ON I. WAVELET BASED APPROACH TO NUMERICAL SOLUTION OF NONLINEAR PARTIAL DIFFERENTIAL EQUATIONS AND II. NONLINEAR WAVES IN FULLY DISCRETE DYNAMICAL SYSTEMS

by

JAMES MATTHEW KEISER

BS, Clarkson University, 1988

MS, Clarkson University, 1989

MS, University of Colorado at Boulder, 1994

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Program in Applied Mathematics

1995

This thesis for the Doctor of Philosophy degree by

James Matthew Keiser

has been approved for the

Program in

Applied Mathematics

by

Gregory Beylkin

Mark J. Ablowitz

William L. Briggs

James H. Curry

Harvey Segur

Date _____

Keiser, James Matthew (Ph. D., Applied Mathematics)

On I. Wavelet Based Approach to Numerical Solution of Nonlinear Partial Differential Equations

and

II. Nonlinear Waves in Fully Discrete Dynamical Systems

Thesis directed by Professor Gregory Beylkin and Professor Mark J. Ablowitz

The first part of this work involves the numerical solution of nonlinear partial differential equations of the form $u_t = \mathcal{L}u + \mathcal{N}f(u)$ where $\mathcal{L}$ and $\mathcal{N}$ are linear differential operators and $f(u)$ is a nonlinear function. The approach is to project the solution u and the operators $\mathcal{L}$ and $\mathcal{N}$ into a wavelet basis. Vanishing moments of the basis functions permit a sparse representation of the solution and operators. Using these sparse representations fast, adaptive algorithms are developed that may be used to solve the evolution equations. These algorithms apply operators to functions and evaluate nonlinear functions. For a wavelet representation of u that contains N_s significant coefficients, the algorithms update the solution using $O(N_s)$ operations. The approach is applied to a number of examples and numerical results are given.

The second part of this work is concerned with the nonlinear wave propagation associated with certain nonlinear partial-difference equations having their dependent variables in finite fields. Solutions of such equations exhibit behavior which are discrete analogues of continuous phenomena, i.e. solitons and their elastic interactions. The main result is a time-reversible and multi-state generalization of an irreversible two-state model previously introduced. A new rule for explicitly constructing special periodic solutions of these finite-difference equations for the reversible k-state rules is given. Detailed consideration of the reversible aspects of these automata are discussed.

My life is a sequence of zeros and ones.

Sometimes it's a zero, sometimes it's a one.

CONTENTS

CHAPTER 1

GENERAL INTRODUCTION

This Thesis discusses two distinct topics: wavelet-based solutions of nonlinear partial differential equations, and nonlinear discrete dynamical systems.

In Chapter 2 fast, adaptive numerical solution of initial boundary value problems of the form

$$u_t = \mathcal{L}u + \mathcal{N}f(u) \tag{1.0.1}$$

with

$$u(x,0) = u_0(x) \qquad 0 \le x \le 1 \tag{1.0.2}$$

$$u(0,t) = u(1,t) \qquad 0 \le t \le T, \tag{1.0.3}$$

is considered where the operators $\mathcal{L}$ and $\mathcal{N}$ are independent of t and the function $f(u)$ is typically nonlinear. Such equations may have solutions possessing either smooth or shock-like behavior or both. Examples of such evolution equations include the diffusion (or heat) equation, reaction-diffusion equations (which have solutions that may blow-up), and equations which model both stationary and moving shocks (e.g. Burgers' equation). Although multi-dimensional problems are not specifically address in this work, it should be noted that the Navier-Stokes equations may also be written in the form (1.0.1), see e.g. [40].

The new approach described in Chapter 2 uses expansions of functions and operators in wavelet bases, which allows one to combine some of the desirable

features found in finite-difference methods, spectral methods and front-tracking or adaptive grid methods into a collection of efficient, generic algorithms. These algorithms take advantage of the fact that wavelet expansions may be viewed as a localized Fourier analysis with multiresolution structure that is automatically adaptive to both smooth and shock-like behavior of the solution of (1.0.1). In smooth regions few wavelet coefficients are needed and in singular regions large variations in the function require more wavelet coefficients. The theoretical analysis of such functions by wavelet methods is well-understood [42, 53, 54]. However, the use of wavelet expansions of functions and operators for numerical purposes requires the development of new algorithms, which are introduced in this Thesis.

Chapter 2 is outlined as follows. We begin with a discussion of the background of and motivation for our studies. We then reformulate the partial differential equation (1.0.1) as a nonlinear integral equation via the semigroup method and develop a method for approximating the integrals to an arbitrary order of accuracy. We continue with the development of two new fundamental algorithms required for the numerical solution of the nonlinear integral equation. In particular we develop fast, adaptive algorithms for (1) multiplying operators and functions, and (2) computing nonlinear functions. Chapter 2 concludes with several numerical examples. We note that our developments use expansions of functions and operators in wavelet bases, and that Appendix A provides a review of the background material used in our approach.

In Chapter 3 we discuss nonlinear fully discrete dynamical systems expressed in terms of nonlinear finite-difference equations having state variables

taken from finite fields. We will refer to this class of nonlinear discrete dynamical systems as cellular automata. Cellular automata have been used to model fluid flows including the Navier-Stokes equations (see e.g. [4]-[7]) and Burgers' equation (see e.g. [8] and [9]). Other physical phenomena modeled by cellular automata include reaction-diffusion, biological and chemical-reactive systems (see e.g. [2, 3]).

In Chapter 3 we investigate a particular cellular automaton known as the Parity Rule Filter Automata (PRFA), [14]. This automaton is of interest because given localized initial data, subsequent evolution of the automaton yields a remarkable number and variety of stable, coherent structures or 'particles'. Moreover, a number of these particles exhibit behavior which can be viewed as a discrete analogue of soliton interactions. For a review of solitons and their importance in physical systems see [16, 17] and the references therein. The model introduced in [14] has the following two features that are addressed in Chapter 3. First the automaton operates on field variables taken from a set of only two elements, e.g. $\{0, 1\}$. Second and perhaps of more fundamental importance is the inherent time irreversibility of the original model, i.e. information may be lost during the evolution of the PRFA. The purpose of Chapter 3 is to describe a multi-state and time-reversible generalization of the Parity Rule Filter Automata.

Chapter 3 is outlined as follows. We begin by setting the notation associated with our study of nonlinear fully discrete dynamical systems. We then review the Parity Rule Filter Automata, and the equivalent and analytically powerful formulation call the Fast Rule Theorem. We then discuss a nonlinear difference equation that is equivalent to the PRFA and that allows us to

immediately generalize the PRFA to multi-states. Moreover, the nonlinear difference equation allows us to identify the source of the time-irreversibility of the PRFA and to formulate a time-reversible automata. Chapter 3 continues with a description of the construction of special periodic solutions of the nonlinear difference equation. We conclude Chapter 3 by describing the qualitative similarities between the new multi-state, time-reversible automaton and the propagation of nonlinear waves through multilayered media. The work contained in this Chapter has been previously published [24] and referenced, see e.g. [26, 27].

CHAPTER 2

WAVELET BASED SOLUTIONS OF NONLINEAR PARTIAL DIFFERENTIAL EQUATIONS

This Chapter is concerned with the fast, adaptive numerical solution of nonlinear partial differential equations having solutions with both smooth and shock-like behavior. The new approach described in this Chapter uses expansions of functions and operators in wavelet bases, which allow us to combine the desirable features of finite-difference approaches, spectral methods and front-tracking or adaptive grid approaches into a collection of efficient, generic algorithms. These algorithms take advantage of the fact that wavelet expansions may be viewed as a localized Fourier analysis with multiresolution structure that automatically distinguishes between smooth and shock-like behavior. In smooth regions few wavelet coefficients are needed and in singular regions large variations in the function require more wavelet coefficients. The theoretical analysis of such functions by wavelet methods is well-understood, [42, 54, 53]. However the use of wavelet expansions of functions and operators for numerical purposes requires the development of new algorithms, which we introduce in this Chapter.

The algorithmic complexity of our approach is proportional to the number of significant coefficients in the wavelet expansions of functions and operators. The wavelet-expansion approach to solving such equations is essentially a projection method. In projection methods the goal is to use the fewest number of

expansion coefficients to represent a function, since this leads to efficient numerical computations. The number of coefficients required to represent a function expanded in a Fourier series (or similar expansions based on the eigenfunctions of a differential operator) depends on the most singular behavior of the function. The functions we are interested in are solutions of partial differential equations that have regions of smooth, non-oscillatory behavior interrupted by a number of well-defined localized shocks or shock-like structures. Therefore expansions of these solutions based upon the eigenfunctions of differential operators require a large number of terms due to the singular regions. Alternately a localized representation of the solution, typified by front-tracking or adaptive grid methods, may be employed in order to distinguish between smooth and shock-like behavior.

There are several reasons for investigating the use of wavelet expansions in the development of new numerical algorithms. Let the wavelet transform of a function consist of N_s significant coefficients near shock-like structures. Our goal is to design fully adaptive algorithms that perform numerical computations in $O(N_s)$ operations, using only the significant wavelet coefficients. In other words, we will look for a general approach that has the desirable properties of specialized adaptive algorithms. Another reason for investigating the use of wavelet-basis expansions in numerical methods is that in wavelet coordinates differential operators may be preconditioned by a diagonal matrix, [46]. Moreover, a large class of operators, namely Calderón-Zygmund and pseudo-differential operators, are sparse in wavelet bases. These observations alone make a good case for developing new numerical algorithms for computing in wavelet bases.

Using properties of the wavelet representation of functions and operators we design two new algorithms for computing solutions of partial differential

equations. Specifically, these two algorithms are: 1. adaptive application of operators to functions, and 2. adaptive pointwise product of functions. In any basis-expansion approach these algorithms are necessary ingredients of any fast, adaptive numerical scheme for computing solutions of partial differential equations. The wavelet representation of a class of operators, which includes differential operators and Hilbert transforms, has a vanishing-moment property described in Section 2.2.2. We will use this property to develop a generic, efficient, adaptive algorithm for applying differential operators to functions using only $O(N_s)$ significant wavelet coefficients. We have also developed an adaptive algorithm for computing the pointwise product of functions, again using only $O(N_s)$ significant wavelet coefficients.

We apply our approach to the problem of computing numerical solutions of initial and boundary value problems of the form

$$u_t = \mathcal{L}u + \mathcal{N}f(u) \tag{2.0.1}$$

with

$$u(x,0) = u_0(x) \qquad 0 \le x \le 1 \tag{2.0.2}$$

$$u(0,t) = u(1,t) \qquad 0 \le t \le T. \tag{2.0.3}$$

The evolution equation (2.0.1) is written in terms of a linear part, $\mathcal{L}u$, and a nonlinear part, $\mathcal{N}f(u)$, where the operators $\mathcal{L}$ and $\mathcal{N}$ are constant-coefficient, differential operators that do not depend on time t, and the function $f(u)$ is typically nonlinear, e.g. $f(u) = u^p$. The class of problems we are interested in has initial conditions $u_0(x)$ which are bounded and may or may not have discontinuities.

Equations such as (2.0.1) can describe the buildup and propagation of shocks and arise in a variety of models of physical phenomena (see e.g. [31]). Examples of such evolution equations in 1+1 dimensions include reaction-diffusion equations which are characterized by blowing-up solutions, e.g.

$$u_t = \nu u_{xx} + u^p \qquad p > 1 \quad \nu > 0, \tag{2.0.4}$$

and models of stationary or moving shocks, e.g. Burgers' equation

$$u_t + u u_x = \nu u_{xx} \qquad \nu > 0, \tag{2.0.5}$$

[32]. An additional, trivial example of equation (2.0.1) is the classical diffusion (or heat) equation

$$u_t = \nu u_{xx} \qquad \nu > 0. \tag{2.0.6}$$

Although we do not address multi-dimensional problems, we note that the Navier-Stokes equations may also be written in the form (2.0.1). A one-dimensional model that may be thought of as a prototype for the Navier-Stokes equation is

$$u_t = \mathcal{H}(u)u, \tag{2.0.7}$$

where $\mathcal{H}(\cdot)$ is the Hilbert transform, [40]. The presence of the Hilbert transform introduces a long range interaction which models that found in the Navier-Stokes equations. Even though the algorithms we will develop are for one-dimensional problems, we take special care that they generalize properly to several dimensions so that we can address these problems in the future.

Our scheme, which uses expansions in wavelet bases, has desirable features of adaptive grid or front-tracking algorithms and pseudo-spectral methods. Our wavelet based expansion approach is similar to the Fourier expansions of

spectral methods. Due to the vanishing moments of the wavelet basis functions, we know that the wavelet transform of a function automatically places significant coefficients in a neighborhood of large gradients present in the function. This is in direct contrast with adaptive grid methods, which refine the grid based on somewhat ad hoc procedures (see e.g. [37]). This combination of basis expansion and adaptive thresholding is the foundation for our fast, adaptive approach.

In order to take advantage of this "adaptive transform" scheme and compute solutions of (2.0.1) in wavelet bases using order N_s operations, we must have at our disposal the two basic adaptive algorithms referred to above: – application of an operator to a function and pointwise products of functions. We rewrite the partial differential equation (2.0.1) in a form amenable to these two algorithms by first applying the semigroup method in order to rewrite the differential equation as a nonlinear integral equation. We then use quadrature formulas to discretize the integral equation in time in order to obtain a system to which we can apply our adaptive algorithms.

Several numerical techniques have been developed to compute approximate solutions of equations such as (2.0.1). These techniques include finite-difference, pseudo-spectral and adaptive grid methods (see e.g. [29, 60]). An important first step in solving equation (2.0.1) by any of these methods is the choice of time discretization. Explicit schemes (which are easiest to implement) may require prohibitively small time steps, usually because of diffusion terms in the evolution equation. On the other hand, implicit schemes allow for larger time steps but require solving a system of equations at each time step and for this reason are somewhat more difficult to implement in an efficient manner. In our approach we have used an implicit time integrator.

The main difficulty in computing solutions of equations like (2.0.1) is the resolution of shock-like structures. Straightforward refinement of a finite-difference scheme easily becomes computationally excessive. The specialized front-tracking or adaptive grid methods require some criteria to perform local grid refinement. Usually in such schemes these criteria are chosen in an ad hoc fashion (especially in multiple dimensions) and are generally based on the amplitudes or local gradients in the solution.

The pseudo-spectral method usually splits the evolution equation into linear and nonlinear parts and updates the solution by superposing the linear contribution, calculated in Fourier space, and the nonlinear contribution, calculated in physical space, [59, 60]. Pseudo-spectral schemes have the advantages that they are easy to understand analytically, spectrally accurate, and relatively straightforward to implement. However pseudo-spectral schemes have a disadvantage in that the superposition of the linear and nonlinear contributions must be done in the same domain, either physical space or Fourier space. This difficulty becomes significant when one attempts to compute solutions of differential equations in multiple dimensions. Moreover, the Fourier transform of intermediate solutions exhibiting shock-like behavior possesses significant frequency contributions across the entire spectrum as the shock becomes more and more pronounced.

Our wavelet approach is comparable to spectral methods in their accuracy, and the automatic placement of significant wavelet coefficients in regions of large gradients parallels general adaptive grid approaches. These observations make a good case for developing numerical methods for computing solutions of partial differential equations which are alternatives to the traditional methods.

This Chapter is outlined as follows. In Section 2.1 we use the semi-group method to write the differential equation (2.0.1) as a nonlinear integral equation and introduce an algorithm for approximating the integral, in terms of certain operators, to any order of accuracy desired. Section 2.2 is concerned with the construction of and calculations with the operators appearing in the quadrature formulas derived in Section 2.1. Specifically we describe a means for constructing the wavelet representation of the operators appearing in approximations introduced in Section 2.1, derive the vanishing-moment property of these operators and describe a fast, adaptive algorithm for applying these operators to functions expanded in a wavelet basis. In Section 2.3 we introduce a new adaptive algorithm for computing the pointwise product of functions expanded in a wavelet basis. The algorithms described in Sections 2.2 and 2.3 are based on the wavelet representation of operators and functions which is reviewed in Appendix A. In Section 2.4 we illustrate the use of these algorithms by providing the results of numerical experiments and comparing them with the exact solutions. Finally in Section 2.5 we draw a number of conclusions based on our results and indicate directions of further investigation.

