $t = 200$ (f) Z at $t = 300$

Figure 40: V_{-4} results illustrating the effect of viscosity on the system. The viscosity is $\nu = 0.0001$. It is to be expected that the stability improves with the inclusion of viscosity. The result here is to make the problem more stable than the non-viscous V_{-6} problem. As usual, a 'switch-on' function was used on the boundary conditions, equal to $\min\{\frac{t}{5}, 1\}$. The largest absolute difference between the two P plots given is approximately 2.25×10^{-4}. The Z plots are similarly close.

than made up for by the problem being linear. This is particularly true since the problem needs only a single (complex) Fourier term for the x dimension (only $e^{\pm 2ix}$ are needed to express $P(x, y, t)$).

Our first set of examples is done in a manner intended to maximize the consistency between the large-scale and small-scale systems. As a result, our primary results are based on the approach outlined in Section 7.2. We choose V_{-4} for the large scale system, over $\Omega = (0, 10)$, and V_{-6} for the small-scale system, over $\Lambda = (3.75, 6.25)$. This means that M (Section 7.2, again) is calculated at full, V_{-6}, resolution on the entire domain Ω. The matrix M will be 1278×1278 in size, and creating it requires multiplying, adding and calculating the inverse of matrices of that size. This takes a



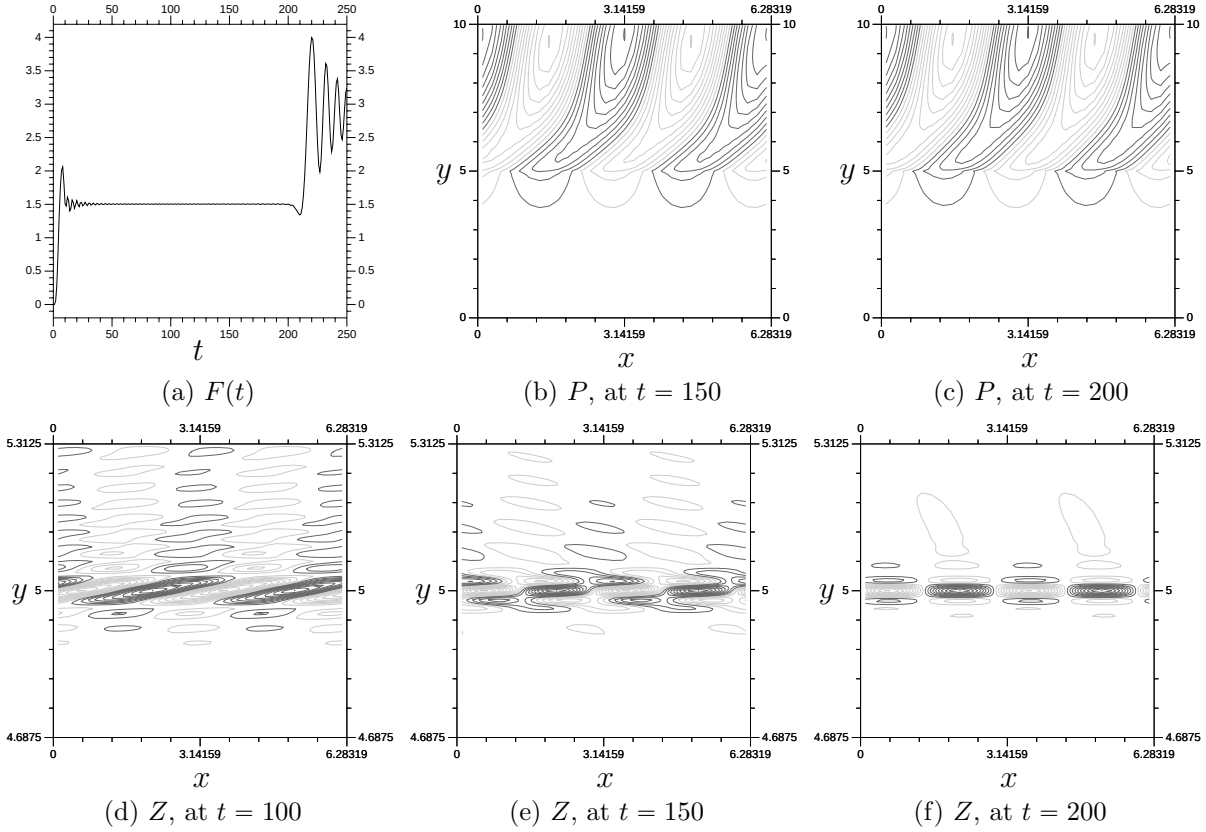
(a) $\max_\kappa \log|b_{\kappa,-4,k}|$ at $t = 50$ (b) $\max_\kappa \log|b_{\kappa,-3,k}|$ at $t = 50$

Figure 41: Plots that show the importance of some W_j spaces. The horizontal is the center of the support of $\psi_{-4,k}(y)$ and $\psi_{-3,k}(y)$, respectively. So, we are plotting $\max_\kappa \log|b_{\kappa,j,k}|$ (for $j = -3, -4$) against the physical location of $\psi_{j,k}$.

fair amount of computing effort, but the calculations only need to be done once, and the resulting accuracy makes it worth it. There are 3 'extra' terms on each side of Λ. The results are encouraging, showing stability of the flux from $t = 0$ to $t = 200$. Recall that this time frame is what we get with a V_{-6} resolution system. The P and Z plots match the V_{-6} results as well. The plots are found in Figure 42. The code used is Example C.6.3.

The next set of results will all be based on double systems that are created (their derivative matrices, and so on) separately. This possibility was discussed in Section 7.4, though discussion about its effects on linear terms begins around (7.4.5). First, an example of what happens when $E_\Lambda \neq D_\Lambda$ but the $(\mathbf{a}_\Lambda^{T_m} + \mathbf{a}_\Lambda^n)$ related term is kept to $\begin{bmatrix} B_\Lambda \\ O \end{bmatrix}$. We will, again, use V_{-4} on $\Omega = (0, 10)$ and V_{-6} on $\Lambda = (3.75, 6.25)$. Results are plotted in Figure 43. Obviously, this arrangement does not work at all. The flux is only stable to $t = 50$ and the P plots do not fit our desired results (those from Figure 37). We can improve things, a little, by including some of the features discussed in Section 7.4. We use the same problem, resolutions, and domains, but include 3 'extra' terms and a multiplier J with

$$J_{i,i} = \quad 0,\, 0,\, 0,\, 0,\, \frac{1}{3},\, \frac{2}{3},\, 1,\, 1,\, \ldots,\, 1,\, 1,\, \frac{2}{3},\, \frac{1}{3},\, 0,\, 0,\, 0,\, 0. \tag{8.4.1}$$

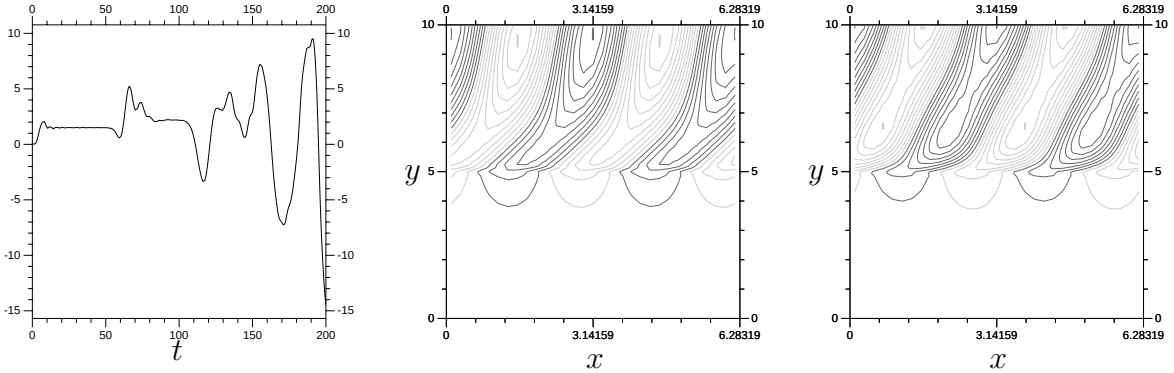


(a) $F(t)$ (b) P, at $t = 150$ (c) P, at $t = 200$

(d) Z, at $t = 100$ (e) Z, at $t = 150$ (f) Z, at $t = 200$

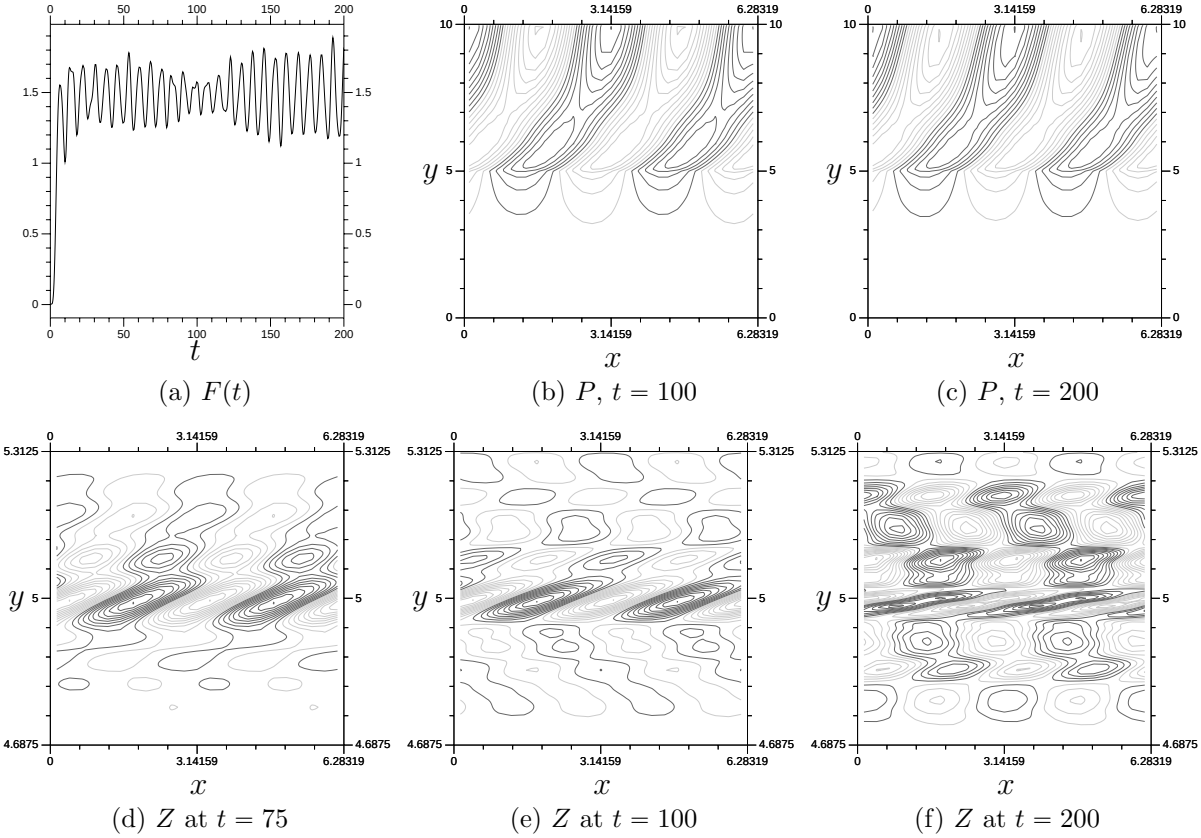
Figure 42: Results from a V_{-4} with localized V_{-6} double system. The small-scale is over $\Lambda = (3.0, 6.5)$, and 3 'extra' terms are used over both interfaces of Λ. The derivative matrices were made by taking apart a full domain V_6 system, to maximize the consistency. There is a little instability in the flux around $t = 200$, but this arrangement is very close to the full V_6 system (with the small-scale resolution over all of Ω), found in Figure 37. This double sized arrangement has $159 + 165$ terms per Fourier mode, and the two systems are solved separately. The full V_{-6} system has 639 elements per Fourier mode, and the system has to be solved all at once.

The results are plotted in Figure 44. They are better than the previous set (Figure 43), but only marginally. There are distortions in the P plots, and the flux is not as smooth as it should be. So, on the whole, we should use the Section 7.2 approach to creating the matrices.



(a) $F(t)$ (b) P, $t = 50$ (c) P, $t = 100$

Figure 43: An effect from not maintaining consistency. This can be what happens when the large and small-scale systems use inconsistent matrices. These are from a $2^4 : 2^2$ system, V_{-4} everywhere and V_{-6} everywhere, with no viscosity. As before, a 'switch-on' function with $t_1 = 5$ is used. In the terminology of Section 7.4, these results use $E_\Lambda \neq D_\Lambda$, with none of the related fixes.



(a) $F(t)$ (b) P, $t = 100$ (c) P, $t = 200$

(d) Z at $t = 75$ (e) Z at $t = 100$ (f) Z at $t = 200$

Figure 44: More effects of not maintaining consistency, mitigated somewhat. These results use a V_{-4} system everywhere and a V_{-6} system on $\Lambda = (3.75, 6.25)$. Also, three 'extra' ϕ terms are used on either side of Λ, with a J multiplier given in (8.4.1). Both of those improvements are discussed in 7.4. As always, there is a 'switch-on' applied to the boundary conditions, with $t_1 = 5$. These are better than those in Figure 43, but do not fully match the proper values (found in Figure 37).

8.5 Non-Linear Versions

The actual benefits of the multi-scale system exist primarily for non-linear problems, those whose time steps require significant computational effort to solve. So, we will switch to looking into some cases where $\epsilon \neq 0$. The same method that was used to linearize the time stepping scheme for Burgers' equation (Section 6.6) will be used again. This time there are two separate terms, $P_y Z_x$ and $P_x Z_y$, making the resulting operator more complicated, but the arrangement is much the same.

There is one more, significant, issue: we will need many more Fourier terms to properly express the x direction when $\epsilon \neq 0$. As was discussed in Section 8.1, the linear problem needs only two Fourier functions, e^{2ix} and e^{-2ix}. Since Z and P are real, we have the added bonus that the coefficient for e^{2ix} would have to be the complex conjugate of the coefficient for e^{-2ix}, meaning we needed only to keep track of one complex term (equivalent to two real terms) for every value y. If $\epsilon \neq 0$ then the $P_x Z_y$ and $P_y Z_x$ terms contribute to Z_t. If Z has non-zero $\widehat{Z}(2, y, t)$ and $\widehat{Z}(2, y, t)$, so non-zero coefficients related to e^{2ix} and e^{-2ix}, then Z_t will almost certainly have non-zero coefficients related to $e^{\pm 4ix}$, e^0 as well as the $e^{\pm 2ix}$. Considering that the boundary conditions are represented entirely by $e^{\pm 2ix}$ functions and that the initial conditions are zero, this means that the Fourier modes we will have to keep track of are $\kappa = 2k$, $k \in \mathbb{Z}$. All the other modes, the odd numbered modes, will have $\widehat{Z}(y, t, \kappa) = 0$. Obviously, we have to restrict the number of Fourier modes we calculate to a finite number, and use a pseudo Fourier scheme (see glossary) to approximate $P_x Z_y$ and $P_y Z_x$. To get an idea of how many Fourier coefficients we will need per y value, assume we want to use Fourier modes $\kappa \in [-2K, 2K]$ ($K \in \mathbb{N}$). This arrangement means we have $2K + 1$ real values worth of Fourier coefficients for every y value. The $\kappa = 0$ will have to be real, so one real value. Next, there will be one complex value for $\kappa = 2, 4, \ldots 2K$, the equivalent of $2K$ real values, so $2K + 1$ total. Since the coefficients for $\kappa = -2, -4, \ldots - 2K$ are the complex conjugates of those for $\kappa = 2, 4, \ldots 2K$, they do not need to be calculated.

To sum up the difficulties, we have to create and solve a new linear system at every step, and the systems themselves will be significantly larger. As a result, we

will keep the y resolution down to a manageable V_{-4}, or 16 elements per unit, 159 total. Combined with 11 elements for the x direction and that makes 1749 real values total. The boundary conditions are the same, and $\beta = 1$, $\delta = 0.2$ as before. We will include some viscosity, $\nu = 0.0001$, to stabilize the system and make the V_{-4} resolution plausible. Recall that V_{-4} had Z noticeably wrong at $t = 50$ without the addition of viscosity (see Figures 39, 40). The only parameter we will change will be ϵ. The size of the time step will be $h = 0.05$.

Our multi-scale results will use the method that was applied to Burgers' equation in Section 7.4. We will use 3 'extra' terms and a J multiplier with

$$J_{i,i} = 0,\ 0,\ 0,\ 0,\ \frac{1}{2},\ 1,\ 1,\ \ldots\ ,\ 1,\ 1,\ \frac{1}{2},\ 0,\ 0,\ 0,\ 0.$$

The linear components of the calculation will be derived the same way as in Section 7.4 as well, from a system using the finest resolution over all Ω.

As discussed at the end of Section 8.1, the flux for the non-linear problem is expected to reach its maximum early, then reduce down to zero and begin oscillating around zero. As for P, it is expected to develop waves similar to the linear problem, at least initially. As t increases, we can expect the tip of the waves in P, the parts near $y = 5$, to break from the rest and create separate, somewhat circular, shapes. This behavior is shown nicely in [7, Figure 2(b) and 14(b)]. Other than the expectation of greater complexity, we will say little of Z. We will, however, use Z to check the accuracy of the multi-scale results.

First, we will look at $\epsilon = 0.01$. The flux is expected to quickly reach the linear system's steady state then slowly reduce to zero and begin oscillating around zero. Take a look at Figures 45 and 46. Our current value of ϵ is not high enough for the flux to reach zero during $t \in [0, 50]$, so instead we see a reduction towards zero for the V_{-4} model. We will set V_{-4} as our top resolution, the results we will want to duplicate efficiently with the multi-scale method. Our 'base' resolution is V_{-3}, a resolution that fails spectacularly at delivering the expected results, with the flux going up rather than down to zero after $t = 30$. Again, see Figure 46 for the contrast between the two resolutions.

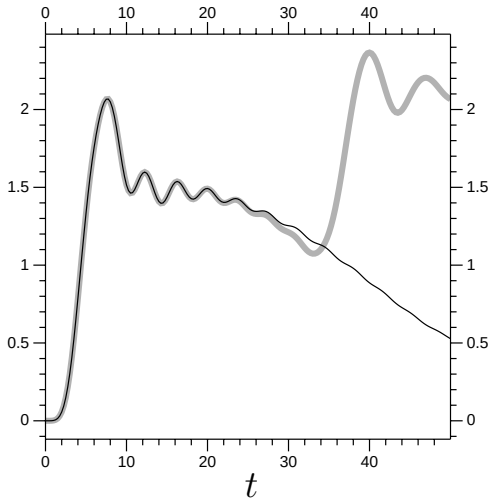
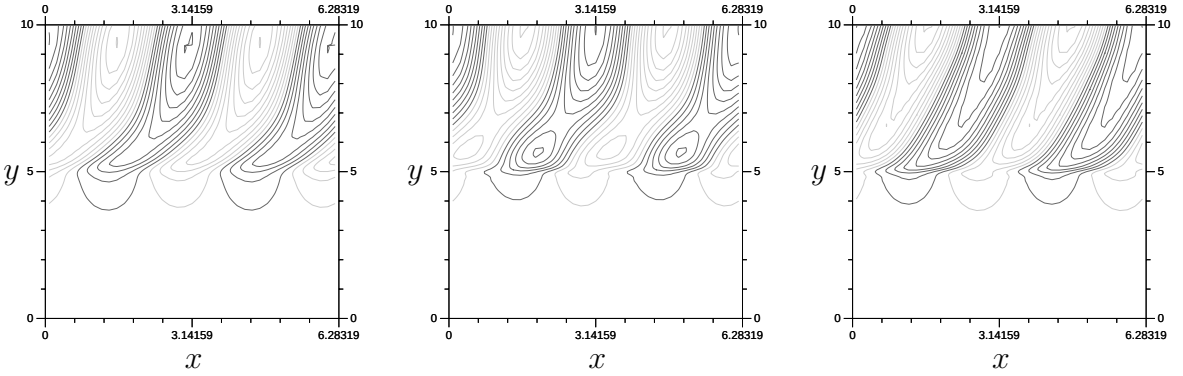


Figure 45: Flux, $F(t)$, for $\epsilon = 0.01$, for V_{-3} and V_{-4}. The lighter, wider, line uses V_{-3}, heading up to 2 at 50. The other uses V_{-4}, and (correctly) approaches zero.