2.1 The Semigroup Approach and Quadratures

This Section describes our approach for reformulating the partial differential equation (2.0.1) as a nonlinear integral equation. Using the semigroup method we recast the partial differential equation as a nonlinear integral equation in time. We approximate the integrals to arbitrary orders of accuracy by quadratures with operator valued coefficients. These operators have wavelet representations with a number of desirable properties described in Section 2.2.1 and 2.2.2.

2.1.1 The Semigroup Approach The semigroup method is a well-known analytical tool which may be used to convert partial differential equations to nonlinear integral equations and to calculate estimates associated with the behavior of their solutions (see e.g. [30, 39]). The integral solution of the initial value problem

$$u_t = \mathcal{L}u + \mathcal{N}f(u) \tag{2.1.1}$$

$$u(x, t_0) = u_0(x) \tag{2.1.2}$$

is given by

$$u(x,t) = e^{(t-t_0)\mathcal{L}}u_0(x) + \int_{t_0}^{t} e^{(t-\tau)\mathcal{L}}\mathcal{N}f(u(x,\tau))d\tau, \tag{2.1.3}$$

where $\mathcal{L}$ and $\mathcal{N}$ are time-independent, constant coefficient differential operators. Expressing solutions of (2.1.1) in the form (2.1.3) allows one to, among other things, prove existence and uniqueness of solutions, compute estimates of the magnitude of solutions, verify dependence on initial and boundary data, and perform asymptotic analysis of the solution, see e.g. [39].

We are interested in using equation (2.1.3) as a starting point for an efficient numerical algorithm. As far as we know, the semigroup method has

had limited use in numerical calculations. A significant difficulty in designing numerical algorithms based directly on the solution (2.1.3) is that the operators appearing in (2.1.3) are not sparse (i.e. the matrices representing these operators are dense). We show in Sections 2.2.1 and 2.2.2 that in the wavelet system of coordinates these operators are sparse and have properties that we use to design fast, adaptive numerical algorithms. In the next Section we describe an approach to approximating the integral in (2.1.3) to an arbitrary order of accuracy.

2.1.2 Quadratures As it follows from (2.1.3) we have to consider approximating integrals of the form

$$I(x,t) = \int_{t_0}^{t} e^{(t-\tau)\mathcal{L}} \mathcal{N} f(u(x,\tau)) d\tau. \tag{2.1.4}$$

As mentioned earlier, the differential operator $\mathcal{N}$ is assumed to be independent of t and the function $f(u)$ is nonlinear. For example, in the case of Burgers' equation $\mathcal{N} = \frac{\partial}{\partial x}$ and $f(u) = \frac{1}{2}u^2$, so that the integrand $\mathcal{N} f(u) = uu_x$ appears as products of u and its derivatives. Therefore we seek approximations to integrals of the form

$$I(t) = \int_{t_0}^{t} e^{(t-\tau)\mathcal{L}} u(\tau) v(\tau) d\tau, \tag{2.1.5}$$

where we have suppressed the explicit x dependence. In order to derive an approximation to this integral, we partition the interval of integration $[t_0, t]$ into m equal subintervals with grid points at $t_i = t_0 + i\Delta t$, for $i = 0, 1, \ldots, m$. Let us denote $u(t_i)$ and $v(t_i)$ by u_i and v_i, respectively.

We seek an approximation to (2.1.5) of the form

$$I(t) = \hat{I}(t) + O(\Delta t^{m+1}) \tag{2.1.6}$$

where

$$\hat{I}(t) = \sum_{i,j=0}^{m} c_{i,j} u_i v_j, \tag{2.1.7}$$

and where the coefficients $c_{i,j}$ are time-independent, operator-valued functions of the operator $\mathcal{L}$. Observe that we have included in (2.1.7) cross terms of the form $u_i v_j$, $i \neq j$; usual quadrature approximations, e.g. the trapezoidal rule, typically begin with (2.1.5) and only use products $u_i v_i$. In this Section we describe a procedure for determining the coefficients $c_{i,j}$ which, for a given order of accuracy, reduce the number of product terms of the form $u_i v_j$ in (2.1.7) from $(m+1)^2$ to $m+1$.

The operator coefficients $c_{i,j}$ are determined by comparing (2.1.6) and (2.1.7) with a scheme of known order of accuracy constructed using Lagrange polynomial approximations in t of the functions $u(t)$ and $v(t)$. Namely, substituting into (2.1.5) approximations of the form

$$u(t) = \sum_{i=0}^{m} L_i(t) u_i + O(\Delta t^{m+1}) \tag{2.1.8}$$

$$v(t) = \sum_{i=0}^{m} L_i(t) v_i + O(\Delta t^{m+1}) \tag{2.1.9}$$

where

$$L_i(t) = \prod_{\substack{k=0 \\ k \neq i}}^{m} \frac{(t - t_k)}{(t_i - t_k)} \tag{2.1.10}$$

yields the following approximation to $I(t)$

$$I(t) = \overline{I}(t) + O(\Delta t^{m+2}) \tag{2.1.11}$$

where

$$\overline{I}(t) = \int_{t_0}^{t} e^{(t-\tau)\mathcal{L}} \sum_{i,j=0}^{m} L_i(\tau) L_j(\tau) u_i v_j d\tau. \tag{2.1.12}$$

Interchanging summation and integration gives

$$\overline{I}(t) = \sum_{i,j=0}^{m} f_{i,j} u_i v_j, \tag{2.1.13}$$

where

$$f_{i,j} = \int_{t_0}^{t} e^{(t-\tau)\mathcal{L}} L_i(\tau) L_j(\tau) d\tau \tag{2.1.14}$$

can be computed explicitly.

The coefficients $c_{i,j}$ in (2.1.7) are determined by straightforward expansion techniques. Namely, one first fixes the order m of the approximations (2.1.11) and (2.1.6) and then compares the coefficients of powers of Δt in these two approximations. This leads to a system of equations for determining the operator coefficients $c_{i,j}$ that, in general, has more than one solution. We then choose a solution of this system of equations which consists of $m+1$ non-zero coefficients $c_{i,j}$. Substituting these $c_{i,j}$ into equation (2.1.6) yields an approximation to $I(t)$ which is $O(\Delta t^m)$ accurate and consists of $m+1$ terms of the form $u_i v_j$. Thus, we arrive at an approximation to the integral (2.1.5) which is of the same order of accuracy as the Lagrange approximation (2.1.13) yet uses significantly fewer terms.

We conclude this Section by providing the results of two examples of this procedure applied to Burgers' equation

$$u_t = \nu u_{xx} - u u_x \tag{2.1.15}$$

wherein $\mathcal{L} = \nu \frac{\partial^2}{\partial x^2}$ and $\mathcal{N} = \frac{\partial}{\partial x}$, and $f(u) = \frac{1}{2}u^2$. The integral corresponding to (2.1.5) which we wish to approximate is given by

$$I(t) = \int_{t_0}^{t} e^{(t-\tau)\mathcal{L}} u(\tau) u_x(\tau) d\tau. \tag{2.1.16}$$

This procedure results in approximations to (2.1.16) which for $m = 1$ are of the form

$$I(t) = \tfrac{1}{2}\mathcal{O}_{\mathcal{L},1}\left(u(t_0)u_x(t_0) + u(t_1)u_x(t_1)\right) + O((\Delta t)^2), \tag{2.1.17}$$

which is equivalent to the standard trapezoidal rule, or

$$I(t) = \tfrac{1}{2}\mathcal{O}_{\mathcal{L},1}\left(u(t_0)u_x(t_1) + u(t_1)u_x(t_0)\right) + O((\Delta t)^2), \tag{2.1.18}$$

which we actually use in our numerical experiments. The operator $\mathcal{O}_{\mathcal{L},m}$ is given by

$$\mathcal{O}_{\mathcal{L},m} = (\mathbf{I} - e^{m\Delta t\mathcal{L}})\mathcal{L}^{-1}, \tag{2.1.19}$$

where $\mathbf{I}$ is the identity operator. For $m = 2$ our procedure yields an analogue of Simpson's rule

$$I(t) = \sum_{i=0}^{2} c_{i,i}u(t_i)u_x(t_i) + O((\Delta t)^3) \tag{2.1.20}$$

where

$$c_{0,0} = \tfrac{1}{6}\mathcal{O}_{\mathcal{L},2} - \tfrac{1}{3}\mathcal{L} \tag{2.1.21}$$

$$c_{1,1} = \tfrac{2}{3}\mathcal{O}_{\mathcal{L},2} \tag{2.1.22}$$

$$c_{2,2} = \tfrac{1}{6}\mathcal{O}_{\mathcal{L},2} + \tfrac{1}{3}\mathcal{L} \tag{2.1.23}$$

The detailed calculations leading to these expressions consist of a sequence of simple but tedious algebraic manipulations and we have employed *Mathematica* to perform these computations. The details of these derivations and program listings for each of these examples can be found in Appendix B.

2.2 Wavelet Representations of Operator Functions

In this Section we recall two natural representations of operators in wavelet bases and construct the wavelet representations of the operators appearing in the approximations to the integral (2.1.5), see e.g. equations (2.1.17) and (2.1.20). In Section 2.2.1 we show how to compute the non-standard form of the operator functions appearing in (2.1.5). We then establish in Section 2.2.2 the vanishing-moment property of the wavelet representation of these operators. Finally, in Section 2.2.3 we describe a fast, adaptive algorithm for applying operators to functions in the wavelet system of coordinates. In Appendix A we provide a detailed review of material that may be required to understand this Section. This Appendix reviews the terminology of multiresolution analysis and the representation of functions and operators in wavelet bases.

2.2.1 The Non-Standard Form of Operator Functions In this Section we construct the non-standard forms (NS-forms) of functions of the differential operator ∂_x. Following [48] we introduce two approaches for computing the NS-forms of operator functions that are characterized by either (i) computing the projection of the operator function on $\mathbf{V}_0$, e.g.

$$P_0 f(\partial_x) P_0, \tag{2.2.1}$$

or, (ii) computing the function of the projection of the operator, e.g.

$$f(P_0 \partial_x P_0). \tag{2.2.2}$$

The difference between these two approaches depends on the properties of $|\hat{\varphi}(\xi)|^2$ as a cutoff function, where $\varphi(x)$ is the scaling function associated with a wavelet basis. It might be convenient to use either (2.2.1) or (2.2.2) in applications.

The operator functions we are interested in are those appearing in solutions of the partial differential equation (2.1.1). For example, solutions of Burgers' equation can be approximated to order Δt^2 by

$$\begin{aligned} u(x,t+\Delta t) &= e^{\Delta t\mathcal{L}}u(x,t) - \\ &\quad \tfrac{1}{2}\mathcal{O}_{\mathcal{L},1}\left[u(x,t)\partial_x u(x,t+\Delta t) + u(x,t+\Delta t)\partial_x u(x,t)\right] \end{aligned} \tag{2.2.3}$$

where $\mathcal{L} = \nu\partial_x^2$ and $\mathcal{O}_{\mathcal{L},1}$ is given by (2.1.19). Therefore, we are interested in constructing the NS-forms of the operator functions

$$f(\partial_x) = e^{\Delta t\mathcal{L}} \tag{2.2.4}$$

and

$$f(\partial_x) = \mathcal{O}_{\mathcal{L},1} = \left(\mathbf{I} - e^{\Delta t\mathcal{L}}\right)\mathcal{L}^{-1}, \tag{2.2.5}$$

for example. In the following we assume that the function f is analytic. In computing solutions of (2.1.1), via (2.2.3), we can precompute the non-standard forms of the operator functions and apply them when necessary.

We note that if the operator function f is homogeneous of degree m (e.g. $m = 1$ and 2 for the first and second derivative operators) then the coefficients appearing in the NS-form satisfy relationships of the form

$$\left.\begin{aligned} \alpha_l^j &= 2^{-mj}\alpha_l^0 \\ \beta_l^j &= 2^{-mj}\beta_l^0 \\ \gamma_l^j &= 2^{-mj}\gamma_l^0 \\ s_l^j &= 2^{-mj}s_l^0, \end{aligned}\right\} \tag{2.2.6}$$

see (A.4.40) and (A.4.41). Thus the coefficients on scale $j = 1, 2, \ldots, J$ are scaled versions of the coefficients on scale $j = 0$. On the other hand, if the operator function f is not homogeneous then we compute $s_{k,k'}^0$ and compute

the coefficients $\alpha^j_{k,k'}, \beta^j_{k,k'}$, and $\gamma^j_{k,k'}$ via equations (A.4.41) for each scale $j = 1, 2, \ldots, J$. We note that if f is a convolution operator then the formulas for $s^0_{k-k'}$ are considerably simplified (see [45]).

We first describe computing the NS-form of an operator function by projecting the operator function into the wavelet basis via (2.2.1). To compute the coefficients

$$s^j_{k,k'} = 2^{-j} \int_{-\infty}^{+\infty} \varphi(2^{-j}x - k) f(\partial_x) \varphi(2^{-j}x - k') dx \tag{2.2.7}$$

let us consider

$$f(\partial_x)\varphi(2^{-j}x - k') = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} f(-i\xi 2^{-j}) \hat{\varphi}(\xi) e^{-i\xi k'} e^{i2^{-j}x\xi} d\xi, \tag{2.2.8}$$

where $\hat{\varphi}(\xi)$ is the Fourier transform of $\varphi(x)$

$$\hat{\varphi}(\xi) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{+\infty} \varphi(x) e^{ix\xi} dx. \tag{2.2.9}$$

Substituting (2.2.8) into (2.2.7) we obtain

$$s^j_{k,k'} = 2^{-j} \int_{-\infty}^{\infty} \hat{\varphi}(\xi) f(-i\xi 2^{-j}) e^{-i\xi k'} \left[\frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \varphi(2^{-j}x - k) e^{i\xi 2^{-j}x} dx \right] d\xi. \tag{2.2.10}$$

Noting that $s^j_{k,k'} = s^j_{k-k'}$ we arrive at

$$s^j_l = \int_{-\infty}^{+\infty} f(-i\xi 2^{-j}) |\hat{\varphi}(\xi)|^2 e^{i\xi l} d\xi. \tag{2.2.11}$$

We can evaluate (2.2.11) by setting

$$s^j_l = \int_{-\infty}^{+\infty} \sum_{k \in \mathbb{Z}} f(-i2^{-j}(\xi + 2\pi k)) |\hat{\varphi}(\xi + 2\pi k)|^2, \tag{2.2.12}$$

or

$$s^j_l = \int_0^{2\pi} g(\xi) e^{i\xi l} d\xi, \tag{2.2.13}$$

where

$$g(\xi) = \sum_{k\in\mathbb{Z}} f(-i2^{-j}(\xi + 2\pi k))|\hat{\varphi}(\xi + 2\pi k)|^2. \tag{2.2.14}$$

We now observe that for a given accuracy ϵ the function $|\hat{\varphi}(\xi)|^2$ acts as a cutoff function in the Fourier domain, i.e. $|\hat{\varphi}(\xi)|^2 < \epsilon$ for $|\xi| > \eta$ for some $\eta > 0$. Therefore the infinite sum in (2.2.12) can be approximated to within ϵ by the finite sum

$$\tilde{g}(\xi) = \sum_{k=-K}^{K} f(-i2^{-j}(\xi + 2\pi k))|\hat{\varphi}(\xi + 2\pi k)|^2, \tag{2.2.15}$$

for large enough K. Using (2.2.15) (in place of $g(\xi)$) in (2.2.13) and uniformly discretizing the interval $[0, 2\pi]$ into N subintervals $[\xi_n, \xi_{n+1}]$ for $\xi_n = 2\pi n/N$ and $n = 0, 1, \ldots N-1$, we obtain an approximation to the coefficients s_l^j,

$$\tilde{s}_l^j = \tfrac{1}{N} \sum_{n=0}^{N-1} \tilde{g}(\xi_n) e^{i\xi_n l}. \tag{2.2.16}$$

The coefficients $\tilde{s}_l^j$ can then be computed by applying the fast Fourier transform to the sequence $\{\tilde{g}(\xi_n)\}$ computed via (2.2.15).