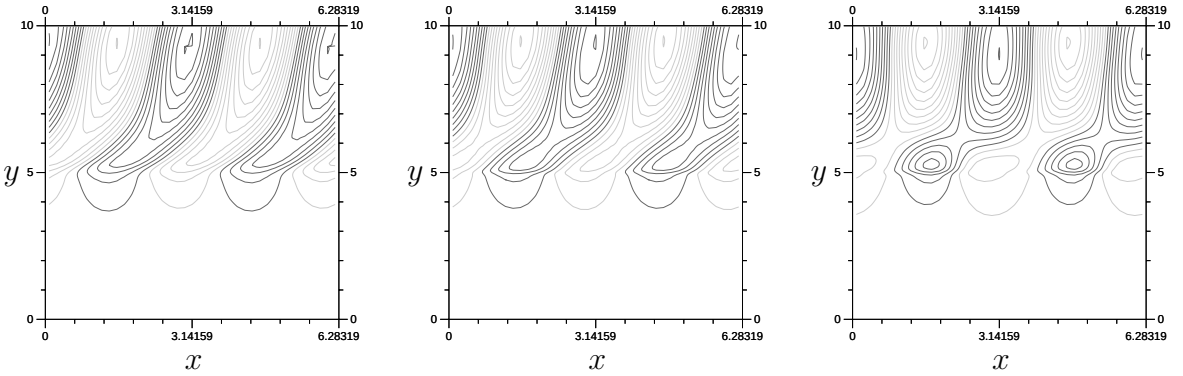
Now to discuss how this particular Rossby wave problem responds to the multi-scale method. While the advantages in efficiency are not quite as great as for Burgers' equation, they are substantial. The problem does seems to require a larger domain for the small-scale system, which cuts into the efficiency. First, note that the V_{-3} system took an average of about 1.32 seconds per time step. The V_{-4} system took an average of about 9.55 seconds per time step. See Figure 47. The $\Lambda = (3.5, 8.5)$ double system requires approximately 3.51 seconds per time step, a significant improvement on the V_{-4} system. However, the results are not quite as close to the V_{-4} system as we would like. Including more elements above $y = 8.5$ improves things, as shown in the next results. However, the $\Lambda = (3.5, 9.0)$ model requires 3.90 seconds per time step, and $\Lambda = (3.5, 9.5)$ requires 4.48 seconds per time step.

Now we take a look to solving the $\epsilon = 0.05$ problem. We can expect a greater deviation from the linear problem, as well as extra instability. The general pattern for the flux should be about the same, though it should evolve faster in time. The flux should increase to the steady state of the linear system then reduce and start oscillating around zero. Results can be found in Figures 48 and 49. The V_{-3} flux plots seem to be oscillating around $\frac{1}{2}$, almost going as high as the linear problem's steady state. The V_{-4} results are, as expected, consistent with previous results. We see the oscillation beginning at $t = 25$.



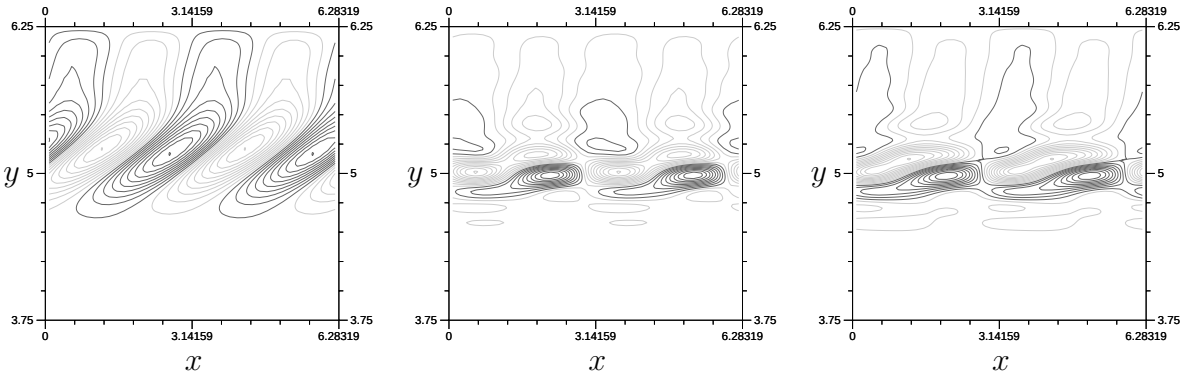
(a) P, $t = 10$ (b) P, $t = 30$ (c) P, $t = 50$

P plots for the V_{-3} setup.



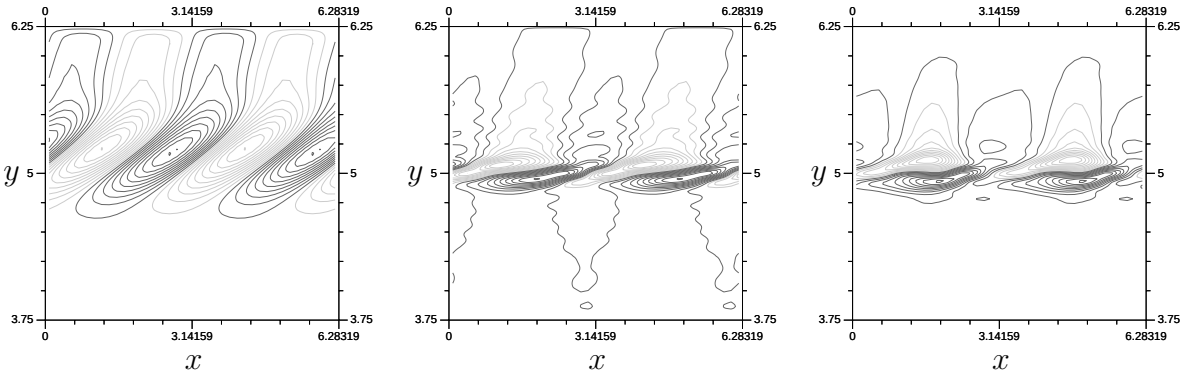
(d) P, $t = 10$ (e) P, $t = 30$ (f) P, $t = 50$

P plots for the V_{-4} setup.



(g) Z, $t = 10$ (h) Z, $t = 30$ (i) Z, $t = 50$

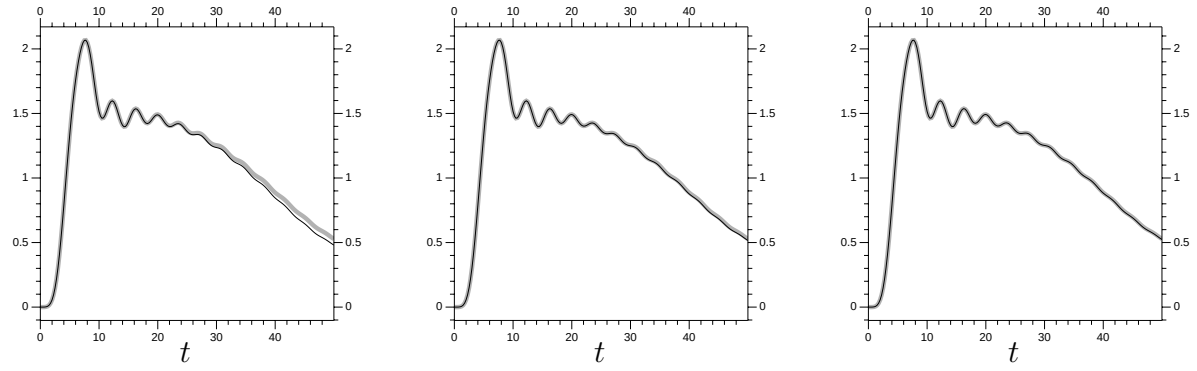
Z plots for the V_{-3} setup.



(j) Z, $t = 10$ (k) Z, $t = 30$ (l) Z, $t = 50$

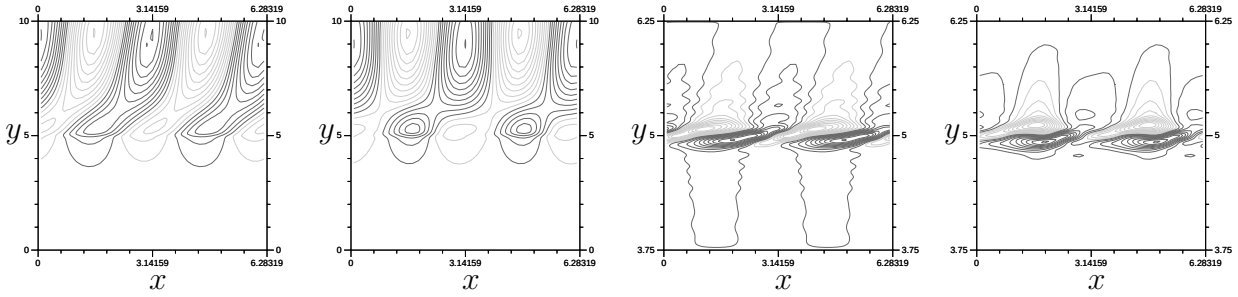
Z plots for the V_{-4} setup.

Figure 46: Plots for the $\epsilon = 0.01$ problem using V_{-3} and V_{-4}.



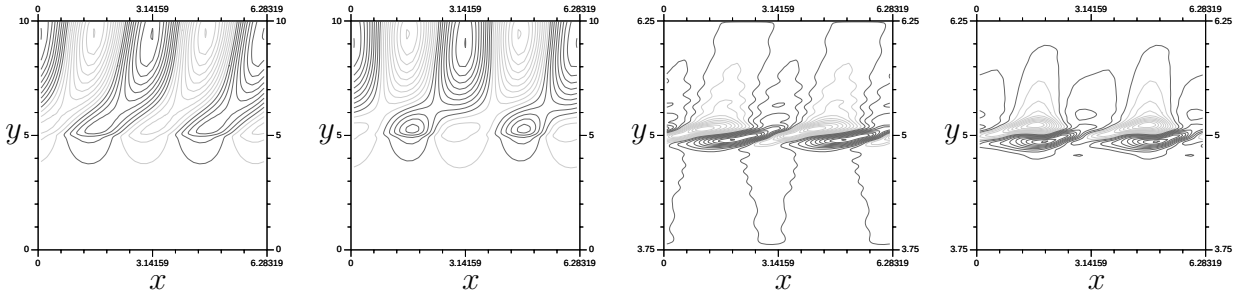
(a) $\Lambda = (1.5, 6.5)$ (b) $\Lambda = (1.0, 6.5)$ (c) $\Lambda = (0.5, 6.5)$

The flux, $F(t)$, for $\epsilon = 0.01$. Each plot has $V_{-3} : V_{-4}$ flux results compared to those from a full V_{-4} system. The lighter, wider, line is the full V_{-4}.



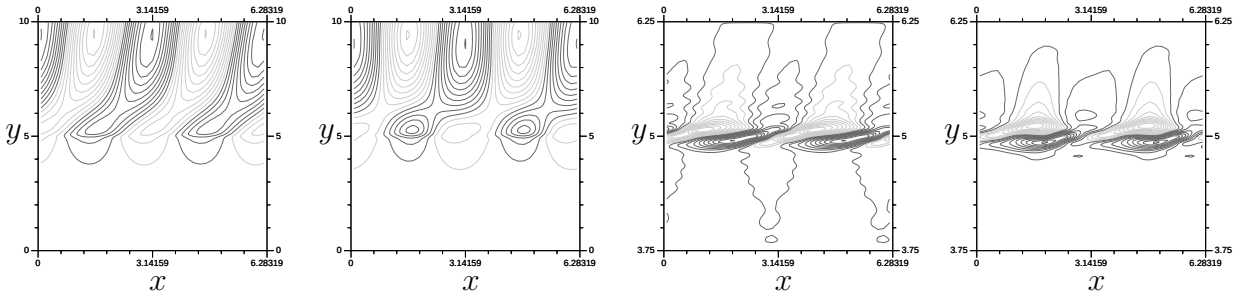
(d) P at $t = 30$ (e) P at $t = 50$ (f) P at $t = 30$ (g) Z at $t = 50$

Results for the $V_{-3} : V_{-4}$ multi-scale model with $\Lambda = (1.5, 6.5)$.



(h) P at $t = 30$ (i) P at $t = 50$ (j) P at $t = 30$ (k) Z at $t = 50$

Results for the $V_{-3} : V_{-4}$ multi-scale model with $\Lambda = (1.0, 6.5)$.



(l) P at $t = 30$ (m) P at $t = 50$ (n) P at $t = 30$ (o) Z at $t = 50$

Results for the $V_{-3} : V_{-4}$ multi-scale model with $\Lambda = (0.5, 6.5)$.

Figure 47: Plots for $V_{-3} : V_{-4}$ multi-scale models of the $\epsilon = 0.01$ Rossby wave problem. Note that we start at $t = 30$ because the different models are very similar for low t.

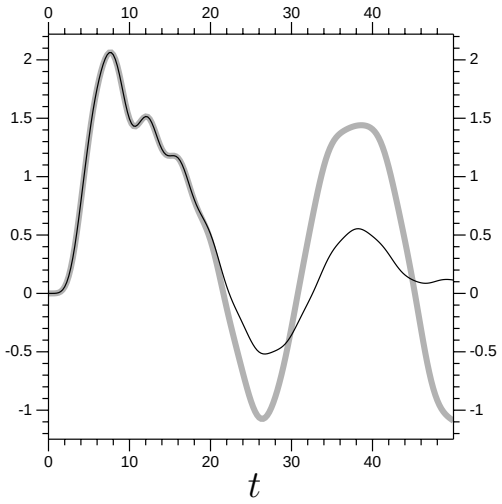
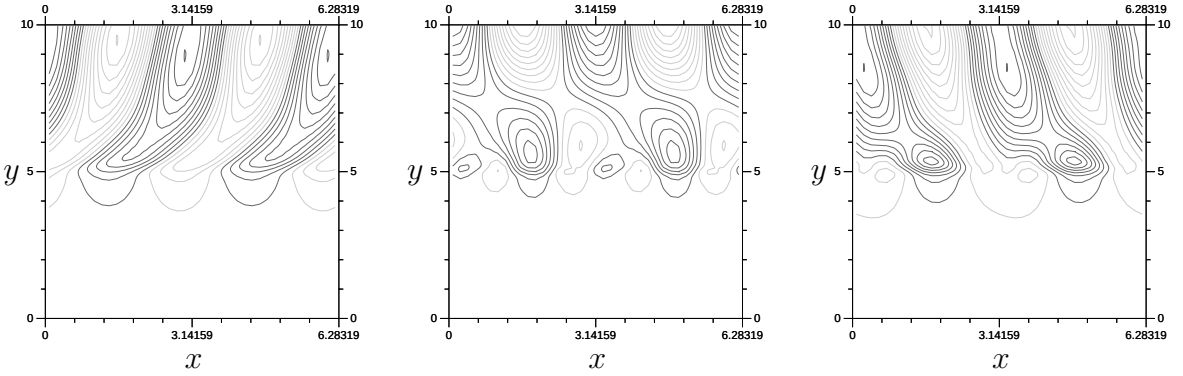


Figure 48: Flux, $F(t)$, for $\epsilon = 0.05$. The lighter, wider, line uses V_{-3}, the other uses V_{-4}. While the more widely oscillating V_{-3} line may look like what we are after, notice that it almost reaches the height of the steady state of the linear system.

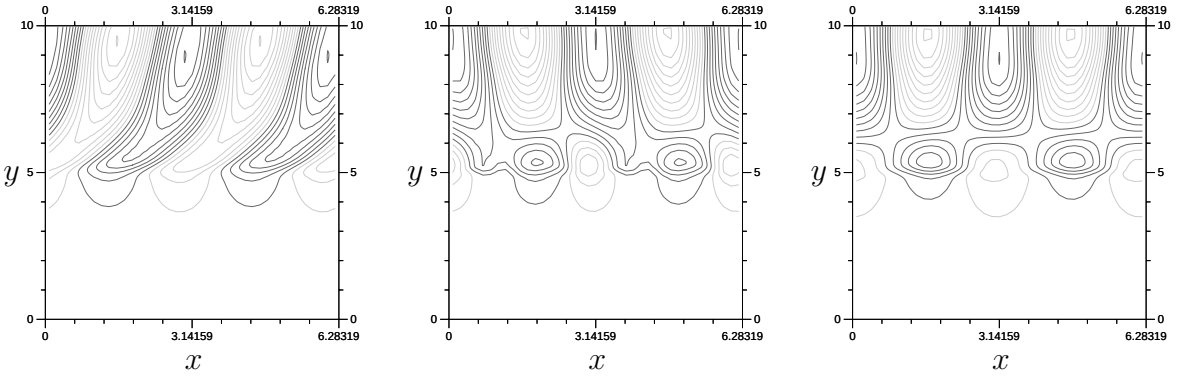
Duplicating the V_{-4} results using multi-scale arrangements is the next task. Results are in Figure 50. Again, fairly large domains were required, which cut into the efficiency benefits. The V_{-3} time steps took an average of 1.30 seconds, the V_{-4} time steps took 9.24. The double systems for $\Lambda = (3.5, 8.5), (3.5, 9.0)$ and $(3.5, 9.5)$ took 3.39, 3.89 and 4.47 seconds respectively. Again, the results are very close, and flux values almost perfect, for the larger Λ domains. The P and Z plots from Figure 49 are basically indistinguishable from those in Figure 50 from the larger Λ domains.

The multi-scale method, while not as perfectly suited to the non-linear Rossby problem as it was to the Burger's equation problem, did perform well. The multi-scale results were shown to be able to duplicate the V_{-4} results using only localized V_{-4} resolution. There was also a significant reduction in the effort, with the time step taking half as long. The method's performance can actually be viewed as better than it first appears. The resolution used was restricted to a coarse V_{-4} to keep computing requirements reasonable. The increase from V_{-3} to V_{-4} means that the small-scale system has approximately half of its coefficients be in V_{-3}, so coefficients that are also in the large-scale system. Simply put, having only a single W_j space in the small-scale system is the least efficient form of the multi-scale method, and it still proved beneficial with the non-linear Rossby wave problem. Table 9 has a summary of the time necessary for calculating the single and multi-scale systems (average computing



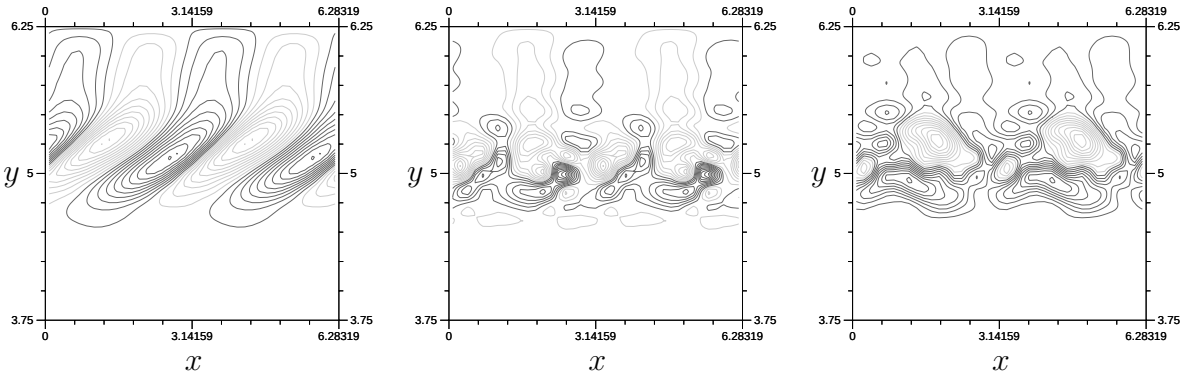
(a) P, $t = 10$ (b) P, $t = 30$ (c) P, $t = 50$

P plots for the V_{-3} setup.



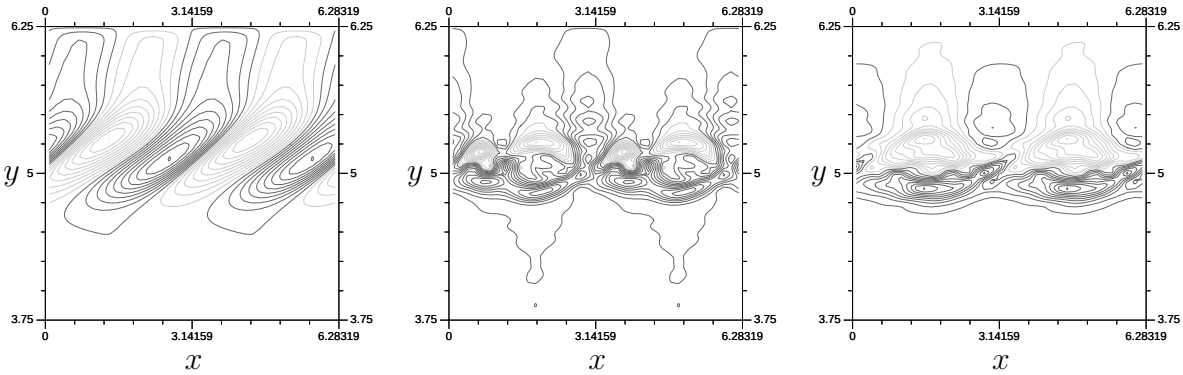
(d) P, $t = 10$ (e) P, $t = 30$ (f) P, $t = 50$

P plots for the V_{-4} setup.



(g) Z, $t = 10$ (h) Z, $t = 30$ (i) Z, $t = 50$

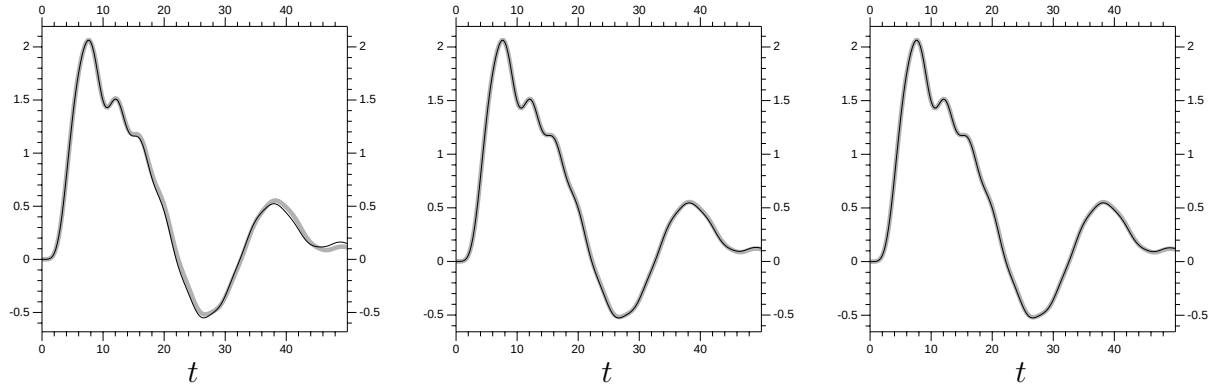
Z plots for the V_{-3} setup.



(j) Z, $t = 10$ (k) Z, $t = 30$ (l) Z, $t = 50$

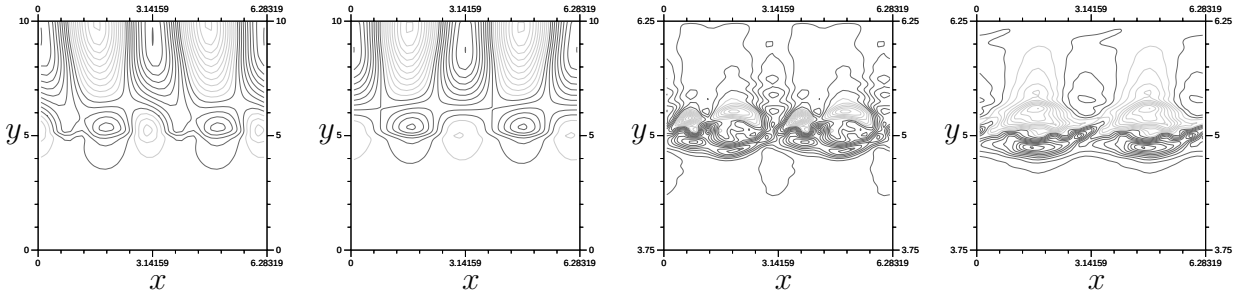
Z plots for the V_{-4} setup.

Figure 49: Plots for the $\epsilon = 0.05$ problem using V_{-3} and V_{-4}.



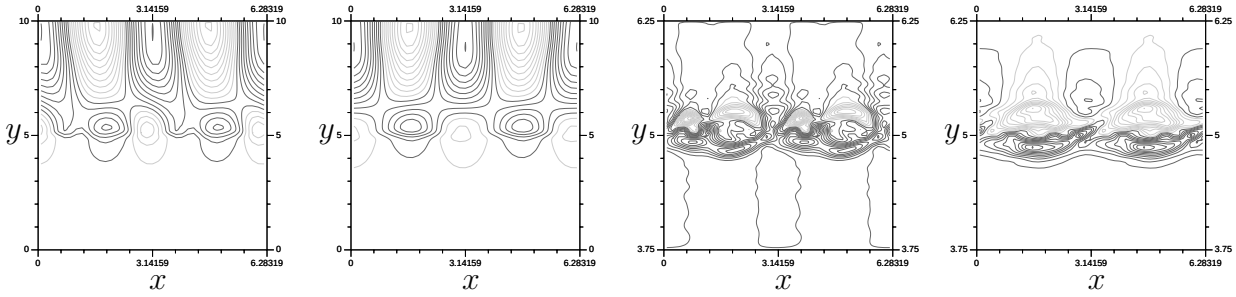
(a) $\Lambda = (3.5, 8.5)$ (b) $\Lambda = (3.5, 9.0)$ (c) $\Lambda = (3.5, 9.5)$

Flux, $F(t)$, for $\epsilon = 0.05$. Each plots the $V_{-3} : V_{-4}$ flux against the full V_{-4} flux. The lighter, wider, line is the full V_{-4}.



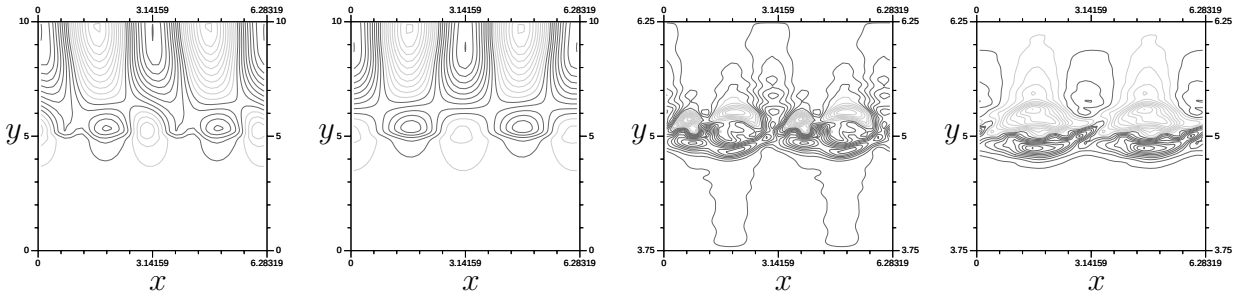
(d) P at $t = 30$ (e) P at $t = 50$ (f) Z at $t = 30$ (g) Z at $t = 50$

Results for the $V_{-3} : V_{-4}$ multi-scale model with $\Lambda = (3.5, 8.5)$.



(h) P at $t = 30$ (i) P at $t = 50$ (j) Z at $t = 30$ (k) Z at $t = 50$

Results for the $V_{-3} : V_{-4}$ multi-scale model with $\Lambda = (3.5, 9.0)$.



(l) P at $t = 30$ (m) P at $t = 50$ (n) Z at $t = 30$ (o) Z at $t = 50$

Results for the $V_{-3} : V_{-4}$ multi-scale model with $\Lambda = (3.5, 9.5)$.

Figure 50: Plots for $V_{-3} : V_{-4}$ multi-scale models of the $\epsilon = 0.05$ Rossby wave problem. Again, we start at $t = 30$ because the different models are very similar for low t.

time per time step). Notice that the nearly identical $\Lambda = (3.5, 9.5)$ results were calculated in half the time.

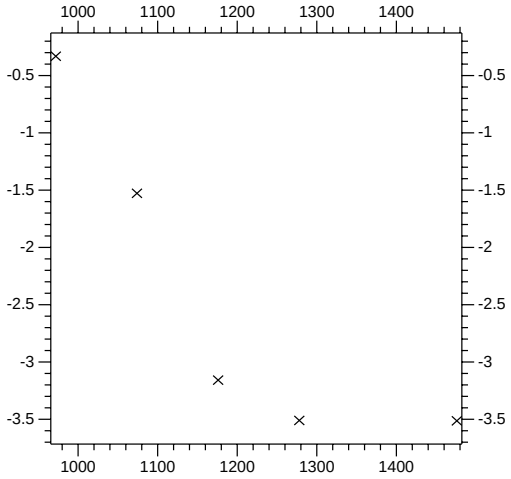
Table 9: Comparison between computing times of different resolutions and Λ domains.

Average Computing Time Per Time Step, in seconds		
System Used	Problem Used	
	$\epsilon = 0.01$	$\epsilon = 0.05$
Single System, V_{-3}	1.32	1.30
Single System, V_{-4}	9.55	9.24
Multi-Scale, V_{-4} on $\Lambda = (3.5, 8.5)$	3.51	3.39
Multi-Scale, V_{-4} on $\Lambda = (3.5, 9.0)$	3.90	3.89
Multi-Scale, V_{-4} on $\Lambda = (3.5, 9.5)$	4.48	4.47

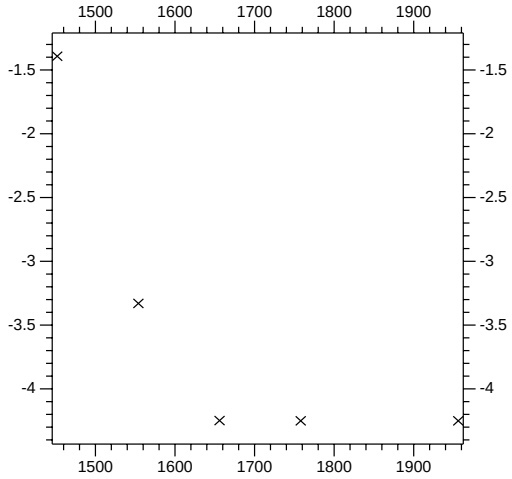
8.6 Convergence

It is somewhat harder to find a sign of convergence with the linear Rossby wave problem than with Burgers' equation. Recall that the initial conditions are zero, while the $y = 10$ boundary is equal to $\cos(2x)$. Any lack of resolution for the boundary function $P_2(x, y)$ (see Section 8.1) will affect the results substantially. Basically, there will be a limit to the accuracy of multi-scale system results (unless a second small-scale system is created at $y = 10$). As with the convergence related results in Section 7.5, we will look at the effect of different Λ sizes and 'extra' terms on the accuracy of multi-scale results. The control results will be a V_{-6} resolution single system. The test systems will use V_{-4} or V_{-5} over Ω and V_{-6} on Λ. Plots of the $\log_{10}$ transformed error can be found in Figure 51. The summarized data is in Table 10.

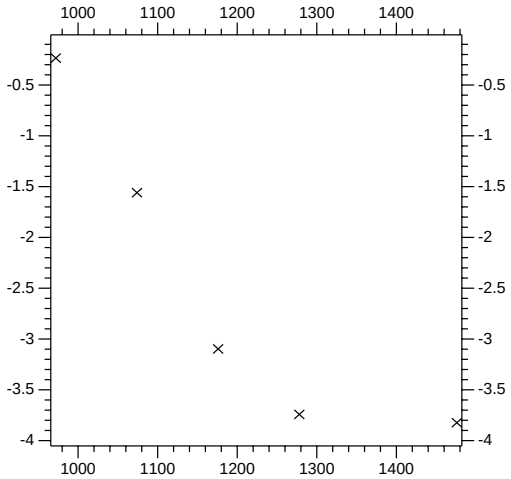
Now we take a look at the convergence from the non-linear results in Section 8.5. Recall that the behavior of the flux showed a certain amount of convergence. Since the behavior of the solution is dominated by the influence of the boundary conditions, we cannot expect a particularly accurate approximation without including the $y = 10$ boundary in a small-scale system. Computational restrictions keep the large-scale system to a resolution of V_{-3}, which causes a fair amount of error. We cannot expect



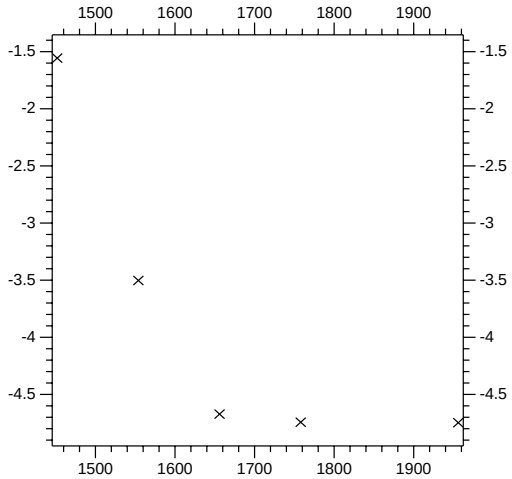
(a) L^2 error for V_{-4} with localized V_{-6}



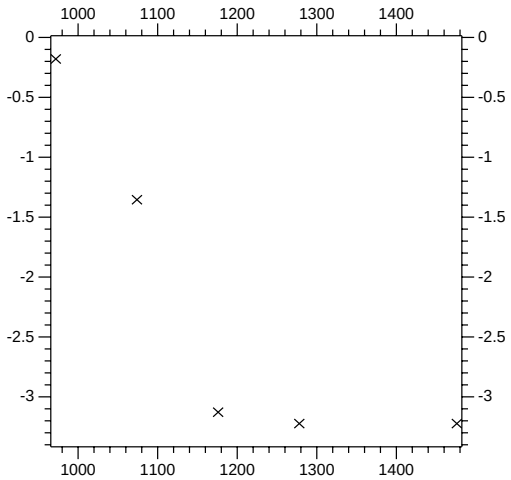
(b) L^2 error for V_{-5} with localized V_{-6}



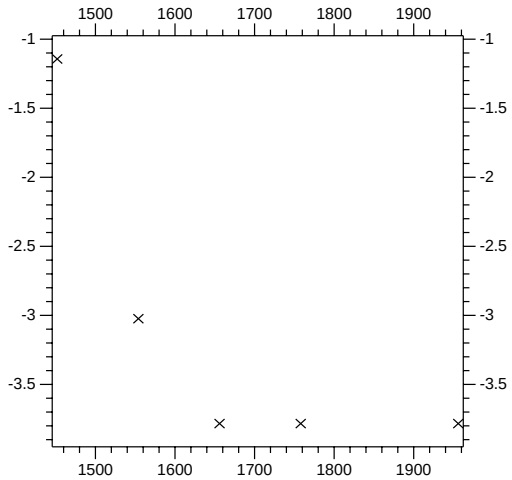
(c) L^1 error for V_{-4} with localized V_{-6}



(d) L^1 error for V_{-5} with localized V_{-6}



(e) L^∞ error for V_{-4} with localized V_{-6}



(f) L^∞ error for V_{-5} with localized V_{-6}

Figure 51: The error at $t = 200$ of the function Z from a multi-scale system, versus the size of the combined systems. Both are compared to a basic V_{-6} system. The $\log_{10}$ of the error is the vertical axis, the horizontal is the number of coefficients $a_{j,k}$ and $b_{j,k}$ used in the systems, all together.

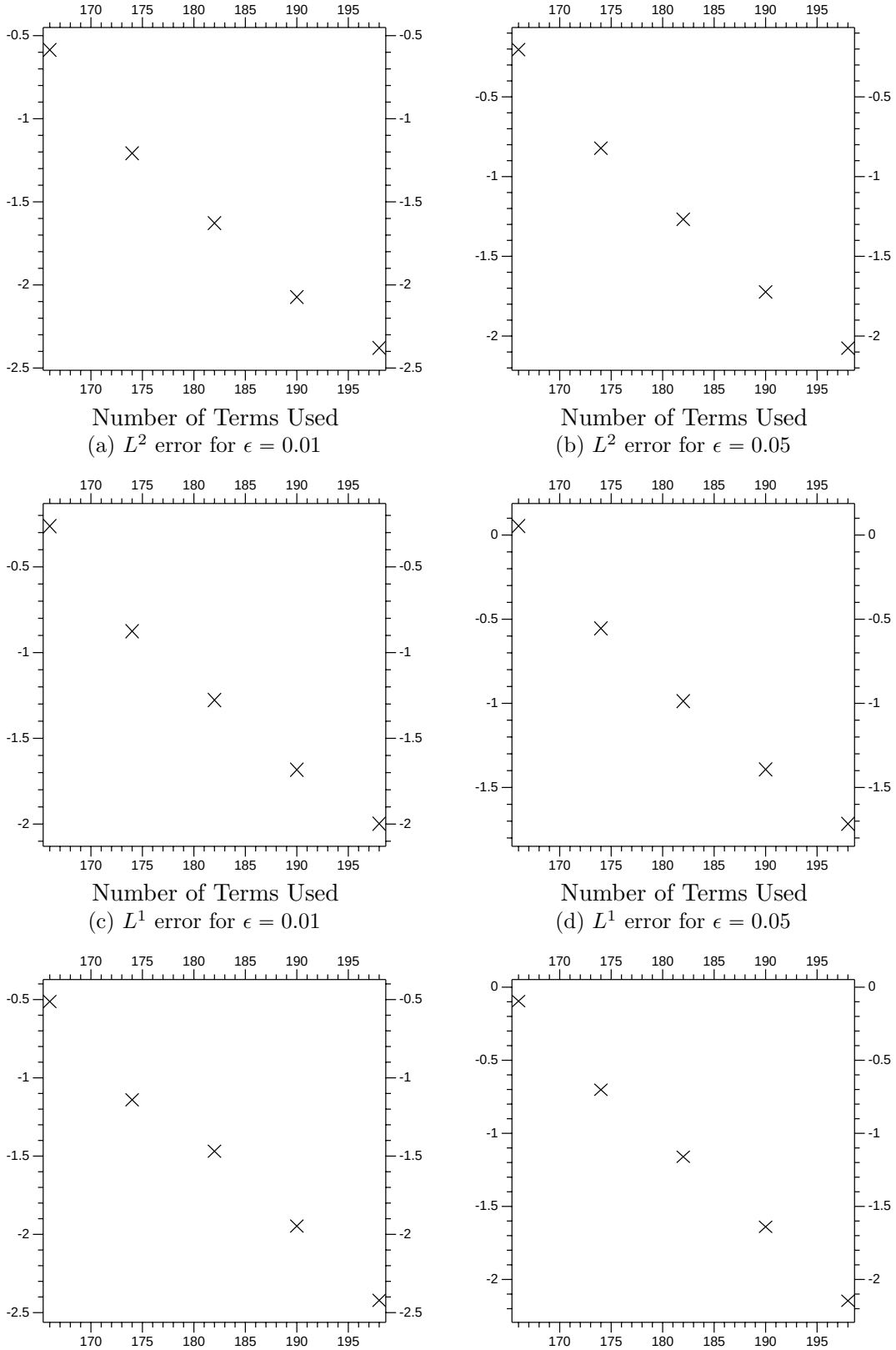
Table 10: Error from multi-scale linear Rossby wave results.