Let us now describe computing the NS-form of an operator function via (2.2.2). In this case we can use the discrete Fourier transform to diagonalize the differential operator appearing in (2.2.2). Starting with the wavelet representation on $\mathbf{V}_0$ of the discretization of ∂_x, we can write the eigenvalues explicitly as

$$\lambda_k = s_0 + \sum_{l=1}^{L} (s_l e^{2\pi i \frac{kl}{N}} + s_{-l} e^{-2\pi i \frac{kl}{N}}), \tag{2.2.17}$$

where the wavelet coefficients of the derivative, $s_l = s_l^0$, are defined by (A.4.38). Since

$$f(\mathcal{A}) = \mathcal{F} f(\Lambda) \mathcal{F}^{-1}, \tag{2.2.18}$$

where Λ is a diagonal matrix of the eigenvalues of $\mathcal{A}$ and $\mathcal{F}$ is the Fourier transform, [39], we compute $f(\lambda_k)$ and apply the inverse Fourier transform to

the sequence $f(\lambda_k)$

$$s_l^0 = \sum_{k=1}^{N} f(\lambda_k) e^{2\pi i \frac{(k-1)(l-1)}{N}}, \tag{2.2.19}$$

to arrive at the projection of the operator functions $f(\partial x)$ on the subspace $\mathbf{V}_0$, i.e. the wavelet coefficients s_l^0. The remaining elements of the non-standard form are then recursively computed using equations (A.4.41).

2.2.2 Vanishing Moments of the B^j Blocks In this Section we establish the vanishing-moment property of the B^j blocks of the NS-form representation of differential operators and the Hilbert transform. Moreover, we will establish this result for the NS-form representation of the operator functions described in Section 2.2.1. This property of the B^j-blocks is discussed in Meyer [55]. In Section 2.2.3 we use the vanishing-moment property to design an adaptive algorithm for multiplying the NS-form of an operator and the wavelet expansion of a function.

Differential Operators For differential operators we show that the rows of the B^j blocks of the NS-form have M vanishing moments, i.e.,

$$\sum_{l=-L}^{L} l^m \beta_l = 0 \tag{2.2.20}$$

for $m = 0, 1, 2, \ldots, M-1$ and where $2L-1$ is the length of each row of the B^j block. We recall the following definition of $\beta_l = \beta_l^0$

$$\beta_l = \int_{-\infty}^{+\infty} \psi(x-l) \frac{d}{dx} \varphi(x) dx, \tag{2.2.21}$$

(see e.g. (A.4.38)) and use the fact (see [54]) that for $0 \leq m \leq M-1$

$$\sum_{l=-\infty}^{+\infty} l^m \varphi(x-l) = P_m(x), \tag{2.2.22}$$

where $P_m(x)$ is a polynomial of degree m. Since $\beta_l = 0$ for $|l| \geq L_\beta - 1$ and using (2.2.21) and (2.2.22) we have

$$\begin{aligned}\sum_{l=-\infty}^{+\infty} l^m \beta_l &= \int_{-\infty}^{\infty} \psi(x) \frac{d}{dx} \left(\sum_{l=-\infty}^{+\infty} l^m \varphi(x+l) \right) dx \\ &= \int_{-\infty}^{\infty} \psi(x) \frac{d}{dx} P_m(x) dx \\ &= \int_{-\infty}^{\infty} \psi(x) \tilde{P}_{m-1}(x) dx \end{aligned} \tag{2.2.23}$$

where $\tilde{P}_{m-1}(x)$ is a polynomial of degree $m-1$. Since the function $\psi(x)$ is orthogonal to polynomials of degree less than M and $\tilde{P}_{m-1}(x)$ is a polynomial of degree at most $M-2$, these integrals are zero. We note that the same analysis holds for derivatives of order p as long as such derivatives exist.

Hilbert Transform In the case of the Hilbert transform

$$(\mathcal{H}f)(x) = \frac{1}{\pi} \text{p.v.} \int_{-\infty}^{\infty} \frac{f(s)}{s-x} ds, \tag{2.2.24}$$

where p.v. indicates the principle value, we show that the rows of the B^j blocks of the NS-form satisfy

$$\sum_{l=-\infty}^{+\infty} l^m \beta_l = 0. \tag{2.2.25}$$

The β_l elements of the NS-form of the Hilbert transform are given by

$$\beta_l = \int_{-\infty}^{+\infty} \psi(x-l)(\mathcal{H}\varphi)(x) dx. \tag{2.2.26}$$

Proceeding as in the case of differential operators we find

$$\begin{aligned}\sum_{l=-\infty}^{+\infty} l^m \beta_l &= \sum_{l=-\infty}^{+\infty} l^m \int_{-\infty}^{+\infty} \psi(x-l)(\mathcal{H}\varphi)(x) dx \\ &= -\sum_{l=-\infty}^{+\infty} l^m \int_{-\infty}^{+\infty} (\mathcal{H}\psi)(x) \varphi(x+l) dx \\ &= -\int_{-\infty}^{+\infty} (\mathcal{H}\psi)(x) \sum_{l=-\infty}^{+\infty} l^m \varphi(x+l) dx \\ &= -\int_{-\infty}^{+\infty} (\mathcal{H}\psi)(x) P_m(x) dx. \end{aligned} \tag{2.2.27}$$

To show that the integrals in (2.2.27) are zero we now show that $(\mathcal{H}\psi)(x)$ has at least M vanishing moments. Let us consider the generalized function

$$\int_{-\infty}^{\infty} (\mathcal{H}\psi)(x) x^m e^{i\xi x} d\xi = \frac{1}{i^m} \partial_\xi^m (\widehat{\mathcal{H}\psi})(\xi). \tag{2.2.28}$$

In the Fourier domain the Hilbert transform of the function g defined by

$$(\widehat{\mathcal{H}g})(\xi) = -i \ \mathrm{sign}(\xi) \hat{g}(\xi), \tag{2.2.29}$$

may be viewed as a generalized function, derivatives of which act on test functions $f \in \mathcal{C}_0^\infty(\mathbb{R})$ as follows

$$\begin{array}{l} < \frac{d^m}{d\xi^m} \left(-i \ \mathrm{sign}(\xi) \hat{g}(\xi) \right), f > = \\ -i \sum_{j=1}^m \binom{m}{l} f^{(j-1)}(0) g^{(m-j)}(0) + i \int_{-\infty}^{\infty} \mathrm{sign}(\xi) \ \hat{g}^{(m)}(\xi) f(\xi) d\xi. \end{array} \tag{2.2.30}$$

In order to show that $(\mathcal{H}\psi)(x)$ has M vanishing moments we recall that in the Fourier domain vanishing moments are characterized by

$$\frac{d^m}{d\xi^m} \hat{\psi}(\xi)|_{\xi=0} = 0 \quad \text{for } m = 0, 1, \ldots, M-1 \tag{2.2.31}$$

where $\hat{\psi}(\xi)$ is the Fourier transform of $\psi(x)$. Setting $g(\xi) = \hat{\psi}(\xi)$ in (2.2.30) we immediately find that the sum on the right hand side of (2.2.30) is zero. We also observe that the integrand on the right hand side of (2.2.30), i.e. $\mathrm{sign}(\xi)\hat{\psi}^{(m)}(\xi)$, is continuous at $\xi = 0$, once again because $\psi(x)$ has M vanishing moments. We can then define functions $\hat{\mathcal{W}}^{(m)}(\xi)$ by

$$\hat{\mathcal{W}}^{(m)}(\xi) = \begin{cases} -i \ \hat{\psi}^{(m)}(\xi) & \xi > 0 \\ 0 & \xi = 0 \\ i \ \hat{\psi}^{(m)}(\xi) & \xi < 0, \end{cases} \tag{2.2.32}$$

for $m = 0, 1, \ldots, M-1$, such that $\hat{\mathcal{W}}^{(m)}(\xi)$ coincides with the m-th derivative of the generalized function (2.2.29) on the test functions $f \in \mathcal{C}_0^\infty(\mathbb{R})$. Since

$\hat{\mathcal{W}}^{(m)}(\xi)$ are continuous functions for $m = 0, 1, \ldots, M-1$, we obtain instead of (2.2.28)

$$\int_{-\infty}^{\infty} (\mathcal{H}\psi)(x) x^m e^{i\xi x} dx = \hat{\mathcal{W}}^{(m)}(\xi). \tag{2.2.33}$$

Since $\hat{\mathcal{W}}^{(m)}(\xi)|_{\xi=0} = 0$ the integrals (2.2.27) are zero and (2.2.25) is proven.

Operator Functions We now consider the B^j-blocks of the NS-form representation of the operator functions described in Section 2.2.1 and show that

$$\sum_{l=-L}^{L} l^m \beta_l = 0 \tag{2.2.34}$$

for $m = 0, 1, \ldots, M-1$ where $2L-1$ is the length of each row of the B^j block. Following the development of the differential operator case, see e.g. (2.2.23), we use the definition of the $\beta^j_{k,k'}$ coefficients (A.3.33) and the fact that $\beta_l = 0$ for $|l| \geq L_\beta - 1$ in the sum (2.2.34) to obtain

$$\sum_{l=-\infty}^{\infty} l^m \beta_l = \int_{-\infty}^{+\infty} \psi(x-k) f(\partial_x) P_m(x) dx, \tag{2.2.35}$$

where $P_m(x)$ is a polynomial of degree m, see (2.2.22).

Since the function $f(\cdot)$ is an analytic function of ∂_x we can expand f in terms of its Taylor series. If $f(\partial_x)$ is then applied to $P_m(x)$ the series for $f(\partial_x)P_m(x)$ is finite and yields a polynomial of degree less than or equal to m, i.e.

$$f(\partial_x) P_m(x) = \tilde{P}_{m'}(x), \tag{2.2.36}$$

where $m' < m$. Then since $\psi(x)$ has m vanishing moments the integrals (2.2.35) are zero.

2.2.3 Adaptive Calculations with the Non-Standard Form

In [48] it was shown that Calderón-Zygmund and pseudo-differential operators

can be applied to functions in $O(-N \log \epsilon)$ operations, where N is the dimension of the finest subspace $\mathbf{V}_0$ (that is assumed to be finite dimensional as in Section 2.2) and ϵ is the desired accuracy. In this Section we describe an algorithm for applying operators to functions with sub-linear complexity, e.g. $O(CN_s)$, where N_s is the number of significant coefficients in the wavelet representation of the function. Pseudo-code for the adaptive algorithm described in this Section can be found in Appendix C.

We are interested in applying operators to functions that are solutions of partial differential equations having regions of smooth, non-oscillatory behavior interrupted by a number of well-defined localized shocks or shock-like structures. The wavelet expansion of such functions, see e.g. (A.2.12), then consists of differences $\{d^j\}$ that are sparse and averages $\{s^j\}$ that may be dense. Adaptively applying the NS-form representation of an operator to a function expanded in a wavelet basis requires the rapid evaluation of

$$\hat{d}_k^j = \sum_l A_{k+l}^j d_{k+l}^j + \sum_l B_{k+l}^j s_{k+l}^j \tag{2.2.37}$$

$$\hat{s}_k^j = \sum_l \Gamma_{k+l}^j d_{k+l}^j \tag{2.2.38}$$

for $j = 1, 2, \ldots, J-1$ and $k \in \mathbb{F}_{2^{n-j}}$ and on the the final, coarse scale

$$\hat{d}_k^J = \sum_l A_{k+l}^J d_{k+l}^J + \sum_l B_{k+l}^J s_{k+l}^J \tag{2.2.39}$$

$$\hat{s}_k^J = \sum_l \Gamma_{k+l}^J d_{k+l}^J + \sum_l T_{k+l}^J s_{k+l}^J \tag{2.2.40}$$

for $k \in \mathbb{F}_{2^{n-J}}$ where n is the number of scales in the multiresolution analysis (see Figure A.6). The difficulty in adaptively applying the NS-form of an operator to such functions is the need to apply the B^j blocks of the operator to the averages $\{s^j\}$ in (2.2.37). Since the averages are "smoothed" versions of the function itself these vectors are not necessarily sparse and may consist of 2^{n-j}

significant coefficients on scale j. Our algorithm uses the fact (established in Section 2.2.2, see also [55]) that for differential operators, the Hilbert transform and the operator functions considered in Section 2.2.1, the rows of the B^j blocks have M vanishing moments. This means that when the row of a B^j block is applied to the "smooth" averages $\{s^j\}$, the resulting vector is sparse, as is illustrated in Figure 2.1.

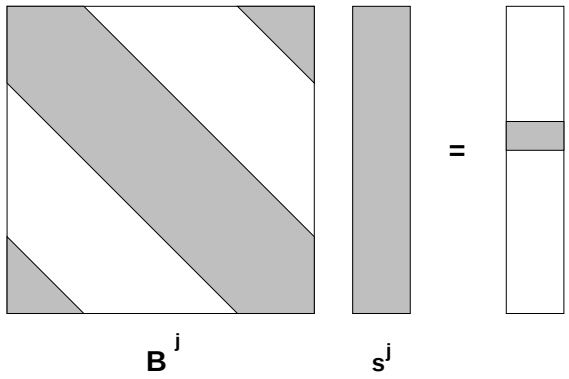

Figure 2.1. For the operators considered in Section 2.2.2 the vanishing-moment property of the rows of the B^j block yields a sparse result (up to a given accuracy ϵ) when applied to a smooth and dense vector $\{s^j\}$.

Using the vanishing-moment property and the indices of the significant wavelet coefficients $\{d^j\}$ we can identify coefficients $\{\tilde{s}^j\}$ that make significant contributions to (2.2.37). In this way we can replace the calculations with dense vectors (2.2.37) by calculations with sparse vectors

$$\tilde{d}_k^j = \sum_l A_{k+l}^j d_{k+l}^j + \sum_l B_{k+l}^j \tilde{s}_{k+l}^j \tag{2.2.41}$$

for $j = 1, 2, \ldots, J-1$ and $k \in \mathbb{F}_{2^{n-j}}$.

In what follows we first define a subspace of $\mathbf{V}_0$ in order to determine $\{\tilde{s}^j\}$. Then we describe a method for determining $\{\tilde{s}^j\}$ using the indices of the significant wavelet coefficients $\{d^j\}$.

Recall that the projection of f on $\mathbf{V}_0$ can be expressed as

$$(P_0 f)(x) = \sum_{j=1}^{J} \sum_{k \in \mathbb{F}_{2^{n-j}}} d_k^j \psi_{j,k}(x) + \sum_{k \in \mathbb{F}_{2^{n-J}}} s_k^J \varphi_{J,k}(x), \tag{2.2.42}$$

where $J \leq n$ and n is the number of scales in the multiresolution analysis. For the functions under consideration the magnitudes of many of the wavelet coefficients d_k^j in (2.2.42) are below a given threshold of accuracy ϵ. The representation of f on $\mathbf{V}_0$ using only coefficients above the threshold ϵ can be expressed by

$$(P_0 f)_\epsilon(x) = \sum_{j=1}^{J} \sum_{\{k:|d_k^j|>\epsilon\}} d_k^j \psi_{j,k}(x) + \sum_{k \in \mathbb{F}_{2^{n-J}}} s_k^J \varphi_{J,k}(x). \tag{2.2.43}$$

The error in using (2.2.43) to represent the projection of the function f instead of (2.2.42) is given by

$$||(P_0 f)_\epsilon(x) - (P_0 f)(x)||_2^2 = \sum_{j=1}^{J} \sum_{\{k:|d_k^j|<\epsilon\}} |d_k^j|^2 < \epsilon^2 N_r, \tag{2.2.44}$$

where N_r is the number of coefficients below the threshold. The number of significant wavelet coefficients may now be defined by $N_s = N - N_r$, where N is the dimension of the space $\mathbf{V}_0$.