	Λ	'extra'	L^2 Error	L^1 Error	L^∞ Error
V_{-4}/V_{-6}	[3.75,6.25]	3	$4.66 \cdot 10^{-1}$	$5.82 \cdot 10^{-1}$	$6.61 \cdot 10^{-1}$
V_{-4}/V_{-6}	[3.50,6.50]	4	$2.97 \cdot 10^{-2}$	$2.76 \cdot 10^{-2}$	$4.42 \cdot 10^{-2}$
V_{-4}/V_{-6}	[3.25,7.00]	5	$6.95 \cdot 10^{-4}$	$7.99 \cdot 10^{-4}$	$7.45 \cdot 10^{-4}$
V_{-4}/V_{-6}	[3.00,7.00]	6	$3.09 \cdot 10^{-4}$	$1.81 \cdot 10^{-4}$	$5.98 \cdot 10^{-4}$
V_{-4}/V_{-6}	[2.50,7.50]	7	$3.65 \cdot 10^{-4}$	$1.50 \cdot 10^{-4}$	$5.98 \cdot 10^{-4}$
V_{-5}/V_{-6}	[3.75,6.25]	3	$4.05 \cdot 10^{-2}$	$2.78 \cdot 10^{-2}$	$7.19 \cdot 10^{-2}$
V_{-5}/V_{-6}	[3.50,6.50]	4	$4.68 \cdot 10^{-4}$	$3.14 \cdot 10^{-4}$	$9.46 \cdot 10^{-4}$
V_{-5}/V_{-6}	[3.25,7.00]	5	$5.64 \cdot 10^{-5}$	$2.13 \cdot 10^{-5}$	$1.65 \cdot 10^{-4}$
V_{-5}/V_{-6}	[3.00,7.00]	6	$5.62 \cdot 10^{-5}$	$1.81 \cdot 10^{-5}$	$1.65 \cdot 10^{-4}$
V_{-5}/V_{-6}	[2.50,7.50]	7	$5.62 \cdot 10^{-5}$	$1.78 \cdot 10^{-5}$	$1.65 \cdot 10^{-4}$

Table 11: Error from non-linear multi-scale Rossby wave results.

	ϵ	Λ	'extra'	L^2 Error	L^1 Error	L^∞ Error
V_{-3}/V_{-4}	0.01	[3.5,8.5]	4	$2.60 \cdot 10^{-1}$	$5.46 \cdot 10^{-1}$	$3.07 \cdot 10^{-1}$
V_{-3}/V_{-4}	0.01	[3.5,9.0]	4	$6.20 \cdot 10^{-2}$	$1.33 \cdot 10^{-1}$	$7.24 \cdot 10^{-2}$
V_{-3}/V_{-4}	0.01	[3.5,9.5]	4	$2.35 \cdot 10^{-2}$	$5.29 \cdot 10^{-2}$	$3.40 \cdot 10^{-2}$
V_{-3}/V_{-4}	0.01	[3.0,9.5]	4	$8.46 \cdot 10^{-3}$	$2.07 \cdot 10^{-2}$	$1.13 \cdot 10^{-2}$
V_{-3}/V_{-4}	0.01	[2.5,9.5]	4	$4.18 \cdot 10^{-3}$	$1.01 \cdot 10^{-2}$	$3.79 \cdot 10^{-3}$
V_{-3}/V_{-4}	0.05	[3.5,8.5]	4	$2.60 \cdot 10^{-1}$	$5.46 \cdot 10^{-1}$	$3.07 \cdot 10^{-1}$
V_{-3}/V_{-4}	0.05	[3.5,9.0]	4	$6.20 \cdot 10^{-2}$	$1.33 \cdot 10^{-1}$	$7.24 \cdot 10^{-2}$
V_{-3}/V_{-4}	0.05	[3.5,9.5]	4	$2.35 \cdot 10^{-2}$	$5.29 \cdot 10^{-2}$	$3.40 \cdot 10^{-2}$
V_{-3}/V_{-4}	0.05	[3.0,9.5]	4	$8.46 \cdot 10^{-3}$	$2.07 \cdot 10^{-2}$	$1.13 \cdot 10^{-2}$
V_{-3}/V_{-4}	0.05	[2.5,9.5]	4	$4.18 \cdot 10^{-3}$	$1.01 \cdot 10^{-2}$	$3.79 \cdot 10^{-3}$

very accurate results, especially for the vorticity function Z at a late time of $t = 50$. The usual plots of the error against the number of $a_{j,k}$ and $b_{j,k}$ coefficients used can be found in Figure 52. The summarized data can be found in Table 11.



(a) L^2 error for $\epsilon = 0.01$

(b) L^2 error for $\epsilon = 0.05$

(c) L^1 error for $\epsilon = 0.01$

(d) L^1 error for $\epsilon = 0.05$

(e) L^∞ error for $\epsilon = 0.01$

(f) L^∞ error for $\epsilon = 0.05$

Figure 52: The $\log_{10}$ of the error from $V_{-3} : V_{-4}$ non-linear multi-scale systems. Each is compared to a system with V_{-4} over the entire domain. The 'number of terms' includes all coefficients $a_{j,k}$ and $b_{j,k}$ included in the large and small scale systems, added together.

.

Chapter 9

Conclusion

Like many numerical methods, our implicit wavelet based multi-scale method has to be used on the correct type of problem. The localized small-scale interactions on our colliding wave Burgers' equation problem made it perfectly suited to our method. The two extra levels of resolution, extending a base of V_{-4} on $\Omega = (0, 10)$ to V_{-6} on $\Lambda = (3.75, 6.25)$, had the obvious advantage of preventing the approximation from becoming totally unstable. More importantly, the multi-scale system closely duplicated the full V_{-6} system's results, and using just over one thirtieth the computational effort (see Section 7.4). Next, a set of V_{-5} with localized V_{-6} multi-system calculations showed convergence towards the full domain V_{-6} system results as Λ increased in size (Section 7.5). So, the Burgers' equation problem was a complete success.

The next step was to apply the multi-scale method to a more substantial problem. Our Rossby wave problem, introduced in Section 8.1, contains many new elements and complications. These include non-zero boundary conditions, a second physical direction, and a much more complicated expression for the derivative, both for the linear and non-linear components (see Equation (8.1.6)). It is also reasonably well suited to the multi-scale method, with fine resolution required around the center of the domain. The linear, $\epsilon = 0$, version responded positively to a localized small-scale system in the center, extending the time frame where the problem performed as desired (Figures 37 and 42). The method is, fundamentally, for non-linear problems, though, and the non-linear Rossby wave problem needed to be tested. At the practical resolutions, V_{-3} to V_{-4}, the non-linear problem is not as conducive to the multi-scale

method as was Burgers' equation. While Burgers' equation was solved with one thirtieth the effort, the non-linear Rossby wave problem was solved satisfactorily with around half the effort (see Section 8.5). Overall the method worked, though not as well as it did with Burgers' equation. As desired, it provided accurate results with less effort (plots are in Figures 46 and 47 for $\epsilon = 0.01$, Figures 49 and 50 for $\epsilon = 0.05$).

To keep our examples as simple as possible, we have avoided a few options to improve the method. Most obviously, the essential elements of the method are useable in an adaptive scheme. The wavelet decomposition makes it easy to detect when the current resolution is insufficient. An adaptive scheme could be designed that would create a localized small-scale system when and where one is required. Our approach could also be helpful to general adaptive schemes. If an algorithm detects that a new W_j based element needs to be included in a system, the use of an implicit time step would make this difficult. Usually, it would need to be included only in the subsequent time step, since including it immediately would require recalculating the entire system. Our method can include any new W_j elements immediately.

Further improvements could involve using more than two systems. For instance, there is the possibility of nesting several systems. This would involve, for instance, having $\Gamma \subset \Lambda \subset \Omega$. The calculations would have Λ housing a small-scale system of Ω, and Γ housing a small-scale system of Λ. The three systems would be calculated sequentially, for significant savings. We could also have multiple small-scale systems for a domain. This would have $\Lambda \subset \Omega$, $\Gamma \subset \Omega$ with Λ and Γ a fair distance from each other. Small-scale values within Λ and Γ would be calculated sequentially, separately from each other. So the time step could be solved in three segments, each a third the total size of the full system. This would add up to greater savings than for a single small-scale system. Yet another advantage is that we need not solve the different systems the same way. Section B.5 has a discussion on the possibility of using a differently sized time step on the small-scale system. This probably understates our options. The systems may not even need to use the same time stepping scheme.

We can say with certainty that, with more substantial programming, sequential localized multi-scale methods could prove useful on stiff problems with localized small-scale interactions.

Appendix A

Useful Concepts

A.1 The Fourier Transform

We will be making heavy use of Fourier transforms, both continuous and discrete, in our proofs, examples, and numerical implementations. Therefore, we need clear definitions in place. The continuous transform and its inverse will be as follows:

$$\begin{aligned} \widehat{f}(\xi) &= \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} f(t)\, e^{-i\xi t}\, dt \\ \check{f}(\xi) &= \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} f(t)\, e^{i\xi t}\, dt. \end{aligned} \tag{A.1.1}$$

These are all defined primarily in the Schwartz space,

$$\mathcal{S} = \left\{ f \in C^{\infty}(\mathbb{R}^n) | \, \|f\|_{\alpha,\beta} < \infty \, \forall \, \alpha, \beta \in \mathbb{N} \right\} \tag{A.1.2}$$

using

$$\|f\|_{\alpha,\beta} = \left\| x^{\alpha} D^{\beta} f \right\|_{\infty}. \tag{A.1.3}$$

The Schwartz space is within the intersection of L^1, L^2 and C^{∞}, so all functions in $\mathcal{S}$ are in C^{∞} and are bounded using both the L^1 and L^2 norms. We will mostly use those properties of $\mathcal{S}$ functions.

There are several useful identities, easily proven using the definition of $\widehat{f}$. For $k \in \mathbb{R}$,

$$\widehat{f(t-k)}(\xi) = e^{-i\xi k} \widehat{f}(\xi), \quad \widehat{f(at)}(\xi) = \frac{1}{|a|} \widehat{f}(\xi/a), \quad \widehat{f}(\xi - k) = \widehat{e^{ikt} f}(\xi). \tag{A.1.4}$$

Also important is how derivatives work. Using integration by parts,

$$\begin{aligned}\widehat{f'(t)}(\xi) &= \int_{\mathbb{R}} f'(t)e^{-i\xi t}\,dt \\ &= \left(f(t)e^{-i\xi t}\right)\Big|_{-\infty}^{\infty} - \int_{\mathbb{R}}(-i\xi)f(t)e^{-i\xi t}\,dt \;=\; (i\xi)\;\widehat{f}(\xi)\end{aligned} \tag{A.1.5}$$

which gives $\widehat{f^{(r)}(t)}(\xi) = (i\xi)^r \widehat{f}(\xi)$ by induction. This means that if $f^{(r)}$ exists and is in L^1 then $|\xi|^r \hat{f}(\xi)$ goes to zero as $\xi \to \pm\infty$.

We also get the following property of $\widehat{f}$ if $f \in L^1$.

Property A.1.1 *If $f \in L^1$ then $\widehat{f}$ is continuous.*

Proof.

An elegant proof of this property can be found in [24, p. 300]. ■

One operation that we will use tied to the Fourier transform is usually referred to as convolution.

Definition A.1.2 (Convolution) *The convolution of two L^1 functions, f and g, is*

$$(f * g)(x) = \int_y f(y)g(x-y)\,dy. \tag{A.1.6}$$

There are many types of sufficient conditions for $f * g$ to be well defined, which we will not need. We will be using convolutions on functions with bounded support that have finite max and min values.

Here is the necessary property.

Property A.1.3 *Given two functions in L^2, f and g, we get*

$$\widehat{f * g}(\xi) = \sqrt{2\pi}\,\widehat{f}(\xi)\,\widehat{g}(\xi). \tag{A.1.7}$$

Proof.

This follows directly from the definitions and from integration rules. First,

$$\widehat{f * g}(\xi) = \frac{1}{\sqrt{2\pi}}\int e^{-ix\xi}\,(f * g)\;dx = \frac{1}{\sqrt{2\pi}}\int e^{-ix\xi}\int f(y)\,g(x-y)\,dy\,dx, \tag{A.1.8}$$

meaning that

$$\widehat{f * g}(\xi) = \frac{1}{\sqrt{2\pi}} \int e^{-ix\xi} f(y)\, g(x-y)\, dydx = \frac{1}{\sqrt{2\pi}} \int e^{-i(x+y)\xi} f(y)\, g(x)\, dydx$$
$$= \sqrt{2\pi} \int e^{-ix\xi} g(x)\, dx \int e^{-iy\xi} f(y)\, dy \tag{A.1.9}$$

which is equal to $\sqrt{2\pi}\, \widehat{f}(\xi)\, \widehat{g}(\xi)$. ■

A.2 Fourier Series

Fourier series will be composed of complex values, and on $[0, 2\pi)$ will use terms

$$e_k(x) = e^{ik\,x}, \quad k \in \mathbb{Z},\, x \in [0, 2\pi), \tag{A.2.1}$$

which makes it easy to see that

$$\langle e_k, e_l \rangle = \begin{cases} 2\pi & k = l, \\ 0 & k \neq l. \end{cases} \tag{A.2.2}$$

The discrete Fourier decomposition of a function $f(x)$ on $[0, 2\pi]$ (or a 2π periodic function on $\mathbb{R}$) will be

$$f(x) = \sum_{k\in\mathbb{Z}} f_k e_k(x) = \sum_{k\in\mathbb{Z}} \frac{\langle f, e_k \rangle}{\sqrt{2\pi}}\, e_k(x). \tag{A.2.3}$$

On $[0, 2\pi)$, the series will converge in L^2 and pointwise almost everywhere (see any text on Fourier analysis, like [24]). We also get the derivative

$$\frac{\partial}{\partial x} f(x) = \sum_{k\in\mathbb{Z}} (ik) f_k e_k(x) \qquad \Longrightarrow \qquad \frac{\partial^r}{\partial x^r} f(x) = \sum_{k\in\mathbb{Z}} (ik)^r f_k e_k(x), \tag{A.2.4}$$

which will be much easier to calculate than through finite difference approximation, for example. Equally importantly, if f is infinitely differentiable, then the coefficients a_k will have to converge to zero exponentially for (A.2.4) to have a finite norm for all r. This means that a finite approximation

$$f_N(x) = \sum_{k=0}^{N} f_k e_k(x) \tag{A.2.5}$$

will have an error $|f_N - f|$ that converges to zero exponentially as N increases. The error from the derivative approximation, $\|f'_N - f'\|$, will also have to converge exponentially to zero. So, $\|f'_N - f'\|$ has an error bound of the form Ce^{AN} ($C > 0$ and $A < 0$). In comparison, a finite difference scheme has an error bound of the form CN^{-2}.

Discrete Fourier series are particularly suited to solving partial differential equations with periodic boundary conditions. For example, the heat equation $u_t = u_{xx}$ on $[0, 2\pi)$ with periodic boundaries can be solved quite easily. If the initial conditions are $u(x, 0) = f(x)$ then

$$u(x, 0) = f(x) = \sum_k f_k e^{ik\,x}. \tag{A.2.6}$$

The solution is then going to be

$$v(x, t) = \sum_k f_k e^{ik\,x}\, e^{-k^2 t}, \tag{A.2.7}$$

notice that

$$v(x, 0) = \sum_k f_k e^{ik\,x}\, e^{-k^2 0} = \sum_k f_k e^{ik\,x} = f(x)$$

and that

$$v_t(x, t) = \sum_k -k^2 f_k e^{ik\,x}\, e^{-(k)^2 t} = \sum_k (ik)^2 f_k e^{ik\,x}\, e^{-(k)^2 t} = v_{xx}(x, t).$$

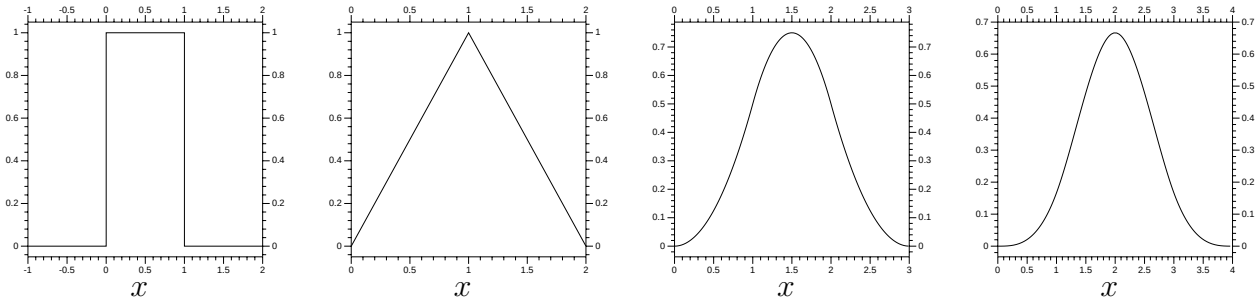
Notice also that the terms for $k \neq 0$ will go to zero, leaving a remainder of f_0, which is basically the mean value of $f(x)$.

A.3 Splines

The splines we will use are functions which are symmetric and have finite support. They are constructed recursively, starting with $B_0 = \phi_H = \mathbf{1}_{[0,1)}$. At each step,

$$B_{i+1}(x) = (B_i * B_0)(x) = (B_i * \phi_H)(x) = \int B_i(x - y)\phi_H(y)\, dy. \tag{A.3.1}$$

The integration results in a smoother function at each step, as shown in Figure 53. We have B_0, discontinuous (at zero and one); B_1, continuous, but only weakly



(a) $B_0(x)$ (b) $B_1(x)$ (c) $B_2(x)$ (d) $B_3(x)$

Figure 53: Some symmetric B splines, at 2^8 per unit resolution.

differentiable; and B_2, which has a continuous derivative. The last function in Figure 53 has a continuous second derivative, and can be written

$$B_3(x) = \begin{cases} 0 & x < 0 \\ \frac{1}{6}x^3 & 0 \leqslant x \leqslant 1 \\ \frac{2}{3} - 2x + 2x^2 - \frac{1}{2}x^3 & 1 \leqslant x \leqslant 2 \\ \frac{2}{3} - 2(4-x) + 2(x-4)^2 - \frac{1}{2}(4-x)^3 & 2 \leqslant x \leqslant 3 \\ \frac{1}{6}(4-x)^3 & 3 \leqslant x \leqslant 4 \\ 0 & x > 4, \end{cases} \tag{A.3.2}$$

So, B_r will be in C^{r-1}, symmetric around a point, constructed from polynomials of degree r, and have compact support. Furthermore, the derivatives are easy to calculate using the function data, since

$$\begin{aligned} \frac{d}{dx}B_{i+1}(x) = \frac{d}{dx}\left(\int_y B_i(x-y)B_0(y)\,dy\right)' &= \frac{d}{dx}\left(\int_y B_i(y)B_0(x-y)\,dy\right) \\ &= \frac{d}{dx}\left(\int_{x-1}^{x} B_i(y)\,dy\right) \\ &= B_i(x) - B_i(x-1), \end{aligned} \tag{A.3.3}$$

which can be iterated for any derivative. We will, however, be calculating these values via steady states of scaling steps, so we will need to calculate those next. This will require taking advantage of a useful property of convolutions.

Property A.3.1 *If ϕ_1 and ϕ_2 are functions with scaling steps associated with functions H_1 and H_2 (see Definition 2.4.1), then $\phi_3 = \phi_1 * \phi_2$ has the scaling step associated with $H_3 = H_1 H_2$.*

Proof.

This uses a basic property of the $*$ operator. As seen in Property A.1.3, $\widehat{\phi_1 * \phi_2} = \sqrt{2\pi}\,\widehat{\phi}_1\widehat{\phi}_2$, so

$$\widehat{\phi}_3(\xi) = \widehat{\phi_1 * \phi_2} = \sqrt{2\pi}\,\widehat{\phi}_1\widehat{\phi}_2 = \sqrt{2\pi}\,H_1\left(\frac{\xi}{2}\right)\widehat{\phi}_1\left(\frac{\xi}{2}\right)H_2\left(\frac{\xi}{2}\right)\widehat{\phi}_2\left(\frac{\xi}{2}\right) \tag{A.3.4}$$

using the Fourier equivalent of the scaling step. Next, we redistribute for

$$\begin{aligned}\widehat{\phi}_3(\xi) &= H_3\left(\frac{\xi}{2}\right)\sqrt{2\pi}\,\widehat{\phi}_1\left(\frac{\xi}{2}\right)\widehat{\phi}_2\left(\frac{\xi}{2}\right) \\ &= H_3\left(\frac{\xi}{2}\right)\widehat{\phi_1 * \phi_2}\left(\frac{\xi}{2}\right) = H_3\left(\frac{\xi}{2}\right)\widehat{\phi}_3\left(\frac{\xi}{2}\right),\end{aligned} \tag{A.3.5}$$

which is equivalent to ϕ_3 using the scaling step associated with H_3. ■

So, if the pairs $\left(\widehat{\phi}_1, H_1\right)$ and $\left(\widehat{\phi}_2, H_2\right)$ are both valid scaling systems then the product set, $\left(\widehat{\phi}_1\widehat{\phi}_2, H_1H_2\right)$, will also be a valid pair. The whole process starts with ϕ_H, which has $H(\xi) = \frac{1}{2}\left(1 + e^{-i\xi}\right)$. As a result,

$$H_n(\xi) = \left(\frac{1 + e^{-i\xi}}{2}\right)^{n+1}. \tag{A.3.6}$$

This makes it easy to calculate the h_k. They are simply

$$h_k = \begin{cases} \frac{\sqrt{2}}{2^{n+1}}\begin{pmatrix} n+1 \\ k \end{pmatrix} & 0 \leqslant k \leqslant n+1 \\ 0 & k \notin [0, n+1]. \end{cases} \tag{A.3.7}$$

As with scaling functions (Theorem 3.1.8, again), $B_n(x)$ can be calculated directly from the coefficients h_k.

Appendix B

Miscellaneous Wavelet Properties

B.1 The Order of ψ

Defining the order requires another definition first. The kth moment of ψ is

$$M_k(\psi) = \begin{cases} \int t^k \psi(t)\, dt & \text{if } t^k\psi \in L^1 \\ \infty & \text{if } t^k\psi \notin L^1. \end{cases} \tag{B.1.1}$$

Definition B.1.1 *The order of the wavelet ψ is N such that*

$$t^N\psi(t) \in L^1, \qquad M_k(\psi) = 0,\ 0 \leqslant k \leqslant N-1, \qquad M_N(\psi) \neq 0. \tag{B.1.2}$$

This implies that $\widehat{\psi} \in C^N$, and that $\widehat{\psi}^{(k)}(0) = 0$ for $k < N$ and $\widehat{\psi}^{(N)} \neq 0$. This, in turn, implies that we get a Taylor series of the form

$$\widehat{\psi}(\xi) = \widehat{\psi}^{(N)}(0)\, \xi^N + O\left(\xi^{N+1}\right). \tag{B.1.3}$$

Theorem B.1.2 *Take ψ of order N, with compact support. Assume that $f \in L^2$ is C^N in the set U around the point b. This implies*

$$Wf(a,b) = |a|^{N+1/2}\left(\gamma f^{(N)}(b) + o(1)\right) \tag{B.1.4}$$

as $a \to 0$, with $\gamma = \operatorname{sign}^N(a)\, \frac{\overline{M_N(\psi)}}{N!}$.

Proof.

This proof is fairly lengthy, and can be found in [5, p. 85]. ■

The implications of this result are hard to read from the expression, to say the least. It uses the continuous transform rather than the discrete version, which causes some issues. Primarily, the a term should be read as being, roughly, equivalent to 2^j, with j the order of the W space used (so, we are in W_j). Theorem B.1.2 implies that, as $j \to -\infty$, the $b_{j,k}$ terms corresponding to a particular location will converge to zero proportionally to $(2^j)^{N+\frac{1}{2}}$, where N is the order of the ψ used. Simply, the higher the order of ψ, the less important the small scale coefficients $b_{j,k}$ become.

The effect is variable, but for reasonably smooth functions it works out surprisingly consistently. Figures 54 and 55 contain a set of examples. First, a set of three Biorthogonal wavelets of orders 6, 4 and 2. Next, a function that will be decomposed into V_0, W_0, up to W_{-3} using each wavelet. Then, the coefficients used to decompose that function are compared, at the resolutions 0 to -3.

B.2 The Fast Wavelet Transform

So, let us assume that we have a function f in L^2, so the projection onto V_j, is

$$P_j f = \sum_k a_{j,k}\, \phi_{j,k} = \sum_k \langle f, \phi_{j,k} \rangle\, \phi_{j,k}. \tag{B.2.1}$$

We can calculate those values from the coefficients a_{j-1} as well:

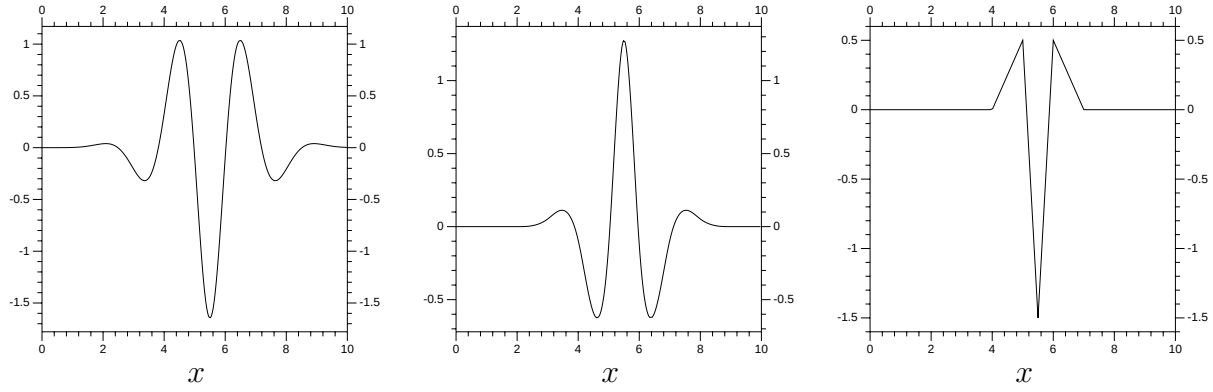
$$a_{j,k} = \langle f\ ,\ \phi_{j,k} \rangle = \left\langle f, \sum_l h_l\, \phi_{j-1,2k+l} \right\rangle = \sum_l \overline{h_l} \langle f, \phi_{j-1,2k+1} \rangle = \sum_l \overline{h_l}\, a_{j-1,2k+l}. \tag{B.2.2}$$

Now we look at the ψ related values. Here is the projection onto W_j:

$$Q_j f = \sum_k d_{j,k} \psi_{j,k} = \sum_k \langle f, \psi_{j,k} \rangle\, \psi_{j,k}. \tag{B.2.3}$$

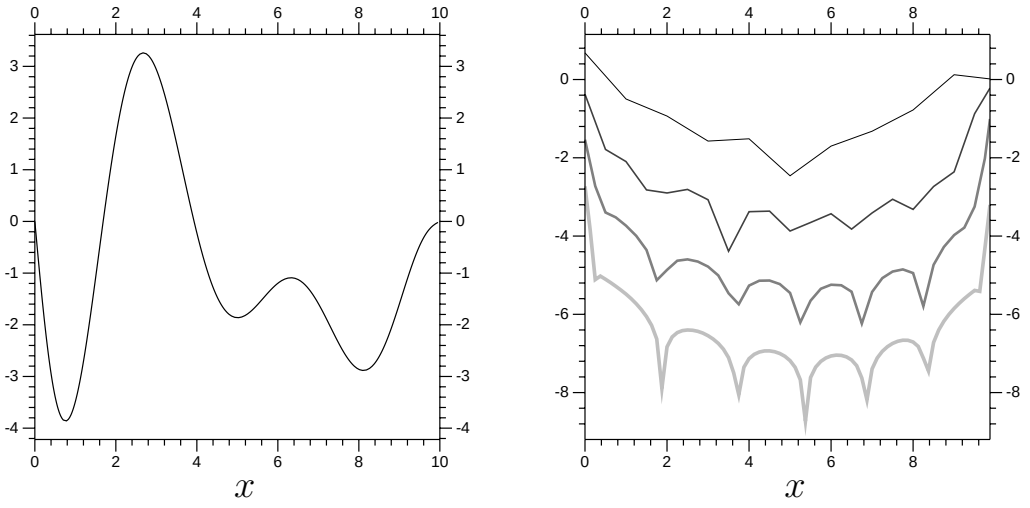
We have a similar relationship between ϕ_{j-1} and ψ_j, which works out similarly for

$$b_{j,k} = \langle f, \psi_{j,k} \rangle \quad = \quad \sum_l \overline{g_l} \langle f, \phi_{j-1,2k+l} \rangle \quad = \quad \sum_l \overline{g_l}\, a_{j-1,2k+l}. \tag{B.2.4}$$

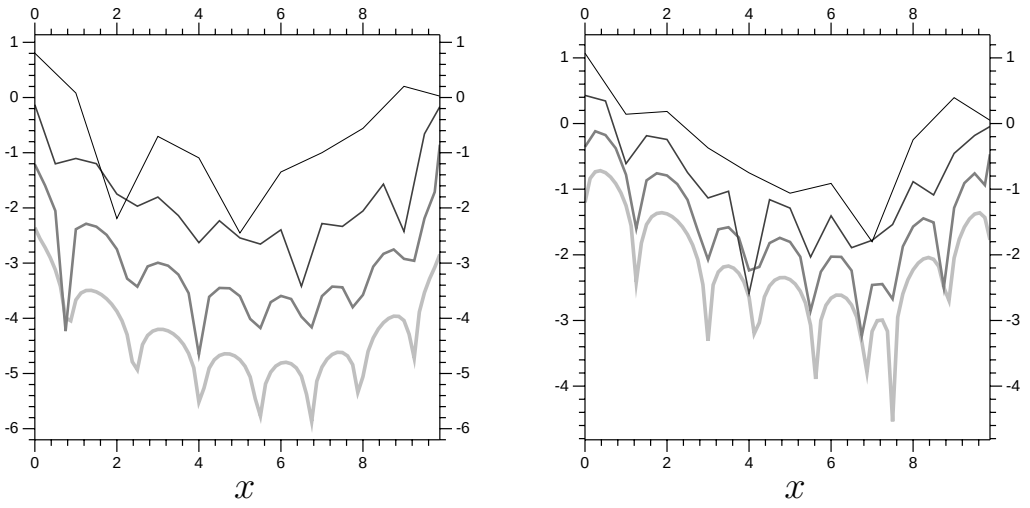


(a) Order 6 (b) Order 4 (c) Order 2

Figure 54: Three Biorthogonal Wavelets. Creating and using the wavelets is discussed in detail in Chapter 4.



(a) The function (b) 6



(c) 4 (d) 2

Figure 55: Logarithmic analysis showing the effect of order. In (a) is the function. It is smooth, but fairly steep, a good candidate to display the effect of wavelet order. The three wavelets used are from Figure 54, the coefficients from the respective decompositions of the function in the same figure. The thinner, darker, lines are for the lower resolutions.

These two equations mean that we can derive, for instance, both the values a_1 and b_1 from the values a_0 merely by using multiplication and addition. As a result, if we do a set of integrations, numerically or otherwise, of the values a_j then we can get a multi-resolution orthonormal decomposition of the function practically for free (at least in computational terms).

For numerical purposes we would have to be able to reverse the process. The forwards transform is from the highest resolution values a to the rest, so we are now going from the full set of values d to the highest resolution of values a. We take advantage of the definition of W_j here: $W_j = V_{j-1}/V_j$ so $P_{j-1}f = P_j f + Q_j f$ and so

$$\sum_k a_{j-1,k}\phi_{j-1,k} = \sum_k a_{j,k}\,\phi_{j,k} + \sum_k d_{j,k}\,\psi_{j,k}. \tag{B.2.5}$$

Taking the inner product of each side with $\phi_{j-1,k}$, we get

$$a_{j-1,k} = \sum_m a_{j,m}\,\langle \phi_{j,m}, \phi_{j-1,k}\rangle + \sum_n d_{j,n}\,\langle \psi_{j,n}, \phi_{j-1,k}\rangle\,. \tag{B.2.6}$$

Those inner products are easily found using the earlier expressions for $\phi_{j,m}$ and $\psi_{j,m}$, they are just g and h terms, so we get

$$a_{j-1,k} = \sum_m a_{j,m}\,h_{k-2m} + \sum_n d_{j,n}\,g_{k-2n}. \tag{B.2.7}$$

This operation is not quite like the forwards version, but it is yet again a fast transform as it gives us the highest resolution from the orthogonal decomposition using only addition and multiplication by known constants. Code for both the transform and its inverse can be found in Example C.1.3.

B.3 Periodic Boundary Conditions

Periodic boundary conditions are actually much easier with wavelets than other boundary conditions. Say that we are using a resolution of j, so we are in V_j. Our functions can be expressed in terms of $\phi_{j,k}$, and we have $2^{-j}10$ of them to cover a domain $[0, 10)$, centered at $k = 0$ to $10 - 2^j$ (the one at $x = 10$ needs to be equal to that at $x = 0$). The $\phi_{j,k}$ centered at $x = 0$ will have $k = 0$, the one at $10 - 2^j$ will

have $k = 10 \cdot 2^{-j} - 1 = K$. To satisfy periodic boundary conditions, what we need for derivative matrices (or any other location invariant operation) is for $\phi_{j,0}$ to affect $\phi_{j,2^{-j}10-1}$ the same way $\phi_{j,1}$ affects $\phi_{j,0}$. This results in the matrices used for such operations having the form

$$A = \begin{bmatrix} a_0 & a_1 & a_2 & \cdots & a_K \\ a_K & a_0 & a_1 & \cdots & a_{K-1} \\ a_{K-1} & a_K & a_0 & & a_{K-2} \\ \vdots & \vdots & & \ddots & \vdots \\ a_1 & a_2 & a_3 & \cdots & a_0 \end{bmatrix}. \tag{B.3.1}$$

The matrices themselves end up periodic in this respect. There is also a similar effect on the FWT and IFWT operations. This arrangement is typically easier than taking into account terms outside the boundary. Example code can be found in Section C.7.

B.4 Other Boundary Conditions

Our objective now is to figure out a general way to keep any number of derivatives to zero. This could also be changed to keep weighted sums of derivatives at zero, and so on. So, we have a scaling function $\phi(x)$, a boundary at $x = 0$, and a function

$$u(x) = \sum_k a_k \phi_{0,k}(x) = \sum_k a_k \, \phi(x - k). \tag{B.4.1}$$

For our purposes, u need only be a function of $x \in \mathbb{R}$. We need to make

$$u^{(n_1)}(0) = u^{(n_2)}(0) = \cdots = u^{(n_m)}(0) = 0, \quad n_1, \ldots, n_m \in \mathbb{Z} \text{ and } \geqslant 0. \tag{B.4.2}$$

What we want is a set of multipliers h_i for any $\phi_{0,k}(x)$ near the boundary such that

$$\frac{d^{(n)}}{dx^{(n)}} \left(\phi_{0,k}(x) + \sum_i h_i \phi_{0,i}(x) \right) = 0 \tag{B.4.3}$$

for all $n_1, \ldots, n_m$, with the i summing only over values where $\phi_{0,i}$ is supported at the boundary but is centered outside the boundary. Finding the h_i is done by setting up a linear system. We have the vector

$$b = \left[\phi^{(n_1)}(0 - k) \, , \, \phi^{(n_2)}(0 - k) \, , \, \ldots \, , \, \phi^{(n_m)}(0 - k) \right]^T , \tag{B.4.4}$$

and a matrix A with columns of the form

$$\left[\phi^{(n_1)}(0+i)\,,\,\ldots\,,\,\phi^{(n_m)}(0+i)\right]. \tag{B.4.5}$$

Solving the system $Ah + b = 0$ will give the values h to satisfy Equation (B.4.3).

A simple example of this system for the heat equation and the function ϕ_1 is

$$\begin{bmatrix} \phi_0(0) & \phi_{-1}(0) \\ (\phi_0)_{xx}(0) & (\phi_{-1})_{xx}(0) \end{bmatrix} \begin{bmatrix} h_0 \\ h_{-1} \end{bmatrix} + \begin{bmatrix} \phi_1(0) \\ (\phi_1)_{xx}(0) \end{bmatrix} = 0. \tag{B.4.6}$$

Of course, if the functions are symmetric then the easiest solution is obvious, so there is no problem. Also, while it may seem natural to use only the first and second functions outside the boundary to do this for, say, ϕ_2, creating an antisymmetry between u in Ω and u outside Ω makes calculating the wavelet transforms far simpler.

These values h can be set up to give us a simple linear relation between the functions ϕ inside and outside the boundary, which can be easily included into the matrices relating to the FWT and IFWT, as well as any used to calculate derivatives and the like.