We now define two subspaces of $\mathbf{V}_0$ that are used to identify the coefficients $\{\tilde{s}^j\}$ that contribute to the adaptive calculation of the sum (2.2.41). By definition the space $\mathbf{V}_0$ is spanned by the basis functions in the expansion (2.2.42). We define the **ϵ-accurate subspace for** f, denoted $\mathbf{D}_f^\epsilon \subset \mathbf{V}_0$, as the subspace spanned by the basis functions in the expansion (2.2.43), i.e.

$$\mathbf{D}_f^\epsilon = \mathbf{V}_J \bigcup \{\text{span } \{\psi_{j,k}(x)\} \ : \ |d_k^j| > \epsilon, 1 \leq j \leq J, k \in \mathbb{F}_{2^{n-j}}\}. \tag{2.2.45}$$

Associated with $\mathbf{D}_f^\epsilon$ are subspaces $\mathbf{S}_{f,j}^\epsilon$ for $j = 0, 1, \ldots, J-1$ that are formed by the span of the basis functions $\{\varphi_{j,2k+\lambda}(x)\}$. The set of basis functions

$\{\varphi_{j,2k+\lambda}(x)\}$ is determined depending on the presence of basis functions $\psi_{j+1,k}(x)$ in the space $\mathbf{D}_f^\epsilon$, i.e. for each $j = 0, 1, \ldots, J-1$

$$\mathbf{S}_{f,j}^\epsilon = \{\text{span } \{\varphi_{j,2k+\lambda}(x)\} \ : \ \psi_{j+1,k}(x) \in \mathbf{D}_f^\epsilon, k \in \mathbb{F}_{2^{n-(j+1)}}\}. \tag{2.2.46}$$

For $j = J$ we define the space

$$\mathbf{S}_{f,J}^\epsilon = \mathbf{V}_J. \tag{2.2.47}$$

In terms of the coefficients d_k^{j+1} the space $\mathbf{S}_{f,j}^\epsilon$ may be defined by

$$\mathbf{S}_{f,j}^\epsilon = \{\text{span } \{\varphi_{j,2k+\lambda}(x)\} \ : \ |d_k^{j+1}| > \epsilon, k \in \mathbb{F}_{2^{n-(j+1)}}\}. \tag{2.2.48}$$

In this way we can use $\mathbf{D}_f^\epsilon$ to 'mask' $\mathbf{V}_0$ forming $\mathbf{S}_{f,j}^\epsilon$. The coefficients $\{\tilde{s}^j\}$ that contribute to the sum (2.2.41) may now be identified as those coefficients corresponding to basis functions in $\mathbf{S}_{f,j}^\epsilon$.

We now show that significant wavelet coefficients $\{d^{j+1}\}$ and contributions of $\{B^j s^j\}$ to (2.2.37) both originate from the same coefficients $\{s^j\}$. In this way we can use the indices of $\{d^{j+1}\}$ to identify the coefficients $\{\tilde{s}^j\}$ that contribute to the sum (2.2.41). We begin by noting that the Taylor series with error term for $f(x + 2^j l)$ can be written

$$f(x + 2^j l) = \sum_{m=0}^{M-1} \frac{f^{(m)}(x)}{m!} 2^{jm} l^m + \frac{f^{(M)}(z)}{M!}(z - x)^M \tag{2.2.49}$$

where $z = z(x, j, l)$ lies between x and $x + 2^j l$. We begin by computing $\overline{d}_k^j = \sum_l \beta_{k+l}^j s_{k+l}^j$ using (2.2.49) and obtain

$$\begin{aligned} \overline{d}_k^j \ &= \ 2^{-j/2} \int_{-\infty}^{\infty} \varphi(2^{-j}x - k) \sum_{m=0}^{M-1} \frac{f^{(m)}(x)}{m!} (2^{jm}) \left(\sum_{l=-L}^{L} \beta_{k+l}^j l^m \right) dx + \\ &\quad \frac{2^{-j/2}}{M!} \sum_{l=-L}^{L} \beta_{k+l}^j \int_{-\infty}^{\infty} \varphi(2^{-j}x - k) f^{(M)}(z)(z - x)^M dx. \end{aligned}$$

By the vanishing-moment property, the first term is zero and we find after a change of variables

$$\overline{d}_k^j = \frac{2^{-j/2}}{M!} \sum_{l=-L}^{L} \beta_{k+l}^j \int_{-\infty}^{\infty} \varphi(x) f^{(M)}(z)(z - 2^j(x+k))^M dx, \tag{2.2.50}$$

for $k = 0, 1, 2, \ldots, 2^{J-j} - 1$.

To compute the differences $d_{k'}^{j+1} = \sum_l g_l s_{2k'+l}^j$ we use the averages

$$s_{2k'+l}^j = 2^{-j/2} \int_{-\infty}^{\infty} \varphi(2^{-j}x - 2k') f(x + 2^j l) dx. \tag{2.2.51}$$

Substituting the Taylor series (2.2.49) into (2.2.51) we obtain

$$\begin{aligned} d_{k'}^{j+1} &= 2^{-j/2} \int_{-\infty}^{\infty} \varphi(2^{-j}x - 2k') \sum_{m=0}^{M-1} \frac{f^{(m)}(x)}{m!} (2^{jm}) \left(\sum_l g_l l^m \right) dx + \\ &\quad \frac{2^{-j/2}}{M!} \sum_l g_l \int_{-\infty}^{\infty} \varphi(2^{-j}x - 2k') f^{(M)}(z)(z-x)^M dx. \end{aligned}$$

Using the vanishing-moment property of the $G = \{g_l\}$ filter we obtain

$$d_{k'}^{j+1} = \frac{2^{-j/2}}{M!} \sum_l g_l \int_{-\infty}^{\infty} \varphi(x) f^{(M)}(z)(z - 2^j(x + 2k'))^M dx, \tag{2.2.52}$$

for $k' = 0, 1, 2, \ldots, 2^{J-(j+1)} - 1$.

To show that $|d_{k'}^{j+1}| < \epsilon$ implies $|\overline{d}_k^j| < C\epsilon$ we consider two cases. If $|d_{k'}^{j+1}| < \epsilon$ and k is even, i.e. $k = 2n$ for $n = 0, 1, 2, \ldots, 2^{J-(j+1)} - 1$ then we see that $\overline{d}_{2n}^j$ and $d_{k'}^{j+1}$ (2.2.52) only differ in the coefficients g_l and β_{2n+l}^j. Since g_l and β_{2n+l}^j are of the same order then $|\overline{d}_{2n}^j| < C\epsilon$ for some constant C. In the case where $k = 2n + 1$ for $n = 0, 1, 2, \ldots, 2^{J-(j+1)} - 1$ we find

$$\overline{d}_{2n+1}^j = \frac{2^{-j/2}}{M!} \sum_{l=-L}^{L} \beta_{2n+1+l}^j \int_{-\infty}^{\infty} \varphi(x+1) f^{(M)}(z)(z - 2^j(x+2n))^M dx, \tag{2.2.53}$$

which again is of the same order as $d_{k'}^{j+1}$. Therefore, if $|d_{k'}^{j+1}| < \epsilon$ for $k' = 0, 1, 2, \ldots, 2^{J-(j+1)} - 1$, then for some constant C, $|\overline{d}_k^j| < C\epsilon$, for $k = 0, 1, 2, \ldots, 2^{J-j} - 1$.

2.3 Evaluating Functions in Wavelet Bases

In this Chapter we are concerned with the numerical approximation of solutions of partial differential equations of the form (2.0.1) via a scheme given, for example, by (2.2.3). Any numerical scheme of the form (2.2.3) uses the operations of applying an operator to a function and computing functions of the solution of the partial differential equation. In Section 2.2 we described an adaptive algorithm for applying operators to functions expanded in a wavelet basis. In this Section we describe an adaptive algorithm for evaluating the pointwise product of functions represented in wavelet bases. More generally, our results may be applied to computing functions $f(u)$ where f is an analytic function and u is expanded in a wavelet basis.

An approach for computing approximations to the solutions of nonlinear partial differential equations using Fourier methods is developed in [60], wherein linear terms are evaluated in the Fourier domain and nonlinear terms in the 'physical' domain. For example this approach applied to the Korteweg-de Vries equation

$$u_t + uu_x + \epsilon u_{xxx} = 0 \tag{2.3.1}$$

leads to a scheme of the form

$$u(x, t + \Delta t) - u(x, t - \Delta t) + \alpha \mathcal{F}^{-1}(\beta \mathcal{F} u(x, t)) - \gamma \mathcal{F}^{-1}(\delta \mathcal{F} u(x, t)) = 0, \tag{2.3.2}$$

where $\mathcal{F}(u) = \hat{u}(\xi)$ is the Fourier transform of $u(x)$ (computed via the FFT) and

$$\left.\begin{array}{rcl} \alpha & = & 2iu\Delta t \\ \beta & = & k \\ \gamma & = & 2i\Delta t \\ \delta & = & k^3 \end{array}\right\} \tag{2.3.3}$$

(see Equation (9), [60]). The usefulness of such an approach lies in the fact that linear terms are updated in the Fourier domain. On the other hand, the nonlinear contributions to the solution are computed in the x-domain. Since the linear and nonlinear contributions must be combined in either x-space or ξ-space, this scheme requires an extra FFT per time step. We observe that the linear and nonlinear terms in (2.3.2) are combined in 'physical' space. Approaches of this form have been referred to as 'pseudo-spectral' schemes, see e.g. [58, 59].

An alternative to the pseudo-spectral scheme is the 'product approximation' method used in combination with standard approximation methods, see e.g. [61, 62]. The product approximation method can be illustrated by considering the two point boundary value problem

$$\left.\begin{aligned} -u'' + f(u) &= 0 \quad 0 < x < 1 \\ u(0) &= u(1) = 0 \end{aligned}\right\} \tag{2.3.4}$$

Introducing a discretization $0 = x_0 < \ldots < x_N = 1$ we can write the standard approximation to $u(x)$ as

$$(\sum_i U_i \phi_i', \phi_j') + (f(\sum_i U_i \phi_i), \phi_j) = 0 \quad 1 \le j \le N \tag{2.3.5}$$

where $U_i = u(x_i)$, $\{\phi_i(x)\}_{i=1}^N$ is the basis for the space of continuous piecewise linear functions, and $(f, g) = \int_{-\infty}^{\infty} f(x)g(x)dx$ is the usual inner product. The product approximation of $u(x)$ is defined by the equations

$$\sum_i U_i(\phi_i', \phi_j') + \sum_i f(U_i)(\phi_i, \phi_j) = 0 \quad 1 \le j \le N \tag{2.3.6}$$

where u and $f(u)$ are approximated independently by different piecewise linear functions, [61]. The difference between approaches (2.3.5) and (2.3.6) is in the representation of the nonlinear term.

In order to compute the product approximation (2.3.6) one need only compute the inner products once and solve the resulting system of nonlinear equations. On the other hand the contribution of the nonlinear term in the standard approximation (2.3.5) must be computed at each step of the iteration. The results in [61, 62] indicate that the product approximation method admits higher local spatial accuracy than the standard approximation, e.g. $O(\Delta x^4)$ as compared to $O(\Delta x^2)$. The high order accuracy notwithstanding, there are two obvious drawbacks to schemes such as (2.3.5) and (2.3.6). The first is that at each iteration a number of computations proportional to the number of elements in the discretization must be employed in order to update the solution. In other words all coefficients U_i must be used in updating the solution via the numerical scheme. Therefore the complexity of the algorithm is at least proportional to the number of elements in the discretization since all coefficients U_i must be included in each computation. A second drawback to either the standard approximation or the product approximation is the lack of adaptivity. In [61] it is observed that the results of either approximation would be improved by introducing a finer discretization in the vicinity of sharp gradients in the solution.

There several difficulties encountered when one computes $f(u)$ when u is expanded in an arbitrary basis. If $u(x)$ is expanded in a basis

$$u(x) = \sum_{i=1}^{N} u_i b_i(x), \tag{2.3.7}$$

where u_i are the expansion coefficients and $b_i(x)$ are the basis functions, then in general

$$f(u(x)) \neq \sum_{i=1}^{N} f(u_i) b_i(x). \tag{2.3.8}$$

For example if

$$u(x) = \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} \hat{u}_j e^{i\pi jx}. \tag{2.3.9}$$

then the Fourier coefficients of the function $f(u)$ do not correspond to the function of the Fourier coefficients

$$f(u(x)) \neq \frac{1}{\sqrt{N}} \sum_{j=0}^{N-1} f(\hat{u}_j) e^{i\pi jx}. \tag{2.3.10}$$

In order to compute $f(u)$ in the wavelet system of coordinates we begin with the assumption that u and $f(u)$ are elements of $\mathbf{V}_0$, i.e. $u, f(u) \in \mathbf{V}_0$. We can then write

$$u(x) = \sum_k s_k^0 \varphi(x-k), \tag{2.3.11}$$

where s_k^0 are coefficients defined by

$$s_k^0 = \int_{-\infty}^{\infty} u(x)\varphi(x-k)dx. \tag{2.3.12}$$

Let us impose the additional assumption that the scaling function is interpolating so that

$$s_k^0 = u(k). \tag{2.3.13}$$

Since we have assumed that u and $f(u) \in \mathbf{V}_0$ we obtain

$$f(u) = \sum_k f(s_k^0)\varphi(x-k), \tag{2.3.14}$$

i.e. $f(u)$ is evaluated by computing the function of the expansion coefficients $f(s_k^0)$. Below we will describe how to relax the requirement that the scaling function be interpolating and still have property (2.3.14).

Typically $f(u)$ is not in the same subspace as u. In what follows we describe an adaptive algorithm for computing the pointwise square of a function, $f(u) = u^2$. In this algorithm we split $f(u)$ into projections on different

subspaces. Then we reconstruct 'pieces' of the wavelet expansion of u onto finer subspaces where we calculate contributions to $f(u)$ using an approximation to (2.3.14). This is in direct comparison with calculating $f(u)$ in a basis where the entire expansion must first be projected into a 'physical' space and then $f(u)$ is computed (see above). Finally, in Section 2.3.2 we discuss an algorithm for adaptively evaluating an arbitrary function $f(u)$.

2.3.1 Adaptive Calculation of u^2 Since the product of two functions can be expressed as a difference of squares, the algorithm for evaluating u^2 allows one to evaluate the pointwise product as well. The algorithm we describe is an improvement over the algorithm found in [47].

In order to compute u^2 in a wavelet basis we first recall that the projections of u on subspaces $\mathbf{V}_j$ and $\mathbf{W}_j$ are given by $P_j u \in \mathbf{V}_j$ and $Q_j u \in \mathbf{W}_j$ for $j = 0, 1, 2, \ldots, J \leq n$, respectively. Let j_f be the finest scale having significant wavelet coefficients that contribute to the ϵ-accurate approximation of u, i.e. the projection of u can be expressed as

$$(P_0 u)_\epsilon(x) = \sum_{j=j_f}^{J} \sum_{\{k:|d_k^j|>\epsilon\}} d_k^j \psi_{j,k}(x) + \sum_{k \in \mathbb{F}_{2^{n-J}}} s_k^J \varphi_{J,k}(x). \tag{2.3.15}$$

Let us first consider the case where u and $u^2 \in \mathbf{V}_0$ in which case we can expand $(P_0 u)^2$ in a 'telescopic' series

$$(P_0 u)^2 - (P_J u)^2 = \sum_{j=j_f}^{J} (P_{j-1} u)^2 - (P_j u)^2. \tag{2.3.16}$$

Decoupling scale interactions in (2.3.16) using $P_{j-1} = Q_j + P_j$ we arrive at

$$(P_0 u)^2 = (P_J u)^2 + \sum_{j=j_f}^{J} 2(P_j u)(Q_j u) + (Q_j u)^2. \tag{2.3.17}$$

Later, we will remove the condition that u and $u^2 \in \mathbf{V}_0$.