A quick note about extending this to higher dimensions may be in order. Take an open set $\Omega \subset \mathbb{R}^n$ and a section of the boundary $\partial\Omega$ equal to $x_1 = 0$. We have, in this arrangement, functions like

$$\phi_{k_1,k_2,\ldots,k_n}(x_1, x_2, \ldots, x_n) = \phi_{k_1}(x_1)\phi_{k_2}(x_2)\phi_{k_3}(x_3)\cdots\phi_{k_n}(x_n) \tag{B.4.7}$$

which have most of their values within Ω, so in this case we want $k_1 > 0$ (if ϕ is symmetric around $x_1 = 0$). Next, let us consider the boundary for a particular $k_1, k_2, \ldots, k_n$, with k_1 relevantly close to the boundary. If we use the one dimensional arrangement, considering only the functions $\phi_{b,k_2,\ldots,k_n}$ for $b = -1, \ldots, -K$, then we get

$$\begin{aligned} &\phi_{k_1,k_2,\ldots,k_n}(x) + \sum_b a_b \phi_{b,k_2,\ldots,k_n}(x) \\ &= \phi_{k_1}(x_1)\left[\phi_{k_2}(x_2)\cdots\phi_{k_n}(x_n)\right] + \left(\sum_b a_b\phi_b(x_1)\right)\left[\phi_{k_2}(x_2)\cdots\phi_{k_n}(x_n)\right] = 0. \end{aligned} \tag{B.4.8}$$

So, all we have to do is keep the coefficients across the boundary related as above, then everything gets canceled out everywhere. The arrangement for a heat example with $\Omega = (-1,1) \times (-1,1)$ would be of the form

$$\begin{array}{c|c|c} & -u(x,2-y) & \\ \hline -u(-2-x,y) & u(x,y) & -u(2-x,y) \\ \hline & -u(x,-2-y) & \end{array} . \tag{B.4.9}$$

B.5 Multiple Time Steps

There is the possible need for differently sized time steps for the large and small-scale systems. A time step appropriate for the large-scale system may cause serious problems for the small-scale system, despite the use of an implicit method. The small-scale system will have 2^{-m} greater resolution, and is supposed to cover a region requiring that resolution, so greater instability and error are to be expected. We may need a second time step $h_2 = 2^{-m}h$ for the small-scale system. This issue can be solved surprisingly easily, however. What is needed is a set of vectors $\mathbf{a}_\Lambda$ at the times between our h sized time-steps. Any calculations needed can be done with a significantly reduced number of components, since we only need those in Λ, though it is probably best to use a greater number of $\phi_{j,k}$ related coefficients in the calculations than are included in the small-scale system. The approximations should be fairly accurate and stable. We could, for instance, calculate the midpoints by averaging

$$u_{n+\frac{1}{2}} \approx u_n + \frac{h}{4}\left(\frac{\partial}{\partial t}u_n + \frac{\partial}{\partial t}u_{n+\frac{1}{2}}\right), \quad u_{n+\frac{1}{2}} \approx u_{n+1} - \frac{h}{4}\left(\frac{\partial}{\partial t}u_{n+1} + \frac{\partial}{\partial t}u_{n+\frac{1}{2}}\right), \tag{B.5.1}$$

to get

$$u_{n+\frac{1}{2}} \approx \frac{1}{2}\left(u_n + u_{n+1}\right) + \frac{h}{4}\left(\frac{\partial}{\partial t}\,u_n - \frac{\partial}{\partial t}\,u_{n+1}\right). \tag{B.5.2}$$

This could be done for any $\frac{k}{2^m}$ step, since the number of terms used is bound to be marginal.

Appendix C

Example Code

Here are the MATLAB programs used to generate some of the graphs.

C.1 Basic Programs

Example C.1.1 *A simple program to approximate a scaling function using the h_k.*

```
function z=BetterPhi09(h,N)
M=size(h)(1,1)-1;   A=zeros(M+1,M+1);   s2=floor((M+1)/2);   s1=M+1-s2;
for i=1:s1   h1(i,1)=h(1+(i-1)*2,1);   endfor
for i=1:s2   h2(i,1)=h(2*i,1);   endfor
for i=1:(M+1)   a = i/2;   b = floor(a);
   if (b*2 >= i)  A(1:(M+1),i)=[zeros(b,1);h2;zeros(M+1-b-s2,1)];  endif
   if (b*2<i)  A(1:(M+1),i)=[zeros(b,1);h1;zeros(M+1-b-s1,1)];endif  endfor
A=sqrt(2)*A;   [V,L]=eig(A);   ERR=1;
for i=1:(M+1)   if(abs(L(i,i)-1)<ERR)   a(1:(M+1),1) = V(1:(M+1),i);
      ERR=abs(L(i,i)-1);   ERRE=L(i,i);   endif   endfor   h=h/ERRE;   b=0;
for i=1:(M+1)   b=b+a(i,1);   endfor   a=a/b;
for n=1:N   A = zeros(M*2^n+1,M*2^(n-1)+1);
   for i=1:(M*2^(n-1)+1)   A(1+2*(i-1),i) = 1;
   endfor;   for k=0:M   for i=1:(M*2^(n-1))   j=2*i-k*2^(n-1);
      if(j>0)  if(j<M*2^(n-1)+2)  A(2*i,j)=h(k+1,1)*sqrt(2);  endif  endif
   endfor   endfor   a=A*a;   endfor   z=[0];
```

```
for i=1:(M*2^N)   z(i+1,1)=i/(2^N);   endfor   z=[z a];
```

Example C.1.2 *This uses ϕ and the h_k to calculate ψ.*

```
function Z=BetterPSI09(h,phi)
s=size(h)(1,1);  if(s-2*floor(s/2))==0)  h=[h;0];  endif  M=size(h)(1,1)-1;
N=size(phi)(1,1)-1;   Res=(N-1)/(phi(N,1)-phi(1,1));   S=sqrt(2);
for i=1:(N/2+1)   A(i,1)=phi(2*i-1,2);   endfor   A(N/2+2:N,1)=0;
for i=1:(M+1)  g(i,1)=-(i-1)+M+1;  g(i,2)=S*(-1)^(g(i,1))*h(i,1);  endfor
N=N+1;   a=[phi(1:N,1) zeros(N,1)];
for i=1:(M+1)  j=g(i,1)*Res/2;
   a(1:N,2)=a(1:N,2)+g(i,2)*[zeros(j,1);A(1:N-j,1)];   endfor   Z=a;
```

Example C.1.3 *A simple Fast Wavelet Transform program, followed by its inverse.*

The 'Shift08' operator merely moves each column forward one space (removes the last column, puts it in front), the 'IShift08' operator does the opposite.

```
function Z=FWT08(h,F)
Z=F;   Fn=size(F)(1,1);   hn=size(h)(1,1);   J=log(Fn)/log(2);
for i=1:(hn)   g(i,1)=-(i-1)+hn-1;   g(i,2)=(-1)^(g(i,1))*h(i,1);   endfor
[s,i]=sort(g(:,1));   g=g(i,:);   g=g(1:hn,2);   ZZ=[];
for i=1:J   Fn=size(F)(1,1);   W=zeros(1,Fn);   W(1,1)=1;   Ah=W*h(1,1);
Ag=W*g(1,1);      for i=2:hn   W=Shift08(W);   Ah=Ah+W*h(i,1);
Ag=Ag+W*g(i,1);   endfor   Bh=Ah;   Bg=Ag;   for j=2:(Fn/2)
Bh=Shift08(Shift08(Bh));   Bg=Shift08(Shift08(Bg));   Ah=[Ah;Bh];
Ag=[Ag;Bg];   endfor   ZZ=[ZZ;Ag*F];   F=Ah*F;   Z=[[Z] [ZZ;F]];   endfor
```

The inverse is very similarly constructed, of course.

```
function F=IFWT08(h,F)
Fn=size(F)(1,1); hn=size(h)(1,1); J=log(Fn)/log(2);
for i=1:(hn)   g(i,1)=-(i-1)+hn-1;   g(i,2)=(-1)^(g(i,1))*h(i,1);   endfor
[s,i]=sort(g(:,1));   g=g(i,:);   g=g(1:hn,2);   h=[h;0;0];   g=[g;0;0];
for i=1:J      ii=2^(i-1);   jj=2^i;   W=zeros(1,ii);   W(1,1)=1;
```

```
AhE=W*h(1,1);   AgE=W*g(1,1);   AhO=W*h(2,1);   AgO=W*g(2,1);
for j=1:(hn/2)  W=ISHIFT08(W);  AhE=AhE+W*h(1+2*j,1);
AgE=AgE+W*g(1+2*j,1);  AhO=AhO+W*h(2+2*j,1);   AgO=AgO+W*g(2+2*j,1);
endfor   Ah=[AhE;AhO];   Ag=[AgE;AgO];   Bh=Ah;   Bg=Ag;
for j=2:(ii)  Bh=Shift08(Bh);  Bg=Shift08(Bg);  Ah=[Ah;Bh];  Ag=[Ag;Bg];
endfor  Z=F(1:Fn,1);
Z(Fn-jj+1:Fn,1)=Ag*F(Fn-jj+1:Fn-ii,1)+Ah*F(Fn-ii+1:Fn,1);  F=[Z F];  endfor
```

Example C.1.4 *A simple cascade method (Section 3.4) program.*

The vector Z is the starting values (just a single vector the size of h).

```
function Z=CascadePhi(h,n,Z)
hn=size(h)(1,1);   for i=1:n   ZZ=zeros(2^i*(hn-1)+1,1);   x=2^(i-1);
for k=1:hn   ZZ=ZZ+h(k,1)*[zeros((k-1)*x,1);Z;zeros((hn-k)*x,1)];
endfor   Z=ZZ;   endfor    Z=[zeros(size(Z)(1,1),1) Z];
x=(hn-1)/(size(Z)(1,1)-1);   xx=0;   for i=1:size(Z)(1,1)
Z(i,1)=xx;   xx=xx+x;   endfor
```

C.2 A Biorthogonal Setup

There is very little left to do for calculating biorthogonal wavelets and scaling functions. The previous section covers their calculations, one only needs the coefficients h_k and g_k. For the biorthogonal spline wavelets, it is easy to find the h_k and g_k, though we will have to take a look at the Fourier terms that will be used. Recall that the h_k are the coefficients from complex trigonometric polynomials (polynomials of e^{ikx}). We will be using the format

$$\begin{bmatrix} a_0 & a_1 & a_2 & \cdots \\ 0 & a_{-1} & a_{-2} & \cdots \end{bmatrix} \Longleftrightarrow \sum_k a_k e^{ik\xi} \tag{C.2.1}$$

in the programs, with associated multiplication and power functions. The first is `MultTwoFouriers`, which takes two matrices in the above format and outputs their properly formatted product. The second is `PowerFouriers` which takes the matrix form of a polynomial $p(\xi)$ and an integer n and outputs $p(\xi)^n$.

Now we calculate the coefficients h_k.

```
function Z=HFunction09(n)
Z1=[1/2 0;0 1/2];   Z1=PowerFouriers(Z1,n);
Z2=zeros(2,n);Z2(2,n)=1;   Z=MultTwoFouriers(Z1,Z2);
```

The coefficients $\widetilde{h}_k$ are marginally more complicated to work out.

```
function Z=H2Function09(n)
P=zeros(2,n);   Z=[1/2 0;0 0]+[0 -1/4;0 -1/4];
for i=0:n-1      Z1=factorial(n+i-1)/factorial(i)/factorial(n-1);
   Z1=Z1*PowerFouriers(Z,i);   sZ=size(Z1)(1,2);   sP=size(P)(1,2);
   if (sZ>sP)  P=[P zeros(2,sZ-sP)];  else  Z1=[Z1 zeros(2,sP-sZ)];  endif
P=P+Z1;   endfor   Z2=zeros(2,n);Z2(2,n)=1;   Z2=MultTwoFouriers(P,Z2);
Z3=PowerFouriers([1/2 0;0 1/2],n);   Z=MultTwoFouriers(Z3,Z2);
```

Thankfully, calculating the g_k and the resulting functions is no more difficult than with the orthogonal arrangement. All that is necessary is to remember that making ψ out of ϕ requires the values $\widetilde{h}_k$, and so forth. See Example C.1.1.

C.3 Heat Example

We will start with the biorthogonal wavelets and their boundary condition setup, since those elements will be retained for use with Burgers' equation and the Rossby wave problem.

First, all the programs will organize the coefficient data with vectors going from $x = 0$ down to $x = 10$, with ψ related coefficients put on top (if there are any). As a result, a V_{-2} equivalent vector could be arranged in three ways:

$$\begin{bmatrix} \mathbf{a}_{-2} \\ \mathbf{a}_{-2} \\ \mathbf{a}_{-2} \\ \mathbf{a}_{-2} \end{bmatrix}, \quad \begin{bmatrix} \mathbf{b}_{-1} \\ \mathbf{b}_{-1} \\ \mathbf{a}_{-1} \\ \mathbf{a}_{-1} \end{bmatrix}, \quad \begin{bmatrix} \mathbf{b}_{-1} \\ \mathbf{b}_{-1} \\ \mathbf{b}_{0} \\ \mathbf{a}_{0} \end{bmatrix}. \tag{C.3.1}$$

Note that the repeated terms are to show the size of the vectors: there should be twice as many $\mathbf{a}_{-2}$ terms as $\mathbf{a}_{-1}$, and so on.

Now to look to creating matrices used for the fast wavelet transforms. By arranging the h_k, g_k, and dual equivalents, appropriately into matrices `C` and `B`, the inverse wavelet transform $\begin{bmatrix} \mathbf{b}_0 \\ \mathbf{a}_0 \end{bmatrix} \to \begin{bmatrix} \mathbf{a}_{-1} \\ \mathbf{a}_{-1} \end{bmatrix}$ can be done with a multiplication by the matrix $[\mathtt{C}_0\ \mathtt{B}_0]$.

Example C.3.1 *The following program creates the matrix* `B` *that handles the scaling step on ϕ terms, so it converts the coefficients $a_{j,k}$ in $f = \sum_k a_k \phi_{j,k}$ into the $a_{j-1,k}$ in $f = \sum_k a_{j-1,k}\phi_{j-1,k}$. This is a component of the* `IFWT`*. The term* `Bound` *relates to the boundary conditions (*`Bound`$= 0$ *for even derivatives set to zero, anything else for odd). The* `res` *term is the highest resolution used, '*`pn`*' is the order of the ψ.*

```
function BB=ProperBiorthIFWTBMatrix(pn,domain,res,Bound)
h=HFunction09(pn);   h2=H2Function09(pn);   s=size(h2)(1,2);
h=[h(1,1) h(2,2:size(h)(1,2))]';   h=[h;zeros(s-size(h)(1,1),1)];
hn=size(h)(1,1);   hc=(hn+1)/2;   H=(h(1:hn,1))'*sqrt(2);   BB=[];
domain=ceil(domain);   if (Bound !=0)   Bound=1;   endif
S1=domain*(2^res)-1+2*Bound;  S2=domain*(2^(res-1))-1+2*Bound;
H=[H zeros(1,S1)];
for jj=1:(S2+1)/2     j=2*jj-1;      if (hc-j+Bound>1)
      H3=H(1,hc-j+Bound:S1+hc-j-1+Bound);
      H3=H3-H(hc+j+2-3*Bound:S1+hc+j+1-3*Bound)*(-1)^Bound;  BB=[BB;H3];
   else   BB=[BB;zeros(1,1+j-hc-Bound) H(1,1:S1+hc-j-1+Bound)];endif
endfor   if (Bound !=0)   BB(1,1:size(BB)(1,2))=BB(1,1:size(BB)(1,2))/2;
endif   BB=[BB;flipud(fliplr(BB))(2:(S2+1)/2,1:S1)]';
```

The `C` matrix, made of coefficients g_k, is arranged similarly, as are those used for the forwards transform.

Example C.3.2 *This program creates a matrix that handles the IFWT. For instance, if we enter* `ProperIFWTM(4,10,3,0,3,0)` *it will create a matrix that will convert a $\psi_{-2,k}$, $\psi_{-1,k}$, $\psi_{0,k}$ and $\phi_{0,k}$ based expression of a function and convert it into a purely $\phi_{-3,k}$ version of a function on a domain $\Omega = (0, 10)$ with zero boundary conditions, using $\phi = B_3(x)$.*

```
function Z=ProperIFWTM(pn,domain,m,m1,m2,Bound)
m1=abs(floor(m1));   m2=abs(floor(m2));   m=abs(floor(m));
mtop=max(m1,m2);mtop=min(mtop,m);   mlow=min(m1,m2);mlow=max(mlow,0);
if (Bound !=0)   Bound=1;   endif   s=domain*2^(m)-1+2*Bound;
s1=domain*2^(mlow)-1+2*Bound;   s2=domain*2^(mtop)-1+2*Bound;   Z=eye(s);
for i=mlow:mtop-1
   CCI=ProperBiorthIFWTCMatrix(pn,domain,i+1,Bound);
   BBI=ProperBiorthIFWTBMatrix(pn,domain,i+1,Bound);   ZZ=[CCI BBI];
   sZ=size(ZZ)(1,1);   Z(s-sZ+1:s,s-sZ+1:s)=ZZ*Z(s-sZ+1:s,s-sZ+1:s);  endfor
```

Example C.3.3 *This program will create a square matrix converting coefficients a_k from $\sum_k a_k \phi_{j,k}$ to the physical values at $k\,2^j$ for the function $\sum_k a_k \frac{\partial^d}{\partial x^d}\,\phi_{j,k}$.*

```
function Z=ExactBiorthDerMatrix(pn,der,res,domain,Bound)
phi2=VarBPhi109(pn,0);   phi1=VarBPhi109(pn-der,0);
phi=[phi2(1:size(phi2)(1,1),1) zeros(size(phi2)(1,1),1)];   c=der;
if (der>0)   phi(c+1,2)=phi1(1,2);
   for j=2:size(phi1)(1,1)   phi(c+j,2)=phi1(j,2)-phi1(j-1,2);   endfor
else   phi=phi2;   endif
for i=2:der   phi1=phi;   phi(1,2)=phi1(1,2);
      for j=2:size(phi1)(1,1)      phi(j,2)=phi1(j,2)-phi1(j-1,2);
   endfor   endfor   h=phi(:,2);   hc=(size(h)(1,1)+1)/2;   hs=size(h)(1,1);
s=domain*2^res+1;   h=[zeros((s-hs)/2,1);h;zeros((s-hs)/2,1)];
if (size(h)(1,1)>s)   h=h(hc-(s-1)/2:hc+(s-1)/2,1);   endif   Z=[h];
for i=1:(s-1)/2      ZZ=[h(i+1:s,1);zeros(i,1)];
   ZZ=ZZ-(-1)^(Bound+der)*[flipud(h(1:i+1,1));zeros(s-i-1,1)];
   Z=[ZZ Z (-1)^der*flipud(ZZ)];   endfor      Z=Z*2^(res/2)*2^(der*res);
if (Bound==0)   Z=Z(2:s-1,2:s-1);
else   Z(1:s,1)=Z(1:s,1)/2;   Z(1:s,s)=Z(1:s,s)/2;   endif
```

So, we can make an `IFWT` matrix to convert $\mathbf{b}_{j+1}, \mathbf{b}_{j+2}, \ldots, \mathbf{b}_0, \mathbf{a}_0$ to $\mathbf{a}_j$, and `FWT` for the inverse. Next, we get a $\texttt{Plot}_{j,d}$ matrix which will convert the $a_k\phi_j$ into physical values for $a_k\phi_j^{(d)}$. Then, we calculate $\texttt{Plot}_{j,0}$, which does the same thing but uses no

derivatives, so just $a_k\phi_{j,k}$. If we combine them all together properly we get the matrix $\texttt{FWT} \times (\texttt{Plot}_{j,0})^{-1} \times \texttt{Plot}_{j,d} \times \texttt{IFWT}$.

That matrix will, in order, convert from the usual wavelet decomposition to that based only on the values of $\phi_{j,k}$, plot the dth derivative of that function at appropriate points, calculate the $\phi_{j,k}$ based decomposition that will give you those values at those points, then convert back to the usual wavelet decomposition. What you get is a good approximation of the dth derivative.

Making a matrix to handle multiplication by a constant function is just as easy. Just line up the values of that function along a diagonal matrix `A` then use the matrix $\texttt{FWT} \times \texttt{Plot}^{-1} \times \texttt{A} \times \texttt{Plot} \times \texttt{IFWT}$.

C.4 Modified Advection Equation

There are several common input values for this, and subsequent, sections. First, we have `Z`, the initial conditions, with the time values on top then $\mathbf{b}_{J+1}$, $\mathbf{b}_{J+2}$ down to $\mathbf{b}_0$ then $\mathbf{a}_0$. Then there is $\texttt{Time} = [\texttt{h}, \texttt{n}, \texttt{m}]$, `h` is simply the length of the time step, `n` is the number of time steps between values put into the output and `m` is the number of output values. Next, we have $\texttt{Setup} = [\texttt{pn}, \texttt{res1}, \texttt{res2}, \texttt{resF}]$, with `pn` the order of the ϕ and ψ used (all biorthogonal, see Chapter 4), `res1` the resolution for the large-scale system (putting it to $V_{-\texttt{res1}}$), `res2` the additional levels of resolution for the small-scale system (only relevant for the multi-scale methods) and `resF` the number of Fourier functions beyond the first, $\kappa = 0$, function, in the modeling of the second spatial dimension (only relevant for the Rossby wave problem). For multi-scale methods there is also $\texttt{Domain} = [\texttt{loc1}, \texttt{loc2}]$, the physical boundaries of Λ, and `extra`.

Example C.4.1 *The program used to calculate the exact solution at a series of times, at a finite resolution.*

```
function Z=ModDifControl(Setup,Time)
h=Time(1,1)*Time(1,2);  n=Time(1,3);  res1=Setup(1,2);  s=10*2^res1-1;
ZZ=zeros(s,1);
```

```
for i=1:s   x=i/(s+1)*10; ZZ(i,1)=x*(x-2)*(x-8)*(x-10)/200;   endfor
t=0;Z=[t;ZZ];
for   i=1:n   t=t+h;   for i=1:s   x=i/(s+1)*10;
      if (x<5)   C=(25/(5-x)^2-1)/exp(5/4*t);   x=5-5/sqrt(1+C);
      elseif (x>5)   C=(25/(5-x)^2-1)/exp(5/4*t);   x=5+5/sqrt(1+C);
      else   x=5;   endif   y=x*(x-2)*(x-8)*(x-10)/200;   ZZ(i,1)=y;
   endfor   Z=[Z [t;ZZ]];      endfor
```

Example C.4.2 *The program used to calculate the numerical results.*

The input value `plotres` is the resolution of the output.

```
function Z=ModDif1(Z,Setup,Time,plotres)
pn=Setup(1,1);res1=Setup(1,2);h=Time(1,1);m=Time(1,2);n=Time(1,3);
PlotMatrix=ProperPlotMatrix(pn,res1,10,0);IPlotMatrix=inv(PlotMatrix);
IFWT=ProperIFWTM(pn,10,res1,res1,0,0);FWT=ProperFWTM(pn,10,res1,res1,0,0);
M1=FWT*IPlotMatrix*ExactBiorthDerMatrix(pn,1,res1,10,0)*IFWT;
s=size(M1)(1,1);   G=zeros(s);  for i=1:s  x=i/(s+1)*10;
G(i,i)=1/40*x*(x-5)*(x-10);  endfor
Tg=FWT*IPlotMatrix*G*PlotMatrix*IFWT;   EMat=eye(s);
Left1 =EMat+h/2*Tg*M1;Left1=inv(Left1);   Right1=EMat-h/2*Tg*M1;
Step=Left1*Right1;   plotter=ProperPlotMatrix(pn,res1+plotres,10,0);
plotter=plotter*ProperIFWTM(pn,10,res1+plotres,res1+plotres,0,0);
plotter=plotter(:,size(plotter)(1,1)-s+1:size(plotter)(1,1));
if size(Z)(1,1)<s+1
   for i=1:s    x=i/(s+1)*10;   F(i,1)=x*(x-2)*(x-8)*(x-10)/200;   endfor
   F=FWT*IPlotMatrix*F;   ZZ1=F;   Z=[0;plotter*ZZ1];   t=0;
else   t=Z(1,size(Z)(1,2));   ZZ1=Z(2:size(Z)(1,1),size(Z)(1,2));   endif
for i=1:n   for j=1:m   t=t+h;   ZZ1=Step*ZZ1;   endfor
   Z=[Z [t;plotter*ZZ1]];  endfor
```

C.5 Burgers' Equation

Apart from the input values from Section C.4, we have '`Problem`'= $[\beta, \nu]$, the constants used in the equation.

We start with the basic, single, system.

Example C.5.1 *A basic Burgers' equation code. Results can be found in Figures 21, 22, 23 and others.*

```
function Z=BasicSingleBurgers(Z,Setup,Problem,Time)
beta=Problem(1,1);vis=Problem(1,2);pn=Setup(1,1);res1=Setup(1,2);h=Time(1,1);
n=Time(1,2); m=Time(1,3); PlotMatrix=ProperPlotMatrix(pn,res1,10,0);
IPlotMatrix=inv(PlotMatrix);   IFWT=ProperIFWTM(pn,10,res1,res1,0,0);
FWT=ProperFWTM(pn,10,res1,res1,0,0);   s1=10*2^(res1)-1;   EMat=eye(s1);
M2=FWT*IPlotMatrix*ExactBiorthDerMatrix(pn,2,res1,10,0)*IFWT;
M1=FWT*IPlotMatrix*ExactBiorthDerMatrix(pn,1,res1,10,0)*IFWT;
Der1=vis*M2;   Left1=EMat-h/2*Der1;   Right1=EMat+h/2*Der1;
Decomp=PlotMatrix*IFWT;  IDecomp=FWT*IPlotMatrix;  NL10=h/2*beta*IDecomp;
NL11=Decomp*M1;NL12=Decomp;Z=[Z(1,1);zeros(s1,1)];t=Z(1,1);ZZ1=zeros(s1,1);
for i=1:m   for j=1:n
      t=t+h;   Temp=PutOnDiag09(NL11*ZZ1)*NL12+PutOnDiag09(NL12*ZZ1)*NL11;
      Temp=NL10*Temp;   ZZ1=(Left1-Temp)\(Right1*ZZ1);
      endfor   Z=[Z [t;ZZ1]];   endfor
```

Next we take a look at the double system Burgers' equation solver. There are several additional functions required, so we will get to those first. Next, we will make use of matrices related to wavelet transforms, plotting values, and so forth. Those can also be found in Section C.2. One new component is the `CropMatrix`.

```
function Z=CropMatrix(res1,res2,loc1,loc2,Bound,extra)
if (Bound != 0)   Bound=1;   endif   Z=[];
loc1=floor(2^res1*loc1);   loc2= ceil(2^res1*loc2);   width=loc2-loc1;
for i=1:res2
   res3=res2-i;   res3=2^res3;   s1=width*res3;   A=zeros(s1,extra*res3);
   A=[A eye(s1) A];  s1=size(A)(1,1);  s2=size(A)(1,2);  s3=size(Z)(1,1);
   s4=size(Z)(1,2);   Z=[[Z zeros(s3,s2)];[zeros(s1,s4) A]];      endfor
s1=width-1+2*Bound+2*extra;  A=eye(s1);  s1=size(A)(1,1);  s2=size(A)(1,2);
s3=size(Z)(1,1);  s4=size(Z)(1,2);  Z=[[Z zeros(s3,s2)];[zeros(s1,s4) A]];
```

`CropMatrix` is related to the 'extra' $\phi_{j,k}$ terms in the small-scale system, explained in Section 7.4. Assume you have a matrix, call it `M`, used to approximate an operator L in the 'extra' expanded domain Λ'. If you want a version of `M` that applies to Λ itself, you multiply `M` like so: `M2` = `CropMatrix` × `M` × `CropMatrix`'. This new matrix `M2` will be compatible with Λ.

Next, we have the `Take` and `Give` matrices, used to convert between the two systems. The `Take` matrix converts the values within Λ in the large-scale system into the lowest resolution of the small-scale system. The `Give` matrix does the opposite.

```
function Z=ProperTakeMatrix2(Setup,Domain,extra)
pn=Setup(1,1);   res1=Setup(1,2);   res2=Setup(1,3);   resF=Setup(1,4);
res3=res1+res2;   loc1=Domain(1,1);   loc2=Domain(1,2);   d2=loc2-loc1;
s1=10*2^res1-1;   s2=d2*2^res3-1+2*extra;   Z=zeros(s2,s1);
loc1=loc1*2^res1+1-extra;   loc2=loc2*2^res1-1+extra;
Z(s2-d2*2^res1-2*extra+2:s2,loc1:loc2)=eye(d2*2^res1-1+2*extra);
Z=Z*ProperIFWTM(pn,10,res1,res1,0,0);   Z=YOpWithFourier(Z,resF);

function Z=ProperGiveMatrix2(Setup,Domain,extra)
pn=Setup(1,1);   res1=Setup(1,2);   res2=Setup(1,3);   resF=Setup(1,4);
res3=res1+res2;   loc1=Domain(1,1);   loc2=Domain(1,2);
d2=loc2-loc1;s1=10*2^res1-1;   s2=d2*2^res3-1+2*extra;
Z=zeros(s1,s2);   loc1=loc1*2^res1+1-extra;   loc2=loc2*2^res1-1+extra;
Z(loc1:loc2,s2-d2*2^res1-2*extra+2:s2)=eye(d2*2^res1-1+2*extra);
Z=ProperFWTM(pn,10,res1,res1,0,0)*Z;   Z=YOpWithFourier(Z,resF);
```

Our final components are the programs that will be used to create the matrices M_Λ and D matrices out of the full matrix M (see Section 7.2). One thing to note: for ease of notation, Chapter 7 has the large-scale resolution V_j written in terms of functions $\phi_{j,k}$ (so the coefficients $a_{j,k}$). These programs write them in terms of functions $\phi_{0,k}$, then $\psi_{0,k}$ to $\psi_{j+1,k}$ (so a_0, b_0, and so on to $b_{j+1,k}$).

```
function Z=BasicCropHigherResDer(Der,Setup,Domain,extra)
Z=[];  loc1=Domain(1,1);loc2=Domain(1,2);  pn=Setup(1,1);  res1=Setup(1,2);
res2=Setup(1,3);resF=Setup(1,4);res3=res1+res2;sx=1+2*resF; s3=10*2^res3-1;
```

```
IFWT=ProperIFWTM(pn,10,res3,res1,0,0); FWT=ProperFWTM(pn,10,res3,res1,0,0);
for x1=1:sx  DerRow=[];  for x2=1:sx  DerTake=Der(1:s3,1+(x2-1)*s3:x2*s3);
DerTake=IFWT*DerTake*FWT;   DerGive=[];
for i=1:res2   l1=loc1*2^(res3-i)+1;   l2=loc2*2^(res3-i);
   DerGive=[DerGive;DerTake(l1:l2,1:s3)];
   DerTake=DerTake(1+10*2^(res3-i):size(DerTake)(1,1),1:s3);   endfor
l1=loc1*2^res1+1-extra;l2=loc2*2^res1-1+extra;
DerGive=[DerGive;DerTake(l1:l2,1:s3)];   DerTake=DerGive;DerGive=[];
s=size(DerTake)(1,1);
for i=1:res2   l1=loc1*2^(res3-i)+1;   l2=loc2*2^(res3-i);
   DerGive=[DerGive DerTake(1:s,l1:l2)];
   DerTake= DerTake(1:s,1+10*2^(res3-i):size(DerTake)(1,2));   endfor
l1=loc1*2^res1+1-extra;l2=loc2*2^res1-1+extra;
DerGive=[DerGive DerTake(1:s,l1:l2)];   DerRow=[DerRow DerGive];
endfor Der=Der(1+s3:size(Der)(1,1),1:size(Der)(1,2));Z=[Z;DerRow];endfor
```

```
function Der1=CollectLowerDer10(Der,Fm,m1)
sx=1+2*Fm;s=size(Der)(1,1);sy=s/sx;   m=log((sy+1)/10)/log(2);
Der1=[];   s1=10*2^m1-1;   loc=sy-s1+1;
for ix=1:sx      Der2=[];      for iy=1:sx
      A=Der(1+sy*(iy-1):sy*iy,1+sy*(ix-1):sy*ix);   A=A(loc:sy,loc:sy);
   Der2=[Der2;A];      endfor   Der1=[Der1 Der2];      endfor
```

Example C.5.2 *A program for Burgers' equation using a double system. It includes many of the improvements introduced in Section 7.4. Results are in Figures 26 and 27.*

Again, `extra` is the number of supplementary functions $\phi_{j,k}$ included in the small-scale system outside Λ, with V_j the highest resolution in the large-scale system (see Section 7.4). The `J` is applied only to the non-linear terms, and is explained in Section 7.4.

```
function Z=MayDoubleBurgers(Z,Setup,Problem,Time,Domain,extra,J)
h=Time(1,1);n=Time(1,2);m=Time(1,3);  beta=Problem(1,1); vis=Problem(1,2);
```

```
pn=Setup(1,1);res1=Setup(1,2);res2=Setup(1,3);   res3=res1+res2;
PlotMatrix=ProperPlotMatrix(pn,res3,10,0);IPlotMatrix=inv(PlotMatrix);
IFWT=ProperIFWTM(pn,10,res3,res3,0,0);FWT=ProperFWTM(pn,10,res3,res3,0,0);
M1=FWT*IPlotMatrix*ExactBiorthDerMatrix(pn,1,res3,10,0)*IFWT;
M2=FWT*IPlotMatrix*ExactBiorthDerMatrix(pn,2,res3,10,0)*IFWT;
loc1=Domain(1,1);loc2=Domain(1,2);d2=(loc2-loc1)*2^res1;
Domain2=[loc1-extra*2^(-res1) loc2+extra*2^(-res1)];
d22=(Domain2(1,2)-Domain2(1,1))*2^res1;
Der1=vis*M2;   Der2=BasicCropHigherResDer(Der1,Setup,Domain,extra);
Der1=CollectLowerDer10(Der1,0,res1);s1=10*2^res1-1;s2=d2*2^res2+2*extra-1;
Cropper=CropMatrix(res1,res2,loc1,loc2,0,extra);
PlotMatrix=ProperPlotMatrix(pn,res1,10,0);   IPlotMatrix=inv(PlotMatrix);
IFWT=ProperIFWTM(pn,10,res1,res1,0,0);FWT=ProperFWTM(pn,10,res1,res1,0,0);
Decomp=PlotMatrix*IFWT;   IDecomp=FWT*IPlotMatrix;
A0=beta*h/2*IDecomp;  A1=Decomp;  A2=Decomp*CollectLowerDer10(M1,0,res1);
PlotMatrix=2^(res1/2)*ProperPlotMatrix(pn,res2,d22,0);
IPlotMatrix=inv(PlotMatrix);   IFWT=ProperIFWTM(pn,d22,res2,res2,0,0);
FWT=ProperFWTM(pn,d22,res2,res2,0,0);
Decomp=PlotMatrix*IFWT;   IDecomp=FWT*IPlotMatrix;
B0=beta*h/2*Cropper*IDecomp;   B1=Decomp*Cropper';
B2=Decomp*BasicCropHigherResDer(M1,Setup,Domain2,0)*Cropper';JNL=eye(d22-1);
for i=1:size(J)(1,2)    JNL(i,i)=J(1,i);   JNL(d22-i,d22-i)=J(1,i);   endfor
Take=ProperTakeMatrix2(Setup,Domain,extra);
Give=ProperGiveMatrix2(Setup,Domain,extra);   EMat=eye(s1);
Left1=EMat-h/2*Der1;Right1=EMat+h/2*Der1;EMat=eye(s2);Left2=EMat-h/2*Der2;
Right2C=Der2;  Right2C(s2-d2-2*extra+2:s2,s2-d2-2*extra+2:s2)=zeros(d22-1);
Right2C=h/2*Right2C;Right2=EMat;Right2(s2-d22+2:s2,s2-d22+2:s2)=zeros(d22-1);
Right2=Right2+Right2C;   loc1=loc1*2^res1-extra;   loc2=loc2*2^res1+extra;
Take2=zeros(s2,s1);   Take2(s2-d22+2:s2,loc1+1:loc2-1)=eye(d22-1);
Decomp=ProperPlotMatrix(pn,res1,10,0);   Take2=Take2*inv(Decomp);
Take2(s2-d22+2:s2,loc1+1:loc2-1)=Take2(s2-d22+2:s2,loc1+1:loc2-1);
Take2=h/2*beta*Take2;
if (size(Z)(1,1)<s1+s2+1)
```

```
   Z=[0;zeros(s1-9,1);0;1;2;1;0;-1;-2;-1;0;zeros(s2,1)];
   Z(2+s1:1+s1+s2,1)=Take*Z(2:1+s1,1);   endif      t=Z(1,size(Z)(1,2));
ZZ1=Z(2:s1+1,size(Z)(1,2));   ZZ2=Z(2+s1:1+s1+s2,size(Z)(1,2));
for i=1:m   for j=1:n
      t=t+h;   Temp =(PutOnDiag09(A1*ZZ1)*A2+PutOnDiag09(A2*ZZ1)*A1);
      TempA=A0*Temp;   ZZ1=(Left1-TempA)\(Right1*ZZ1);   Taken=Take*ZZ1;
      TempB=B0*(PutOnDiag09(B1*ZZ2)*B2+PutOnDiag09(B2*ZZ2)*B1);
      Taken2=TempB*Taken;   Taken2=Taken2-Take2*Temp*ZZ1;
      Taken2(s2-d22+2:s2,1)=JNL*Taken2(s2-d22+2:s2,1);
      ZZ2=(Left2-TempB)\(Right2*ZZ2+Right2C*Taken+Taken2);
      ZZ1=ZZ1+Give*ZZ2;   ZZ2=ZZ2+Taken;
   endfor   Z=[Z [t;ZZ1;ZZ2(1:s2-d22+1,1);zeros(d22-1,1)]];   endfor
```

C.6 The Rossby Wave Problem

The data will be decomposed via scaling functions and wavelets in the y direction and Fourier series in the x direction. If there are s elements in the V_j space used then the first s elements will be the coefficients of $\phi_{j,k}(y)$ (or its equivalents), corresponding with the zero Fourier term for x. The next s sized bunches will be the a then the b terms from $\phi_{j,k}(y)$ $((a-bi)e^{ix}+(a+bi)e^{-ix})$. Since

$$(a-bi)e^{ix}+(a+bi)e^{-ix}=2a\cos(x)+2b\sin(x) \tag{C.6.1}$$

this gives all the values simple physical equivalents. We will use the expression `ProperXDerMatrix(s1,resF,d)` to create derivative matrices. The first input is the y direction size, s_1, `resF` is the highest Fourier term included and `d` is the derivative used. It creates matrices of the form

$$\frac{\partial}{\partial x}\begin{bmatrix} O & O & O & O & O \\ O & O & -I & O & O \\ O & I & O & O & O \\ O & O & O & O & -2I \\ O & O & O & 2I & O \end{bmatrix}, \qquad \frac{\partial^2}{\partial x^2}\begin{bmatrix} O & O & O & O & O \\ O & -I & O & O & O \\ O & O & -I & O & O \\ O & O & O & -4I & O \\ O & O & O & O & -4I \end{bmatrix}, \tag{C.6.2}$$

with each O and I an `s1` sized zero or identity matrix.