Remark: Equation (2.3.17) is written in terms of a finite number of scales. If however j ranges over $\mathbb{Z}$ then (2.3.17) can be written

$$u^2 = \sum_{j \in \mathbb{Z}} 2(P_j u)(Q_j u) + (Q_j u)^2, \tag{2.3.18}$$

which is essentially the paraproduct, see [50].

We observe that in (2.3.17) we need to compute products $(Q_j u)^2$ and $(P_j u)(Q_j u)$ where $Q_j u$ and $P_j u$ are elements of subspaces on the same scale, and thus have basis functions with the same size support. In addition we need to compute $(P_J u)^2$, however this computation involves only the coarsest scale and is not computationally expensive. The difficulty in evaluating (2.3.17), as compared with the Haar case (see [47]), is that the terms $(Q_j u)^2$ and $(P_j u)(Q_j u)$ may not necessarily belong to the same subspace as the multiplicands, e.g. $\mathbf{V}_j$. However, since

$$\mathbf{V}_j \bigoplus \mathbf{W}_j = \mathbf{V}_{j-1} \subset \mathbf{V}_{j-2} \subset \ldots \subset \mathbf{V}_{j-j_0} \subset \ldots, \tag{2.3.19}$$

we may think of both $P_j u \in \mathbf{V}_j$ and $Q_j u \in \mathbf{W}_j$ as elements of a finer subspace, that we denote $\mathbf{V}_{j-j_0}$, for some $j_0 \geq 1$. We can therefore compute the coefficients of $P_j u$ and $Q_j u$ in $\mathbf{V}_{j-j_0}$ using the reconstruction algorithm. Then on $\mathbf{V}_{j-j_0}$ we can calculate contributions to (2.3.17) using (2.3.14). The key observation is that in order to apply (2.3.17) we may always choose j_0 in such a way that, to within a given accuracy ϵ, $(Q_j u)^2$ and $(P_j u)(Q_j u)$ belong to $\mathbf{V}_{j-j_0}$.

In order to determine j_0 we begin by assuming $u \in \mathbf{V}_{j_0}$ and u approximates the solution of the partial differential equation (2.0.1). These assumptions imply that in the Fourier domain, the support of $\hat{\varphi}(\xi)$ overlaps the support of $\hat{u}(\xi)$. We begin by showing that, for scaling functions with a sufficient number

of vanishing moments, the coefficients s_l^0 and the values $u(x_l)$, for some x_l, may be made to be within ϵ of each other.

The coefficients of the expansion of $u \in \mathbf{V}_{j_0}$ are given by

$$s_l^{j_0} = 2^{-j_0/2} \int_{-\infty}^{\infty} u(x)\varphi(2^{-j_0}x - l)dx, \tag{2.3.20}$$

which in terms of $\hat{u}(\xi)$, (see e.g. (2.2.9)), can be written

$$s_l^{j_0} = 2^{-j_0/2} \int_{-\infty}^{\infty} \hat{u}(2^{-j_0}\xi)\overline{\hat{\varphi}(\xi)}e^{-i\xi l}d\xi. \tag{2.3.21}$$

The integral in (2.3.21) can be written

$$s_l^{j_0} = 2^{-j_0/2} \sum_{k\in\mathbb{Z}} \int_{-\pi}^{\pi} \hat{u}(2^{-j_0}(\xi + 2\pi k))\overline{\hat{\varphi}(\xi + 2\pi k)}e^{-i\xi l}d\xi. \tag{2.3.22}$$

Since we have assumed that u satisfies (2.0.1), we have that the support of $\hat{u}$ is compact and centered about the $k = 0$ term in (2.3.22). Therefore the infinite series (2.3.22) consists of one term

$$s_l^{j_0} = 2^{-j_0/2} \int_{-\pi}^{\pi} \hat{u}(2^{-j_0}\xi)\overline{\hat{\varphi}(\xi)}e^{-i\xi l}d\xi. \tag{2.3.23}$$

In order to evaluate (2.3.23) we recall that $\varphi(x)$ has M shifted vanishing moments, i.e. $\int_{-\infty}^{\infty}(x-\alpha)^m\varphi(x)dx = 0$, where $\alpha = \int_{-\infty}^{\infty} x\varphi(x)dx$. We can then write

$$\int_{-\infty}^{\infty}(x-\alpha)^m\varphi(x)dx = \frac{1}{(-i)^m}\frac{\partial^m}{\partial\xi^m}e^{i\alpha\xi}\int_{-\infty}^{\infty}\varphi(x)e^{-i\xi x}dx\Big|_{\xi=0} = 0 \tag{2.3.24}$$

for $m = 1, 2, \ldots, M$ and arrive at

$$\frac{1}{(-i)^m}\frac{\partial^m}{\partial\xi^m}\overline{\hat{\varphi}(\xi)}e^{i\alpha\xi}\Big|_{\xi=0} = 0, \tag{2.3.25}$$

and

$$\overline{\hat{\varphi}(\xi)}e^{-i\alpha\xi}\Big|_{\xi=0} = 1. \tag{2.3.26}$$

Expanding $\overline{\hat{\varphi}(\xi)}e^{i\alpha\xi}$ in a Taylor series near $\xi = 0$ we arrive at

$$\overline{\hat{\varphi}(\xi)}e^{i\alpha\xi} = 1 + \frac{\xi^{M+1}}{(M+1)!}\frac{\partial^{M+1}}{\partial\xi^{M+1}}\,\overline{\hat{\varphi}(\xi)}e^{i\alpha\xi}\Big|_{\xi=z} \tag{2.3.27}$$

where $z = z(\xi)$ lies between ξ and zero. Substituting (2.3.27) in (2.3.23) yields

$$s_l^{j_0} = 2^{-j_0/2}\int_{-\pi}^{\pi}\hat{u}(2^{-j_0}\xi)e^{-i\xi(l+\alpha)}d\xi + E_{M,j_0} \tag{2.3.28}$$

where

$$E_{M,j_0} = \frac{2^{-j_0/2}}{(M+1)!}\int_{-\pi}^{\pi}\hat{u}(2^{-j_0}\xi)e^{-i\xi(l+\alpha)}\;\xi^{M+1}\frac{\partial^{M+1}}{\partial\xi^{M+1}}\left(\overline{\hat{\varphi}(\xi)}e^{i\alpha\xi}\right)\Big|_{\xi=z}d\xi, \tag{2.3.29}$$

is the error introduced by the Taylor expansion (2.3.27).

To describe the algorithm for computing the pointwise product, let us denote by $\mathcal{R}_{j_0}^{j}(\cdot)$ the operator to reconstruct (represent) a vector on subspace $\mathbf{V}_j$ or $\mathbf{W}_j$ in the subspace $\mathbf{V}_{j-j_0}$. On $\mathbf{V}_{j-j_0}$ we can then use the coefficients $\mathcal{R}_{j_0}^{j}(P_j u)$ and $\mathcal{R}_{j_0}^{j}(Q_j u)$ to calculate contributions to the product (2.3.17) using ordinary multiplication (as in (2.3.14)). To this end we compute the projections

$$(P_{j-j_0}u)^2 = 2(\mathcal{R}_{j_0}^{j}(P_j u))(\mathcal{R}_{j_0}^{j}(Q_j u)) + (\mathcal{R}_{j_0}^{j}(Q_j u))^2, \tag{2.3.30}$$

for $j = j_f, j_f+1, \ldots, J-1$, and on scale J we compute

$$(P_{J-j_0}u)^2 = (\mathcal{R}_{j_0}^{j}(P_J u))^2 + 2(\mathcal{R}_{j_0}^{j}(P_J u))(\mathcal{R}_{j_0}^{j}(Q_J u)) + (\mathcal{R}_{j_0}^{j}(Q_J u))^2. \tag{2.3.31}$$

As we have shown, $P_{j-j_0}(P_{j-j_0}u)^2 = (P_{j-j_0}u)^2$ to within a given ϵ.

We now proceed to decompose the projection on $\mathbf{V}_{j-j_0}$, for $j = j_f, \ldots J$ into the wavelet basis. This procedure is completely equivalent ot the decomposition one has to perform after applying the NS-form. The algorithm for computing the projection of u^2 in a wavelet basis is illustrated in Figure 2.3.1.

To demonstrate that the algorithm is adaptive, we recall that u has a sparse representation in the wavelet basis. Thus, evaluating $(Q_j u)^2$ for $j = 1, 2, \ldots, J$ requires manipulating only sparse vectors. Evaluating the square of the final coarse scale averages $(P_J u)^2$ is inexpensive. The difficulty in evaluating (2.3.30) lies in evaluating the products $\mathcal{R}^j_{j_0}(P_j u)(\mathcal{R}^j_{j_0} Q_j u)$ since the vectors $P_j u$ are typically dense. The adaptivity of the algorithm comes from an observation that in the products appearing in (2.3.30) we may use the coefficients $Q_j u$ as a 'mask' of the $P_j u$ (this is similar to the algorithm for adaptively applying operators to functions). In this way contributions to (2.3.30) are calculated based on the presence of significant wavelet coefficients $Q_j u$ and therefore significant products $\mathcal{R}^j_{j_0}(P_j u)(\mathcal{R}^j_{j_0} Q_j u)$. The complexity of our algorithm is therefore automatically adaptable to the complexity of the wavelet representation of u.

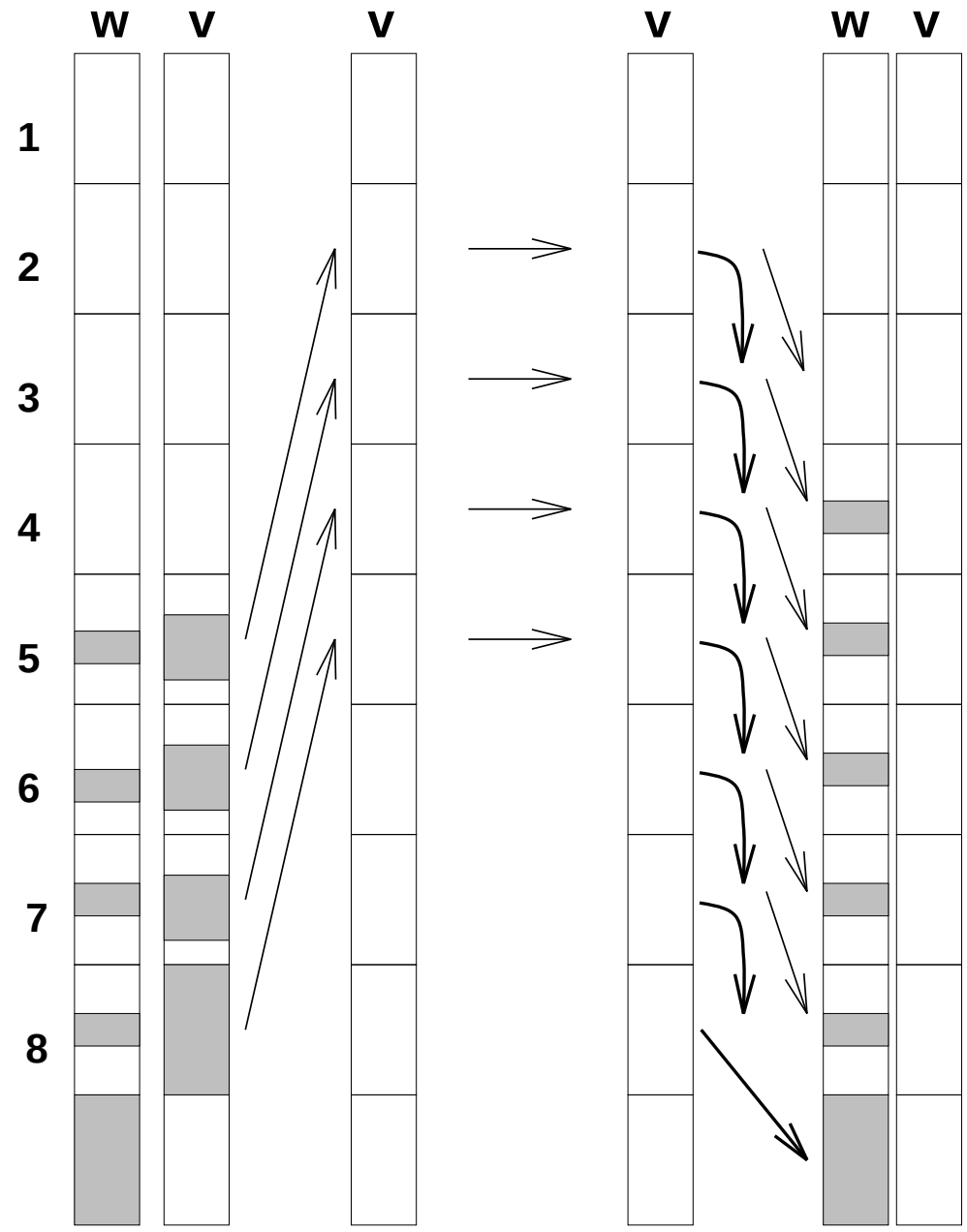

Figure 2.2. Adaptively computing the square of a function in the wavelet basis.

2.3.2 Notes on the Adaptive Calculation of General $f(u)$

This Section consists of a number of observations regarding the evaluation of functions other than $f(u) = u^2$ in wavelet bases. For analytic $f(u)$ we can apply the same approach as in Section 2.3.1 wherein we assume $f(P_0u) \in \mathbf{V}_0$ and expand the projection $f(P_0u)$ in the 'telescopic' series

$$f(P_0u) - f(P_Ju) = \sum_{j=1}^{J} f(P_{j-1}u) - f(P_ju). \tag{2.3.32}$$

Using $P_{j-1} = Q_j + P_j$ to decouple scale interactions in (2.3.32) and assuming $f(\cdot)$ to be analytic we substitute the Taylor series

$$f(Q_ju + P_ju) = \sum_{n=0}^{N} \frac{f^{(n)}(P_ju)}{n!}(Q_ju)^n + E_{j,N}(f,u) \tag{2.3.33}$$

to arrive at

$$f(P_0u) = f(P_Ju) + \sum_{j=1}^{J}\sum_{n=1}^{N} \frac{f^{(n)}(P_ju)}{n!}(Q_ju)^n + E_{j,N}(f,u). \tag{2.3.34}$$

We note that for $f(u) = u^2$ (2.3.34) is equivalent to (2.3.17).

This approach can be used for functions $f(u)$ that have rapidly converging Taylor series expansions, e.g. $f(u) = \sin(u)$, for $|u|$ sufficiently small. In this case, for a given accuracy ϵ we fix an N so that $|E_{j,N}(f,u)| < \epsilon$. We note that the partial differential equation (2.0.1) typically involves functions $f(\cdot)$ that are not only analytic but in many cases polynomial in u and of degree p. If this is the case then for each fixed j the series in (2.3.33) is of degree p and $E_{j,N}(f,u) = 0$ for $N > p$. In any event we are led to the need to evaluate the sum

$$\sum_{j=1}^{J}\sum_{n=1}^{N} \frac{f^{(n)}(P_ju)}{n!}(Q_ju)^n, \tag{2.3.35}$$

appearing in (2.3.34) that can be done by using the algorithm described in Section 2.3.1.

The approach described above can be used for functions $f(u)$ that have rapidly converging Taylor series expansions. If on the other hand the Taylor series expansion of $f(u)$ does not converge rapidly, e.g. $f(u) = e^u$, we are led to consider the following alternatives. Let us consider the specific case where $f(u) = e^u$ and examine two approaches to evaluating $f(u)$. In the first approach we begin, as above, by expanding e^u in the 'telescopic' series

$$e^{P_0 u} - e^{P_J u} = \sum_{j=1}^{J} e^{P_{j-1} u} - e^{P_j u}, \tag{2.3.36}$$

and using $P_{j-1} = Q_j + P_j$ to decouple scale interactions. We then arrive at

$$e^{P_0 u} = e^{P_J u} + \sum_{n=1}^{J} e^{P_j u}(e^{Q_j u} - 1). \tag{2.3.37}$$

Since the wavelet coefficients $Q_j u$ are sparse, the multiplicand $e^{Q_j u} - 1$ is significant only where $Q_j u$ is significant. Therefore we can evaluate (2.3.37) using the masking algorithm described in Section 2.3.1 where in this case the mask is determined by significant values of $e^{Q_j u} - 1$. We note that the terms in (2.3.37) involve exponentials of scalars, i.e. $P_j u$ and $Q_j u$, and can be computed in a straightforward manner using the exponential function.