Example C.6.1 *A program used to calculate the boundary condition function.*

It creates the function, but puts it at y=5 not y=10. The derivative input `Der` has to be setup so as to give the proper derivative at that point. The next two are the `M2` matrix for approximating Δ, then the total resolution used. The next is the resolution of the resulting boundary function (we want it to be much smaller than the total resolution, it's usually zero). Finally, we get the order of the ϕ used and the Fourier term used for the boundary function and the value of $Z = \Delta P$ desired at the boundary.

The vectors `A1` and `A2` calculate the physical value of the given derivative at that point, for cos and sin respectively. The eventual result is going to be solved so as to get those values to zero. Next, `A3` and `A4` are used to control the $Z = \Delta P$ of the function at the boundary. Finally, the physical value of the boundary function is controlled via `A5` and `A6`.

```
function Z=ProperCalcBC3(Der,M2,res1,res2,resF,pn,f,ZAtBound)
FWT1=ProperFWTM(pn,10,res1,res1,0,0);IFWT1=ProperIFWTM(pn,10,res1,res1,0,0);
FWT2=ProperFWTM(pn,10,res1,res2,0,0);IFWT2=ProperIFWTM(pn,10,res1,res2,0,0);
s1=10*2^(res1)-1;s2=10*2^(res2)-1;   c1=(s1+1)/2;c2=(s2+1)/2;
PlotMatrix=ProperPlotMatrix(pn,res1,10,0);
Der=YOpWithFourier(PlotMatrix*IFWT1,resF)*Der*YOpWithFourier(FWT2,resF);
M2=YOpWithFourier(PlotMatrix*IFWT1,resF)*M2*YOpWithFourier(FWT2,resF);
Plt=PlotMatrix*IFWT1*FWT2;Out1=s1*(2*f+0)-c1+1;Out2=s1*(2*f+1)-c1+1;
Inp1=s1*(2*f+0)-c2;Inp2=s1*(2*f+1)-c2;
A1=[Der(Out1,Inp1-1:Inp1+1) Der(Out1,Inp2-1:Inp2+1)];
A2=[Der(Out2,Inp1-1:Inp1+1) Der(Out2,Inp2-1:Inp2+1)];
A3=[ M2(Out1,Inp1-1:Inp1+1)  M2(Out1,Inp2-1:Inp2+1)];
A4=[ M2(Out2,Inp1-1:Inp1+1)  M2(Out2,Inp2-1:Inp2+1)];
A5=[Plt(c1,s1-c2-1:s1-c2+1) [0 0 0]];A6=[[0 0 0] Plt(c1,s1-c2-1:s1-c2+1)];
M=[A1;A2;A3;A4;A5;A6];   ZZ=M\[0;0;ZAtBound;0;1/2;0];
Z=zeros(s1,2*resF+1);  Z(s1-c2-1:s1-c2+1,2*f:2*f+1)=[ZZ(1:3,1) ZZ(4:6,1)];
Z=UnRect(Z);   Z=YOpWithFourier(FWT2,resF)*Z;
```

That program does not work when $\nu = 0$, since the resulting system becomes singular. For that case we need to use a different program, one that uses fewer coefficients.

```
function Z=ProperCalcBC(Der,res1,res2,resF,pn,f)
FWT1=ProperFWTM(pn,10,res1,res1,0,0);IFWT1=ProperIFWTM(pn,10,res1,res1,0,0);
FWT2=ProperFWTM(pn,10,res1,res2,0,0);IFWT2=ProperIFWTM(pn,10,res1,res2,0,0);
s1=10*2^(res1)-1;   s2=10*2^(res2)-1;   c1=(s1+1)/2;   c2=(s2+1)/2;
PlotMatrix=ProperPlotMatrix(pn,res1,10,0);
M1=YOpWithFourier(PlotMatrix*IFWT1,resF)*Der*YOpWithFourier(FWT2,resF);
spot1=c1+s1*(f*2);  spot2=s1-c2+s1*((f-1)*2+1);  M1=M1(spot1,spot2:spot2+1);
M2=PlotMatrix*IFWT1*FWT2;M2=M2(c1,s1-c2:s1-c2+1);M=[M1;M2];   ZZ=M\[0;1/2];
Z=zeros(s1*(2*resF+1),1);Z(spot2:spot2+1,1)=ZZ;Z=YOpWithFourier(FWT2,resF)*Z;
```

Example C.6.2 *A basic solver for $\epsilon = 0$, the linear case. All linear results using a single system in Chapter 8 used this program.*

The inputs are much the same as for the examples in Section C.5. One difference is a fourth term in `Setup`, `resF`, the order of the highest Fourier term used. `Problem`= [δ, β, ν, ϵ], though the last one should be zero in this case. `T1` is the t_1 from the 'switch-on' function, `BCres` is the resolution of the boundary function calculated, so it will be in V_{-BCres}. Finally we have `FMult`, which is a multiplier applied to all Fourier terms, so if `FMult`=2, we only work with the even ones. Finally, 'u' is the value of $\overline{u}(y)$ at the boundary, usually tanh(5).

There are a few very simple programs used here that need explaining. `YOpWithFourier(M,resF)` outputs a matrix with M repeated along its diagonal, one for every Fourier coefficient. `ProperU1` makes vectors out of values of $\overline{u}(y)$ (`ProperU2` gives $\overline{u}''(y)$). The first value is the resolution needed, the next two gives the domain. The matrix `Locale` is applied to the boundary function, and what it does is move a function in the y direction by five points. The boundary function is created centered at $y = 5$, so this moves it up to where it needs to be.

```
function Z=BasicRosF2(Z,Setup,Problem,Time,BCres,u,FMult,T1)
```

```
del=Problem(1,1);beta=Problem(1,2);vis=Problem(1,3);
h=Time(1,1); n=Time(1,2); m=Time(1,3);
pn=Setup(1,1);res1=Setup(1,2);resF=Setup(1,4);
sx=1+2*resF;   s1=10*2^(res1)-1;   EMat=eye(s1*sx);
PlotMatrix=ProperPlotMatrix(pn,res1,10,0);  IPlotMatrix=inv(PlotMatrix);
IFWT=ProperIFWTM(pn,10,res1,res1,0,0);FWT=ProperFWTM(pn,10,res1,res1,0,0);
M1x=FMult*ProperXDerMatrix(s1,resF,1);M2x=FMult^2*ProperXDerMatrix(s1,resF,2);
M2y=FWT*IPlotMatrix*ExactBiorthDerMatrix(pn,2,res1,10,0)*IFWT;
M2y=YOpWithFourier(M2y,resF);   M2=M2y+del*M2x;   IM2=inv(M2);
TU1=ProperU1(res1,0,10);   TU2=ProperU2(res1,0,10);
TU1=YOpWithFourier(FWT*IPlotMatrix*PutOnDiag09(TU1)*PlotMatrix*IFWT,resF);
TU2=YOpWithFourier(FWT*IPlotMatrix*PutOnDiag09(TU2)*PlotMatrix*IFWT,resF);
Der1=-(TU1*M1x+(beta*EMat-TU2)*M1x*IM2-vis*M2);
Left1=EMat-h/2*Der1;   Left1=inv(Left1);   Right1=EMat+h/2*Der1;
Locale=ProperLocale(FWT,IFWT,PlotMatrix,IPlotMatrix,res1,resF);
BC=-(u*M1x*M2+(beta-(-2*u)*(1-u^2))*M1x-vis*M2*M2);
u=(-1/2)*(beta-(-2*u)*(1-u^2))/u;
if (vis==0)   BC=ProperCalcBC(BC,res1,BCres,resF,pn,2/FMult);   else
   BC=ProperCalcBC3(BC,M2,res1,BCres,resF,pn,2/FMult,u);   endif
BC=h*(-TU1*M1x*Locale*M2*BC-(beta*EMat-TU2)*Locale*M1x*BC+vis*Locale*M2*M2*BC);
Zn=size(Z)(1,2);   t=Z(1,Zn);   ZZ=Z(2:size(Z)(1,1),Zn);
for i=1:m   for j=1:n
      FSwitch=t/T1;  t=t+h;  FSwitch=FSwitch+t/T1;  FSwitch=min(FSwitch/2,1);
      ZZ=Left1*(Right1*ZZ+BC*FSwitch);   endfor   Z=[Z [t;ZZ]];   endfor
```

Example C.6.3 *A workable multi-system solver for the linear Rossby wave problem. Figure 42 comes from this program.*

```
function Z=MarchRosMultF2(Z,Setup,Problem,Time,Domain,extra,BCres,u,BCF,T1)
del=Problem(1,1);beta=Problem(1,2);vis=Problem(1,3);nonlin=Problem(1,4);
h=Time(1,1);n=Time(1,2);m=Time(1,3);pn=Setup(1,1);res1=Setup(1,2);
res2=Setup(1,3);  resF=1;   sx=1+2*resF;   res1=res1+res2;
PlotMatrix=ProperPlotMatrix(pn,res1,10,0);IPlotMatrix=inv(PlotMatrix);
IFWT=ProperIFWTM(pn,10,res1,res1,0,0);FWT=ProperFWTM(pn,10,res1,res1,0,0);
```

```
s1=10*2^(res1)-1;EMat=eye(s1*sx);
M1x=BCF*ProperXDerMatrix(s1,resF,1);M2x=BCF^2*ProperXDerMatrix(s1,resF,2);
M2y=FWT*IPlotMatrix*ExactBiorthDerMatrix(pn,2,res1,10,0)*IFWT;
M2y=YOpWithFourier(M2y,resF);M2=M2y+del*M2x;IM2=inv(M2);
TU1=ProperU1(0,res1,0,10,0);   TU2=ProperU2(0,res1,0,10,0);
TU1=YOpWithFourier(FWT*IPlotMatrix*PutOnDiag09(TU1)*PlotMatrix*IFWT,resF);
TU2=YOpWithFourier(FWT*IPlotMatrix*PutOnDiag09(TU2)*PlotMatrix*IFWT,resF);
Locale=ProperLocale(FWT,IFWT,PlotMatrix,IPlotMatrix,res1,resF);
BC=-(u*M1x*M2+(beta-(-2*u)*(1-u^2))*M1x-vis*M2*M2;
u=(-1/2)*(beta-(-2*u)*(1-u^2))/u;
if (vis==0)   BC=ProperCalcBC(BC),res1,BCres,resF,pn,1);   else
   BC=ProperCalcBC3(BC,M2,res1,BCres,resF,pn,1,u);   endif
BC=-TU1*M1x*Locale*M2*BC-(beta*EMat-TU2)*Locale*M1x*BC+vis*Locale*M2*M2*BC;
BC=h*BC;BC=ReRect(BC,s1,sx);BC=BC(s1-10*2^(res1-res2)+2:s1,:);BC=UnRect(BC);
Der=-(TU1*M1x+(beta*EMat-TU2)*M1x*IM2-vis*M2);   res1=res1-res2;
loc1=Domain(1,1);loc2=Domain(1,2);   Der1=CollectLowerDer10(Der,resF,res1);
Der2=BasicCropHigherResDer(Der,[pn res1 res2 1],Domain,extra);
EMat=eye(size(Der1)(1,1));   Left1=EMat-h/2*Der1;Left1=inv(Left1);
Right1=EMat+h/2*Der1;   EMat=eye(size(Der2)(1,1));   Left2=EMat-h/2*Der2;
Left2=inv(Left2);  Right2C=Der2;  s=size(Der2)(1,1)/sx;
d2=(loc2-loc1)*2^res1+2*extra;
for i=1:sx   for j=1:sx    Right2C(s*i-d2+2:s*i,s*j-d2+2:s*j)=zeros(d2-1);
endfor   endfor   Take=ProperTakeMatrix(pn,res1,res2,resF,loc1,loc2,extra);
Give=ProperGiveMatrix(pn,res1,res2,resF,loc1,loc2,extra);
Right2=EMat-Take*Give+h/2*Right2C;   Right2C=h/2*Right2C;
s1=10*2^res1-1;s2=(d2-2*extra)*2^res2-1+2*extra;
if size(Z)(1,1)<((s1+s2)*sx)  ZZ1=zeros(s1*sx,1);ZZ2=zeros(s2*sx,1);  t=0;
else   ZZ1=Z(2:1+s1*sx,Zx);   ZZ2=Z(2+s1*sx:1+s1*sx+s2*sx,Zx);
   t=Z(1,size(Z)(1,2));   endif
for i=1:m  for j=1:n  FShift=t/T1;  t=t+h;  FShift=min((FShift+t/T1)/2,1);
      ZZ1=Left1*(Right1*ZZ1+BC*FShift);   Taken=Take*ZZ1;
      ZZ2=Left2*(Right2*ZZ2+Right2C*Taken);ZZ1=ZZ1+Give*ZZ2;
      ZZ2=ZZ2+Taken;   endfor   Z=[Z [t;ZZ1;ZZ2]];   endfor
```

C.7 Periodic Boundary Conditions

First, an example B matrix for the FWT (with coefficients h_k as its elements). This is for Biorthogonal versions, but it could easily be re-arranged for others (just change the coefficients h_k).

```
function B=VarBiorthFWT09BCMatrix(i,order)
   h=H2Function09(order);   h=[h(1,1) h(2,2:size(h)(1,2))]';
   hn=size(h)(1,1);   hc=(hn+1)/2;   H=(h(1:hn,1))'*sqrt(2);   B=[];
   S1=10*(2^i); S2=10*(2^(i-1)); H=[H(1,hc:hn) zeros(1,S1-hn) H(1,1:hc-1)];
   B=H;   for jj=2:S2   H=Shift09(H);   B=[B;H];   endfor
```

The `Shift08` operator merely moves each column forward one space (removes the last column, puts it in front)

Next, a matrix for making derivative matrices. The `Derivative1` operator creates an identically supported approximation (via finite difference) of $\frac{d}{dx}$ out of whatever functional data is given.

```
function Z=CreatePeriodicDerMatrix(phi,n,d)
   for i=1:d   phi=Derivative1(phi);   endfor
   s=size(phi)(1,1);   t=phi(s,1);   r=(s-1)/t;   phiP=phi(1,2);
   for i=1:t   phiP=[phiP;phi(1+i*r,2)];   endfor   r=size(phiP)(1,1);
   t=(r+1)/2;   s=10*2^n;   phiP=[phiP(t:r,1);   zeros(s-r,1);
   phiP(1:t-1,1)];   Z=[];   for i=1:s   Z=[Z phiP];   phiP=[phiP(s,1);
   phiP(1:s-1,1)];   endfor   Z=2^(n/2)*2^(d*n)*Z;
```

The key part is the second last line, where the function data `phiP` is merely rotated at every step. This is much easier than symmetric or anti-symmetric boundary conditions.

Bibliography

[1] J.M. Alam, N.K.R. Kevlahan and O.V. Vasilyev, *Simultaneous Space-Time Adaptive Wavelet Solution of Nonlinear Parabolic Differential Equations*, Journal of Computational Physics **214** (2006), 829–857.

[2] M. Béland, *Numerical Study of the Nonlinear Rossby Wave Critical Level Development in a Barotropic Zonal Flow*, Journal of the Atmospheric Sciences **33** (1976), 2066–2078.

[3] G. Beylkin and J.M. Keiser, *On the Adaptive Numerical Solution of Nonlinear Partial Differential Equations in Wavelet Bases*, Journal of Computational Physics **132** (1997), Issue 2, 233–259.

[4] G. Beylkin, J.M. Keiser and L. Vozovoi, *A New Class of Time Discretization Schemes for the Solution of Nonlinear PDEs*, Journal of Computational Physics **147** (1998), 362–387.

[5] C. Blatter, *Wavelets, A Primer*, A K Peters Ltd, Natick MA, 1998.

[6] W. Cai and J. Wang, *Adaptive Multiresolution Collocation Methods for Initial-Boundary Value Problems of Nonlinear PDEs*, SIAM Journal on Numerical Analysis **33** (1996), Issue 3, 937–970.

[7] L.J. Campbell, *Wave-Mean-Flow Interactions in a Forced Rossby Wave Packet Critical Layer*, Studies in Applied Mathematics **112** (2004), Issue 1, 39–85.

[8] L.J. Campbell and S.A. Maslowe, *Forced Rossby Wave Packets in Baratropic Shear Flows with Critical Layers*, Dynamics of Atmospheres and Oceans **28** (1998), 9–37.

[9] W. Dahmen, A. Kurdila and P. Oswald, *Multiscale Wavelet Methods, Volume 6*, Academic Press, Toronto, 1997.

[10] I. Daubechies, *Ten Lectures on Wavelets*, SIAM, Philadelphia, PA, 1992.

[11] M.O. Dominigues, S.M. Gomes and L.M.A. Diaz, *Adaptive Wavelet Representation and Differentiation on Block-Structured Grids*, Applied Numerical Mathematics **47** (2003), 421–437.

[12] G. Erlebacher, M.Y. Hussaini and L.M. Jameson, *Wavelets: Theory and Applications*, Oxford University Press, New York, 1996.

[13] M. Farge and K. Schneider, *Coherent Vortex Simulation (CVS), a Semi-Deterministic Turbulence Model Using Wavelets*, Flow, Turbulence and Combustion **66** (2001), 393–426.

[14] E. Hernández and G. Weiss, *A First Course on Wavelets*, CRC Press, New York, 1996.

[15] J.S. Hesthaven and L.M. Jameson, *A Wavelet Optimized Adaptive Multi-domain Method*, Journal of Computational Physics **145** (1998), 280–296.

[16] A. Iserles, *A First Course in the Numerical Analysis of Differential Equations*, Cambridge University Press, New York, 2004.

[17] L. Jameson, *A Wavelet-Optimized, Very High Order Adaptive Grid and Order Numerical Method*, SIAM Journal on Scientific Computing **19** (1998), Issue 6, 1980–2013.

[18] P.K. Kundu and I.M. Cohen, *Fluid Mechanics*, Third Edition, Elsevier Academic Press, New York, 2004.

[19] S. Léonard, M. Terracol and P. Sagaut, *A Wavelet-Based Adaptive Mesh Refinement Criterion for Large-Eddy Simulation*, Journal of Turbulence **7** (2006).

[20] M. Mehra and N. K.-R. Kevlahan, *An Adaptive Multilevel Wavelet Solver for Elliptic Equations on and Optimal Spherical Geodesic Grid*, SIAM Journal on Scientific Computing **30** (2008), 3073-3086.

[21] S. Müller and Y. Stiriba, *A Multilevel Finite Volume Method with Multiscale-Based Grid Adaptation for Steady Compressible Flows*, Journal of Computational and Applied Mathematics **227** (2009), 223–233.

[22] H.L. Royden, *Real Analysis*, Prentice Hall Canada Inc., Toronto, 1988.

[23] K. Schneider and O.V. Vasilyev, *Wavelet Methods in Computational Fluid Dynamics*, Annual Review of Fluid Mechanics **42** (2010), 473–503.

[24] E. Stade, *Fourier Analysis*, J Wiley and Sons, Hoboken, NJ, 2005.

[25] R.B. Stull, *Meteorology for Scientists and Engineers*, Second Edition, Brooks/Cole, Pacific Grove, CA, 2000.

[26] K. Urban, *Wavelet Methods for Elliptic Partial Differential Equations*, Oxford University Press, New York, 2009.

[27] O. V. Vasilyev and N. K.-R. Kevlahan, *An Adaptive Multilevel Wavelet Collocation Method for Elliptic Problems*, Journal of Computational Physics **206** (2005), 412–431.

[28] O.V. Vasilyev and S. Paolucci, *A Dynamically Adaptive Multilevel Wavelet Collocation Method for Solving Partial Differential Equations in a Finite Domain*, Journal of Computational Physics, **125** (1996), 498–512.