The difficulty in using such an approach depends on the relative size, or dynamic range, of the variable u. If, for example, $u(x) = \alpha \sin(2\pi x)$ on $0 \leq x \leq 1$ then $e^{-\alpha} \leq u(x) \leq e^{\alpha}$. Even for relatively moderate values of α the function e^u may range over several orders of magnitude. Figure 2.3 illustrates this behavior for $\alpha = 5$.

A solution to calculating e^u taking into consideration the possible dynamic range of e^u is to apply a scaling and squaring approach. Instead of computing e^u directly one calculates $e^{2^{-j}u}$ and repeatedly squares the result via $(e^{2^{-j}u})^{2^j}$. The constant j depends on the magnitude of u and is chosen so that

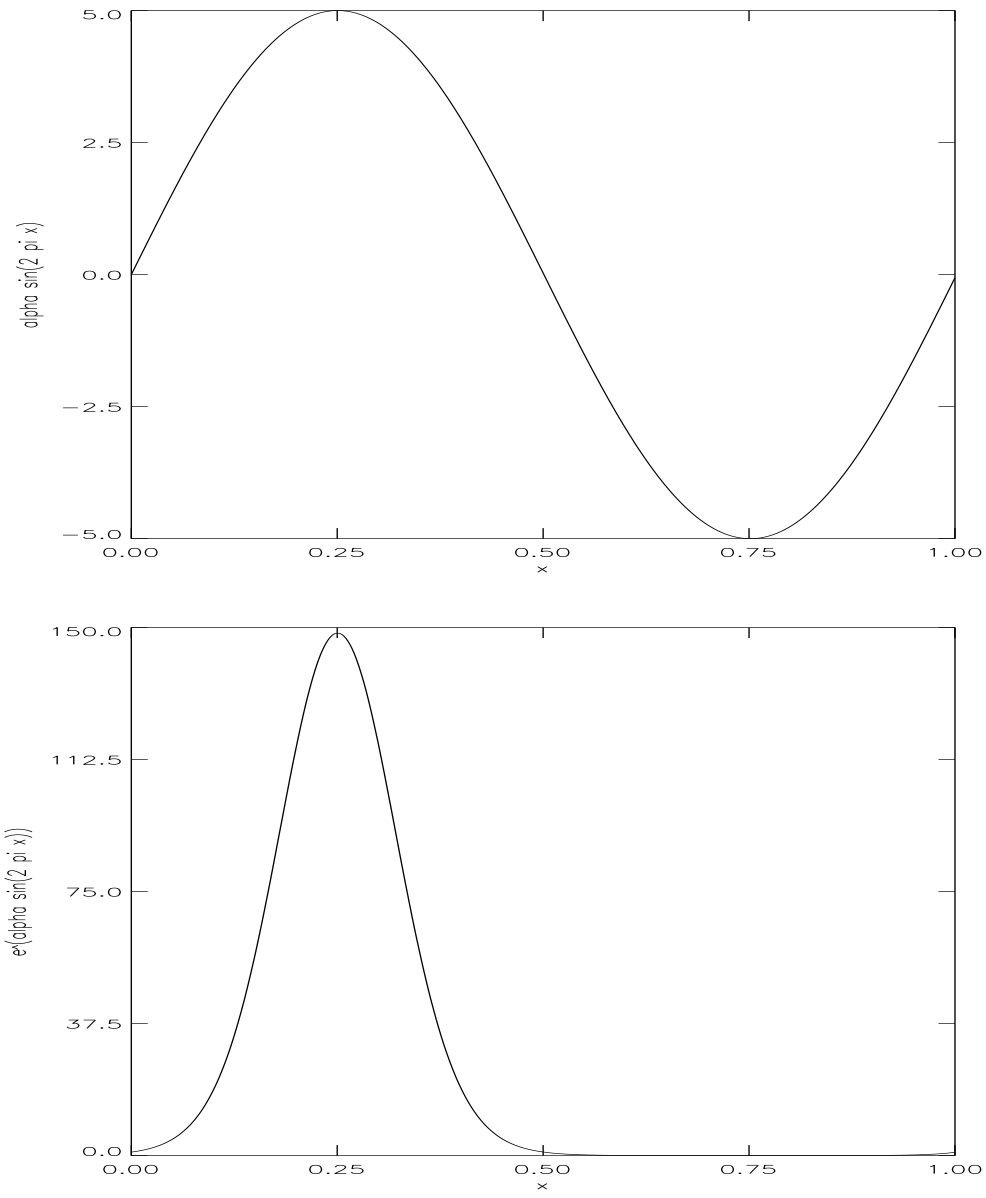

Figure 2.3. Illustration of the large dynamic range in $f(u) = e^{\alpha \sin(2\pi x)}$ over $0 \leq x \leq 1$ for $\alpha = 5$. Even for the relatively moderate α, $f(u) = e^u$ ranges between $0.006 < e^u < 150$.

the variable u is scaled as, for example, $-1 \leq 2^{-j}u \leq 1$. In this interval, calculating $e^{2^{-j}u}$ can be accomplished as described by equation (2.3.37) and the masking algorithm of Section 2.3.1. Then one repeatedly applies the algorithm for the pointwise square to $e^{2^{-j}u}$ to arrive at the wavelet expansion of e^u.

2.4 Results of Numerical Experiments

In this Section we present the results of a number of numerical experiments. In each example we approximate solutions of an initial value problem of the form

$$u_t = \mathcal{L}u + \mathcal{N}f(u) \tag{2.4.1}$$

$$u(x,0) = u_0(x) \qquad 0 \le x \le 1 \tag{2.4.2}$$

with periodic boundary conditions

$$u(0,t) = u(1,t) \qquad 0 \le t \le T. \tag{2.4.3}$$

Each numerical experiment consists of the following steps. First one chooses a discretization of the integral in the solution of (2.4.1)

$$u(x,t) = e^{(t-t_0)\mathcal{L}}u_0(x) + \int_{t_0}^{t} e^{(t-\tau)\mathcal{L}}\mathcal{N}f(u(x,\tau))d\tau, \tag{2.4.4}$$

as discussed in Section 2.1. In each experiment described below we will identify the discretization of (2.4.4). The wavelet representation of the operators appearing in the approximation to (2.4.4) are computed via the methods outlined in Section 2.2.1.

We then fix the wavelet basis and the parameters of the experiment. This includes choosing the number of scales in the multiresolution analysis, n (which defines $\Delta x = 2^{-n}$), the depth of the decomposition, $J \le n$, and the accuracy with which the numerical experiment is performed, ϵ. We have chosen the wavelet basis having a scaling function with shifted vanishing moments, the so-called 'Coiflets', in order to demonstrate the algorithm for evaluating $f(u)$ developed in Section 2.3. We choose the number of vanishing moments M and the quadrature mirror filters $\{h_k\}$ and $\{g_k\}$.

We project the initial condition (2.4.2) on $\mathbf{V}_0$. This amounts to evaluating the coefficients

$$s_l^0 = \int_{\infty}^{\infty} u_0(x)\varphi(x-l)dx. \tag{2.4.5}$$

For sufficiently smooth initial conditions we can bypass directly evaluating the integral (2.4.5). Instead we use the shifted vanishing moments of the scaling function $\varphi(\cdot)$ to compute the coefficients s_l^0 to within ϵ via

$$s_l^0 \approx u(l-\alpha), \tag{2.4.6}$$

(see the discussion in Section 2.3.1). We note that in this case the discretization of the initial condition is similar to traditional discretizations where one sets

$$U(x_i, t_0) = u_0(i\Delta x), \tag{2.4.7}$$

for $i = 0, 1, 2, \ldots, 2^n - 1$. We use the initial condition (2.4.6) to start the iteration and evaluate solutions of (2.4.4) at successive time steps.

Since approximations to the integral in (2.4.4) are implicit in time (see (2.1.18)) we must solve an equation of the form

$$U(t_{j+1}) = E(U(t_j)) + I(U(t_j), U(t_{j+1})), \tag{2.4.8}$$

at each time step for $U(t_{j+1})$. Note that we have dropped the explicit x dependence on $U(t_j)$. In (2.4.8) E is the explicit part of the approximation to (2.4.4) and I is the implicit part that depends on the approximation of the integral. One can use specialized techniques for solving (2.4.8), e.g. using preconditioners to accelerate the convergence of the iteration. However, in our experiments we use a straightforward fixed point method to compute $U(t_{j+1})$. We begin by setting

$$U_0(t_{j+1}) = E(U(t_j)) + I(U(t_j), U(t_j)), \tag{2.4.9}$$

and repeatedly evaluate

$$U_{k+1}(t_{j+1}) = E(U(t_j)) + I(U(t_j), U_k(t_{j+1})), \tag{2.4.10}$$

for $k = 0, 1, 2 \ldots$. We terminate the iteration when

$$\|U_{k+1}(t_{j+1}) - U_k(t_{j+1})\| < C\epsilon, \tag{2.4.11}$$

for some constant C (in our experiments we choose $C = 1$), and where

$$\|U_{k+1}(t_{j+1}) - U_k(t_{j+1})\|^2 = 2^{-n} \sum_{i=1}^{2^n} (U_{k+1}(x_i, t_{j+1}) - U_k(x_i, t_{j+1}))^2. \tag{2.4.12}$$

When (2.4.11) is satisfied we set

$$U(t_{j+1}) = U_{k+1}(t_{j+1}). \tag{2.4.13}$$

Again we note that one can use a more sophisticated iterative scheme and a different stopping condition for evaluating (2.4.8) (e.g. simply compute (2.4.10) for a fixed number of iterations).

2.4.1 The Heat Equation We begin with this simple linear example in order to illustrate several results and provide a bridge to the nonlinear problems discussed below. In this Section we are concerned with the calculation of numerical solutions of the heat equation on the unit interval

$$u_t = \nu u_{xx} \qquad 0 \leq x \leq 1 \quad 0 \leq t \leq 1, \tag{2.4.14}$$

for $\nu > 0$, subject to the initial condition

$$u(x, 0) = u_0(x) \quad 0 \leq x \leq 1 \tag{2.4.15}$$

and the periodic boundary condition

$$u(0, t) = u(1, t) \quad 0 \leq t \leq 1 \tag{2.4.16}$$

There are several well-known approaches for solving (2.4.14) with (2.4.15) and (2.4.16) and more general equations of this type having variable coefficients. Equation (2.4.14) can be viewed as a simple representative of this class of equations and although it is used in this example we emphasize that the following remarks are applicable to the case where $\nu = \nu(x)$.

It is well known that such equations can be solved using finite differences. However, explicit finite difference schemes are conditionally stable and the discretization must satisfy the stability condition $\nu\Delta t/\Delta x^2 < 1$, [28, 29]. This condition tends to require prohibitively small time steps. An alternate finite difference approach is, for example, the implicit Crank-Nicolson scheme, [28, 29], which is unconditionally stable and accurate to $O(\Delta t^2 + \Delta x^2)$. At each time step, the Crank-Nicolson scheme requires solving a system of equations

$$AU(t_{j+1}) = BU(t_j) \tag{2.4.17}$$

for $j = 0, 1, 2, \ldots, M-1$, where we have suppressed the dependence of $U(x,t)$ on x. The matrices A and B are given by

$$\begin{aligned} A &= \text{diag}(-\tfrac{\alpha}{2}, 1+\alpha, -\tfrac{\alpha}{2}) \\ B &= \text{diag}(\tfrac{\alpha}{2}, 1-\alpha, \tfrac{\alpha}{2}), \end{aligned} \tag{2.4.18}$$

where $\alpha = \nu\frac{\Delta t}{\Delta x^2}$.

The wavelet based solution of (2.4.14) is formulated as follows. Following the discussion in Section 2.1, we can write the solution of (2.4.14) with (2.4.15) and (2.4.16) as

$$u(x,t) = e^{t\mathcal{L}}u_0(x), \tag{2.4.19}$$

where $\mathcal{L} = \nu\partial_{xx}$. We compute (2.4.19) by discretizing the time interval $[0,1]$ into M subintervals of length $\Delta t = 1/M$ and repeatedly applying the NS-form

representation of the operator $e^{\Delta t \mathcal{L}}$ via

$$U(t_{j+1}) = e^{\Delta t \mathcal{L}} U_(t_j), \tag{2.4.20}$$

for $j = 0, 1, 2, \ldots, M-1$ using the projection of the initial condition,

$$U(0) = u_0(i\Delta x). \tag{2.4.21}$$

In this example our goal is to use the wavelet representation of the operators appearing in the Crank-Nicolson scheme (2.4.17) to identify a source of error in low order schemes. In the wavelet domain we compare the Crank-Nicolson scheme (which we have taken as an example of a low order scheme) with our 'exact', high order, unconditionally stable and explicit method (2.4.20). The fact that the Crank-Nicolson scheme is unconditionally stable allows one to choose Δt independently of Δx; in particular one can choose Δt to be proportional to Δx. In order to emphasize our point we 'mis-use' the Crank-Nicolson scheme in that we set $\Delta x = \Delta t$ and $\nu = 1$. On one hand, since Crank-Nicolson is second order accurate in both time and space, such choices of the parameters Δx, Δt, and ν appear to be reasonable. However, by analyzing the scheme in the Fourier domain we find that high frequency components in an initial condition decay very slowly.

For example, let us consider the following initial condition

$$u_0(x) = \begin{cases} x & 0 \le x \le \frac{1}{2} \\ 1 - x & \frac{1}{2} \le x \le 1 \end{cases} \tag{2.4.22}$$

that has a discontinuous derivative at $x = \frac{1}{2}$. Figure 2.4 illustrates the evolution of the initial condition (2.4.22) via (2.4.17) with $\Delta t = \Delta x$ and $\nu = 1$. The slow decay of high frequency components in the initial condition is clearly illustrated in Figure 2.4. We have implemented equation (2.4.20) and display the result in

Figure 2.5 for the case where $\nu = 1.0$, $\Delta t = \Delta x = 2^{-n}$ and $n = 9$. We note that there is a proper decay of the sharp peak in the initial condition.

The slow decay of high frequency components in the initial condition via the Crank-Nicolson scheme can be explained as follows. The matrices A and B can be diagonalized by the Fourier transform so that the Crank-Nicolson scheme (2.4.17) becomes

$$U(t_{j+1}) = \Lambda U(t_j), \tag{2.4.23}$$

where Λ is the (diagonal) matrix of eigenvalues of $A^{-1}B$. The eigenvalues of A and B are given by

$$\begin{aligned} \lambda_{A,l} &= 1 + 2\alpha \sin^2(\tfrac{l\pi}{2N}) \\ \lambda_{B,l} &= 1 - 2\alpha \sin^2(\tfrac{l\pi}{2N}) \end{aligned} \tag{2.4.24}$$

and, thus, the elements of Λ are given by

$$\lambda_l = \frac{1 - 2\alpha \sin^2(\frac{l\pi}{2N})}{1 + 2\alpha \sin^2(\frac{l\pi}{2N})} \tag{2.4.25}$$

for $l = 1, 2, \ldots, N$. We note that $|\lambda_l| \leq 1$. Let us consider the highest frequency eigenvalue that corresponds to $l = N$

$$\lambda_N = \frac{1 - 2\alpha}{1 + 2\alpha}. \tag{2.4.26}$$

To clarify our explanation let us choose $\nu = 1$ and $\Delta t = \Delta x$ so that $\alpha = \frac{1}{\Delta x}$. Figure 2.6 illustrates the behavior of λ_N as a function of α. We see that as α becomes large, or equivalently Δx becomes small, the eigenvalue λ_N tends to -1. Eigenvalues with l close to N are the cause of the slow decay of the high frequency components in the initial condition. We note that there are various ad hoc remedies (e.g. smoothing) used in conjunction with the Crank-Nicolson scheme to remove these slowly decaying high frequency components.

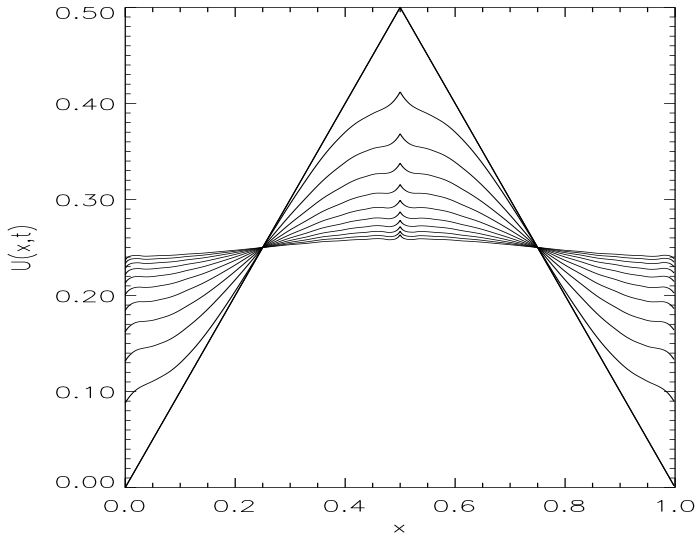

Figure 2.4. Solution of the heat equation using the Crank-Nicolson method (2.4.17) with $\Delta t = \Delta x = 2^{-9}$ and $\nu = 1.0$. Note the slowly decaying peak in the solution that is due to the eigenvalue $\lambda_N = -0.99902344$.

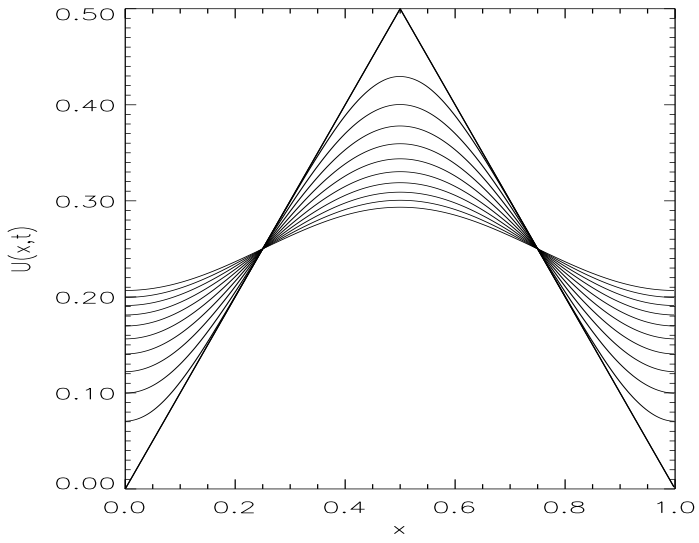

Figure 2.5. Solution of the heat equation using the NS-form of the exponential with $\Delta t = \Delta x = 2^{-9}$ and $\nu = 1.0$, i.e. equation (2.4.20).

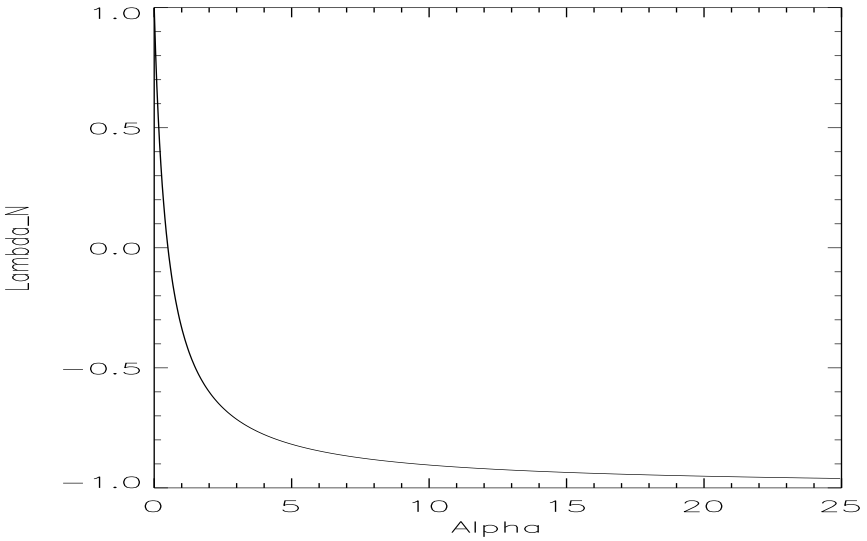

Figure 2.6: Maximum eigenvalue of the matrix $A^{-1}B$ as a function of $\alpha = \frac{1}{\Delta x}$.

Let us explain the difference between the results of our wavelet based approach and those of the Crank-Nicolson scheme in the wavelet basis. We may consider the Crank-Nicolson scheme in the following explicit form

$$U(t_{j+1}) = A^{-1}BU(t_j), \tag{2.4.27}$$

and construct the NS-form representation of the operator $A^{-1}B$ and compare it with that of $e^{\Delta t\mathcal{L}}$. The NS-form representation of an operator explicitly separates elements of the operator that act on the high frequency components of u into the finer scale blocks. These finer scale or high frequency blocks are located in the upper left corner of the NS-form. Therefore, the blocks of the NS-form of the operator $A^{-1}B$ (used in (2.4.27)) that are responsible for the high frequency components in the solution are the significant entries in the upper left portion of Figure 2.7. One can compare Figure 2.7 with Figure 2.8 which illustrates the NS-form representation of the exponential operator used in (2.4.20).

We note that although the Crank-Nicolson scheme is not typically used for this regime of parameters (i.e. $\nu = 1$ and $\Delta t = \Delta x$), a similar phenomena will be observed for any low order method, namely for a given cutoff the NS-form representation of the matrix for the low order scheme will have more entries than that of the exponential operator in the wavelet basis.

Let us conclude this example by reiterating that the wavelet based scheme via (2.4.19) is explicit, and unconditionally stable, since we are computing the exponential of a negative definite operator. The accuracy in the spatial variable of our scheme is $O(h^{2M})$ where M is the number of vanishing moments, $h = \Delta x = 2^{-n}$ and n is the number of scales in the multiresolution analysis. Additionally, our scheme is spectrally accurate in time. Our scheme is adaptive simply by virtue of using a sparse data structure to represent the operator

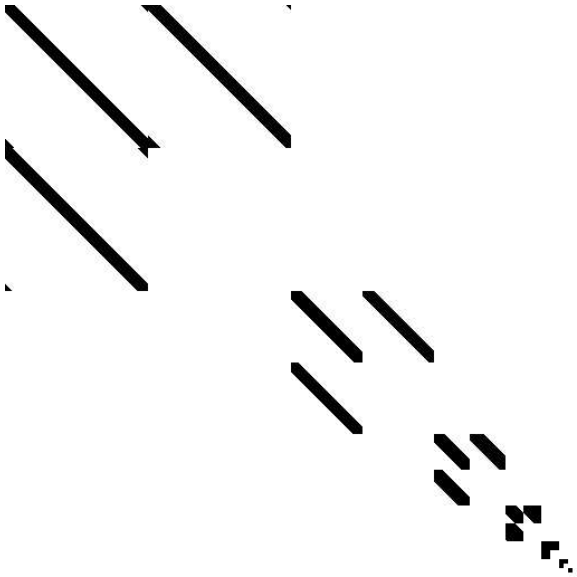

Figure 2.7. NS-form representation of the operator $A^{-1}B$ used in the Crank-Nicolson scheme (2.4.17). Entries of absolute value greater than 10^{-8} are shown in black. The wavelet basis is Daubechies with $M = 6$ vanishing moments ($L_f = 18$), the number of scales is $n = 9$ and $J = 7$. We have set $\nu = 1.0$ and $\Delta t = \Delta x = 2^{-9}$. Note that the top left portion of the Figure contains non-zero entries which indicate high frequency components present in the operator $A^{-1}B$.

Figure 2.8. NS-form representation of the operator $e^{\nu \Delta t \mathcal{L}}$ used in (2.4.20). Entries of absolute value greater than 10^{-8} are shown in black. The wavelet basis is Daubechies with $M = 6$ vanishing moments ($L_f = 18$), the number of scales is $n = 9$ and $J = 7$. We have set $\nu = 1.0$ and $\Delta t = \Delta x = 2^{-9}$.

$e^{\nu \Delta t \partial_{xx}}$ and the sparsity of the solution in the wavelet basis.

Referring to Figures 2.7 and 2.8 it is clear that the NS-form of the operator $e^{\Delta t \mathcal{L}}$ in our high order scheme is sparser than the NS-form for the operator $A^{-1}B$ in the second order Crank-Nicolson scheme. In order to quantify 'sparser' we consider the compression ratio of a matrix defined by

$$c = \frac{N^2}{N_s} \tag{2.4.28}$$

where $N = 2^n$ (the dimension of the finest subspace $\mathbf{V}_0$) and N_s is the number of significant entries present in the matrix. The compression ratio for the NS-form of the operator $A^{-1}B$ for the Crank-Nicolson scheme shown in Figure 2.7 is $c = 10.7$. The compression ratio for the NS-form of the exponential operator for our scheme shown in Figure 2.8 is $c = 35$.

Finally we note that if we were to consider (2.4.14) with variable coefficients, e.g.

$$u_t = a(x)u_{xx}, \tag{2.4.29}$$

the exponential operator $e^{\Delta t a(x) \mathcal{L}}$ can be computed in $O(N)$ operations using the scaling and squaring method outlined in [49].

2.4.2 Burgers' Equation Our next example is the numerical calculation of solutions of Burgers' equation on the unit interval

$$u_t = \nu u_{xx} - u u_x \qquad 0 \leq x \leq 1 \quad t \geq 0, \tag{2.4.30}$$

for $\nu > 0$, together with the initial condition

$$u(x, 0) = u_0(x) = \sin(2\pi x) \quad 0 \leq x \leq 1 \tag{2.4.31}$$

and periodic boundary conditions

$$u(0, t) = u(1, t) = 0 \quad t \geq 0. \tag{2.4.32}$$

Burgers' equation is the prototypical example of a nonlinear partial differential equation incorporating both linear diffusion and nonlinear advection. Solutions of Burgers' equation consist of stationary or moving shocks and capturing such behavior is an important simple test of a new numerical method. Additionally, this is the reason why a number of wavelet based schemes have been applied to this equation, see e.g. [63, 64].

Burgers' equation may be solved analytically by the Cole-Hopf transformation [33, 34] wherein it is observed that a solution of (2.4.30) may be expressed as

$$u(x,t) = -2\nu\frac{\phi_x}{\phi}, \tag{2.4.33}$$

where $\phi = \phi(x,t)$ is a solution of the heat equation with initial condition

$$\phi(x,0) = e^{-\frac{1}{4\nu\pi}\int u(x,0)dx}. \tag{2.4.34}$$

Remark: We note that if ν is small enough, e.g. $\nu = 10^{-3}$, then using (2.4.33) as the starting point for a numerical method turns out to be a poor approach. This is due to the large dynamic range of the transformed initial condition (2.4.34) (approximately 70 orders of magnitude for the initial condition (2.4.31)). Consequently the finite arithmetic involved in a numerical scheme leads to a loss of accuracy in calculating $u(x,t)$ via (2.4.33), notably within the vicinity of the shock.

Our numerical scheme for computing approximations to the solution of (2.4.30) is formulated as follows. Following the discussion in Section 2.1 we may write an $O(\Delta t^2)$ approximation to the solution of (2.4.30) as

$$U(t_{i+1}) = E(U(t_i)) + I(U(t_i), U(t_{j+1})), \tag{2.4.35}$$

where

$$E(U(t_i)) = e^{\Delta t \mathcal{L}} U(t_i) \tag{2.4.36}$$

$$I(U(t_i), U(t_{j+1})) = \tfrac{1}{2}\mathcal{O}_{\mathcal{L},1}\left(U(t_i)\partial_x U(t_{i+1}) + \partial_x U(t_i) U(t_{i+1})\right), \tag{2.4.37}$$

where $\mathcal{L} = \nu \partial_{xx}$, and

$$\mathcal{O}_{\mathcal{L},1} = \left(e^{\Delta t \mathcal{L}} - \mathbf{I}\right) \mathcal{L}^{-1}. \tag{2.4.38}$$

Since (2.4.35) is implicit in $U(t_{i+1})$ we are led to consider the iteration

$$U_{k+1}(t_{i+1}) = E(U(t_i)) + I(U(t_i), U_k(t_{j+1})), \tag{2.4.39}$$

for $k = 0, 1, 2, \ldots$ where we use

$$U_0(t_{i+1}) = e^{\Delta t \mathcal{L}} U(t_i) + \mathcal{O}_{\mathcal{L},1}\left(U(t_i)\partial_x U(t_i)\right) \tag{2.4.40}$$

as the initial guess. Note that we have suppressed the explicit x dependence in these expressions. The stopping criteria for the iteration (2.4.39) is that the standard deviation between two successive iterates be within ϵ of each other, i.e.

$$\|U_{k+1}(t_{i+1}) - U_k(t_{i+1})\| < \epsilon, \tag{2.4.41}$$

where the left hand side is given by (2.4.12). When (2.4.41) is satisfied the solution at t_{i+1} is set to the final iterate,

$$U(t_{i+1}) = U_{k+1}(t_{i+1}). \tag{2.4.42}$$

We note that since the solution is expressed as the sum (2.4.35), and $E(U(t_i))$ is equivalent to the operator used in the solution of the heat equation, the linear diffusion in (2.4.30) is accounted for in an essentially exact way. Thus we may attribute numerical artifacts in the solution to the nonlinear advection term in (2.4.30).

Evaluating the iteration (2.4.39) is done as follows. We first compute the linear contribution (2.4.36) using the adaptive multiplication algorithm described in Section 2.2.3. We then compute each summand in the right hand side of (2.4.37)

$$U(t_i)\partial_x U_k(t_{i+1}) + \partial_x U(t_i) U_k(t_{i+1}), \tag{2.4.43}$$

using the pointwise product algorithm described in Section 2.3 and multiply this result by the NS-form representation of $\mathcal{O}_{\mathcal{L},1}$ again using the adaptive multiplication algorithm described in Section 2.2.3. We note that the number of operations needed to perform each of these operations is proportional to the number of significant wavelet coefficients present in the representation of the appropriate functions and operators.

In order to illustrate the applicability of our approach we may vary a number of parameters. In order to demonstrate the algorithm for evaluating $f(u)$ developed in Section 2.3 We have chosen the wavelet basis having a scaling function with shifted vanishing moments, the so-called 'Coiflets'. In our experiments we fix the number of vanishing moments to $M = 6$ which is equivalent to choosing $L_f = 18$. The choice of basis and number of vanishing moments fixes the quadrature mirror filters $\{h_k\}$ and $\{g_k\}$. We have also chosen $\epsilon = 10^{-6}$ so that only coefficients of absolute value greater than ϵ are used in our calculations of the solution $u(x,t)$. For a given N the value of J is chosen so that L_f is less that the dimension of either subspace $\mathbf{V}_J$ or $\mathbf{W}_J$, i.e. $L_f < 2^{N-J}$. In this case the filter does not overlap itself in the decomposition/reconstruction algorithms. The choices of Δt and ν are made in such a way that we may illustrate the flexibility of our algorithm to variations in these variables.

For each of the following examples, we illustrate the accuracy of our

approach by comparing the approximate solution u_w with the exact solution u_e using

$$\|u_w - u_e\|^2 = 2^{-n} \sum_{i=0}^{2^n-1} (u_w(x_i,t) - u_e(x_i,t))^2. \tag{2.4.44}$$

The approximation u_w is computed via (2.4.39) and the exact solution u_e is computed via, see e.g. [31],

$$u(x,t) = \frac{\int_{-\infty}^{\infty} \frac{x-\eta}{t} e^{-G(\eta;x,t)/2\nu} d\eta}{\int_{-\infty}^{\infty} e^{-G(\eta;x,t)/2\nu} d\eta} \tag{2.4.45}$$

where

$$G(\eta;x,t) = \int_0^{\eta} F(\eta')d\eta' + \frac{(x-\eta)^2}{2t} \tag{2.4.46}$$

and $F(\eta) = u_0(\eta)$ is the initial condition (2.4.31). The initial conditions have been chosen so that (2.4.46) may be evaluated analytically and we compute the integrals in (2.4.45) using a high order method.

Example 1. In this example $n = 10$, $J = 4$, $\Delta t = 0.001$, $\nu = 0.1$. We refer to Figures 2.9-2.12. The large value of ν controls the evolution of the solution and does not permit a sharp shock to form. Figure 2.9 illustrates the projection of the solution on $\mathbf{V}_0$ computed using $\epsilon = 10^{-6}$. Figure 2.10 illustrates the error (2.4.44) per time step in the wavelet solution u_w as compared with the exact solution u_e. The number of operations per time step used to update the solution is proportional to the number of significant coefficients in the wavelet representation of the solution. Figure 2.11 illustrates the number of significant coefficients per time step needed to represent the solution in the wavelet basis. Figure 2.12 illustrates the number of iterations per time step required to satisfy the stopping criterion (2.4.41). We note that the compression ratio (2.4.28) for the NS-form representation of the first derivative, exponential and nonlinear operators are 14.2, 17.3, and 26.3, respectively.

Example 2. In this example $n = 10$, $J = 4$, $\Delta t = 0.001$, and $\nu = 0.01$. We refer to Figures 2.13-2.16. Figure 2.13 illustrates the projection of the solution on $\mathbf{V}_0$ computed using $\epsilon = 10^{-6}$. Figure 2.14 illustrates the error (2.4.44) per time step in the wavelet solution u_w as compared with the exact solution u_e. The number of operations per time step used to update the solution is proportional to the number of significant coefficients in the wavelet representation of the solution. Figure 2.15 illustrates the number significant coefficients per time step needed to represent the solution in the wavelet basis. Figure 2.16 illustrates the number of iterations per time step required to satisfy the stopping criterion (2.4.41). We note that the compression ratio (2.4.28) for the NS-form representation of the first derivative, exponential and nonlinear operators are 14.2, 15.4, and 21.3, respectively.

Example 3. In this example $n = 10$, $J = 4$, $\Delta t = 0.001$, $\nu = 0.001$. We refer to Figures 2.17-2.19. Decreasing the viscosity by a factor of 10 as compared with ν in Example 2 causes the shock to appear more quickly and to be steeper than in the previous examples. Moreover using $n = 10$ scales to represent the solution in the wavelet basis is insufficient to represent the high frequency components present in the solution. Figure 2.17 illustrates the projection of the solution on $\mathbf{V}_0$ beyond the point in time where the solution is well represented by $n = 10$ scales. The data in this Figure was computed using $\epsilon = 10^{-6}$. We see that high frequency oscillations have appeared in the projection which may be viewed as a local analogue of the well-know Gibbs phenomenon. Figure 2.18 illustrates the number of significant coefficients per time step needed to represent the solution for various values of ϵ. Figure 2.19 illustrates the number of iterations per time step required to satisfy the stopping criterion (2.4.41). We note that the

compression ratio (2.4.28) for the NS-form representation of the first derivative, exponential and nonlinear operators are 14.2, 15.4 and 21.3, respectively. In Example 4 we resolve the shock by introducing more scales.

Example 4. In order to resolve the shock and represent high frequency components present in the solution, as illustrated by Example 3, we must introduce more scales. In this example we set $n = 15$ and $J = 9$ and leave $\Delta t = 0.001$ and $\nu = 0.001$. The subspace $\mathbf{V}_0$ may now be viewed as a discretization of the unit interval into 2^{15} grid points with the step size $\Delta x = 2^{-15}$. We refer to Figures 2.20-2.23. Figure 2.20 illustrates the projection of the solution on $\mathbf{V}_0$ computed using $\epsilon = 10^{-6}$. Figure 2.21 illustrates the error (2.4.44) per time step in the wavelet solution u_w as compared with the exact solution u_e. Figure 2.22 illustrates the number of significant coefficients per time step needed to represent the solution $u(x,t)$ in the wavelet basis. Again we note that the number of operations needed to update the solution is proportional to the number of significant coefficients. Figure 2.23 illustrates the number of iterations per time step required to satisfy the stopping criterion (2.4.41). We note that the error increases as the shock forms and then decreases as the viscosity causes the solution to decay and become more smooth. We note that the compression ratio (2.4.28) for the NS-form representation of the first derivative, exponential and nonlinear operators are 442.2, 3708.5 and 1364.9, respectively.

Example 5. In this final example we compute the solution to Burgers' equation for the initial condition

$$u(x,t) = \sin(2\pi x) + \frac{1}{2}\sin(4\pi x). \tag{2.4.47}$$

This initial condition leads to the formation of left and right moving shocks.

In this example $n = 12$, $\nu = 0.001$, $\Delta t = 0.001$, $J = 6$, and $\epsilon = 10^{-6}$. We refer to Figures 2.24-2.27. Figure 2.24 illustrates the projection of the solution on $\mathbf{V}_0$. Figure 2.25 illustrates the error (2.4.44) per time step in the wavelet solution u_w as compared with the exact solution u_e. The number of operations per time step used to update the solution is proportional to the number of significant coefficients in the wavelet representation of the solution. Figure 2.26 illustrates the number of significant coefficients per time step needed to represent the solution in the wavelet basis. Figure 2.27 illustrates the number of iterations per time step required to satisfy the stopping criterion (2.4.41). We note that the compression ratio (2.4.28) for the NS-form representation of the first derivative, exponential and nonlinear operators are 442.2, 3708.5 and 1364.9, respectively.

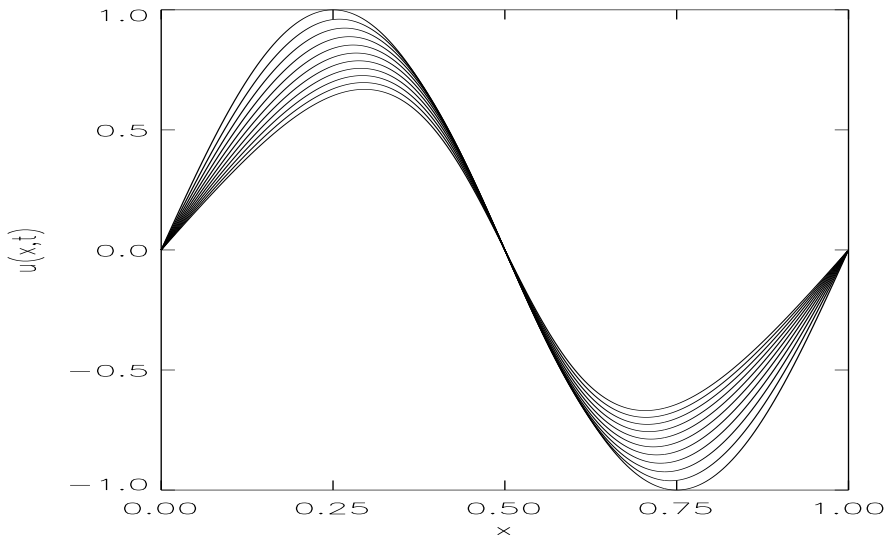

Figure 2.9. The projection on $\mathbf{V}_0$ of the solution of Burgers' equation at various time steps computed via the iteration (2.4.39). In this experiment $N = 2^{10}$, $\nu = 0.1$, $\Delta t = 0.001$, $J = 4$, and $\epsilon = 10^{-6}$. This Figure corresponds to Example 1 of the text and Figures 2.10, 2.11, and 2.12 below.

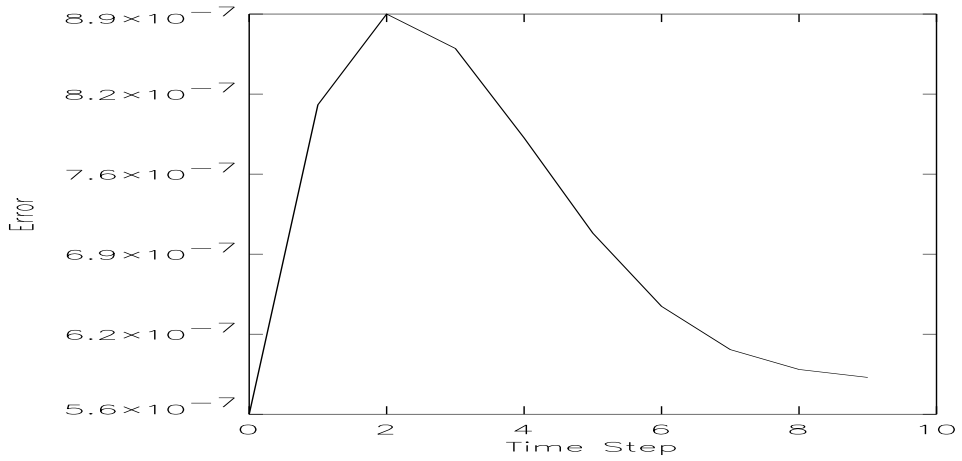

Figure 2.10. The error (2.4.44) per time step in the approximation (2.4.39) as compared with the exact solution (2.4.45). This Figure corresponds to Example 1 of the text and Figure 2.9 above.

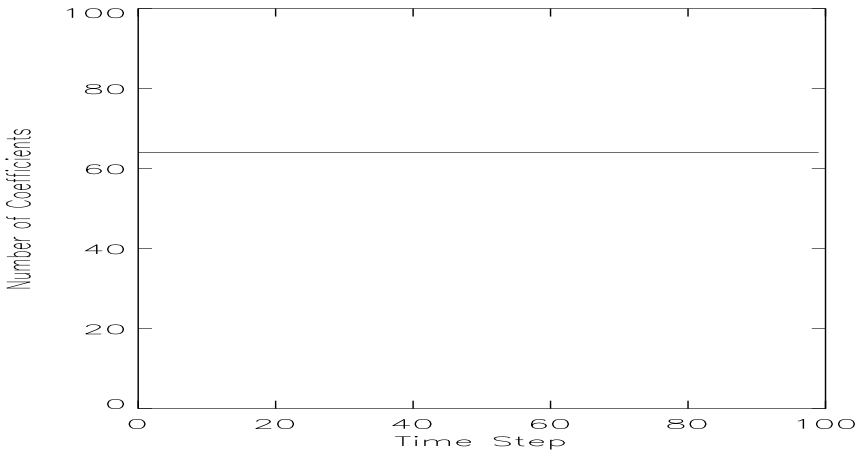

Figure 2.11. The total number of significant wavelet coefficients per time step. We note that the solution is sufficiently smooth that the number of significant coefficients is the same for $\epsilon = 10^{-4}, 10^{-5}$, and 10^{-4}. This Figure corresponds to Example 1 of the text and Figure 2.9 above.

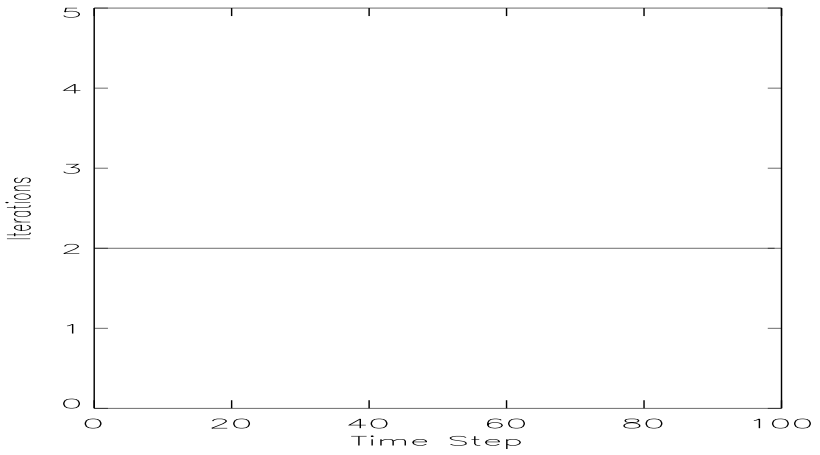

Figure 2.12. The number of iterations per time step needed to satisfy the stopping criterion (2.4.11). This Figure corresponds to Example 1 of the text and Figure 2.9 above.

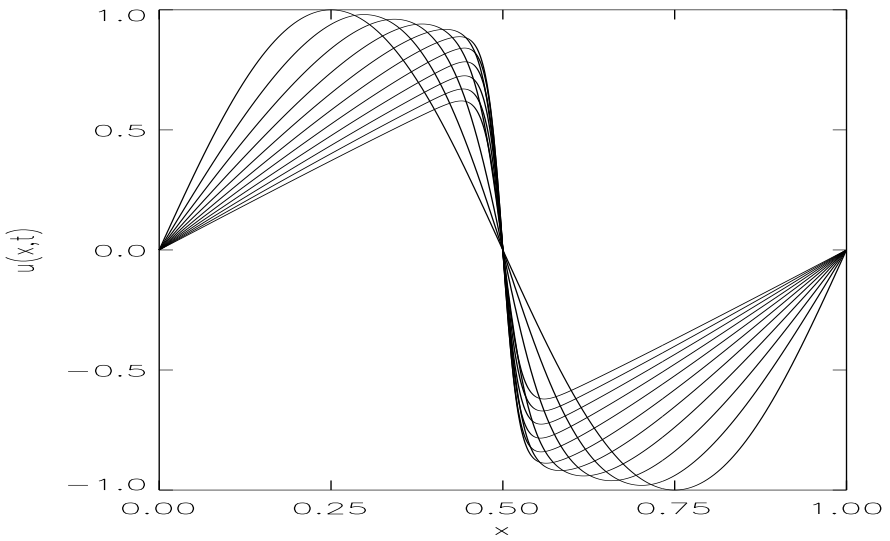

Figure 2.13. The projection on $\mathbf{V}_0$ of the solution of Burgers' equation at various time steps computed via the iteration (2.4.39). In this experiment $N = 2^{10}$, $\nu = 0.01$, $\Delta t = 0.001$, $J = 4$, and $\epsilon = 10^{-6}$. This Figure corresponds to Example 2 of the text and Figures 2.14, 2.15, and 2.16 below.

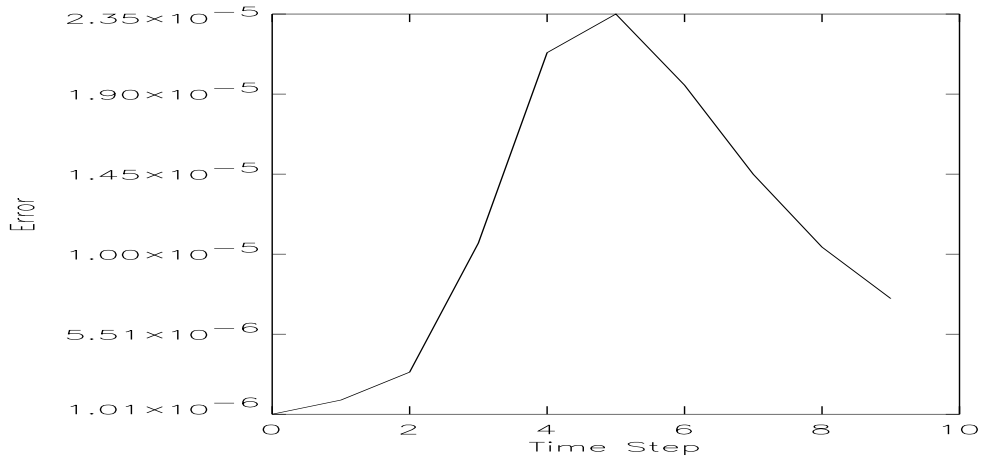

Figure 2.14. The error (2.4.44) per time step in the approximation (2.4.39) as compared with the exact solution (2.4.45). This Figure corresponds to Example 2 of the text and Figure 2.13 above.

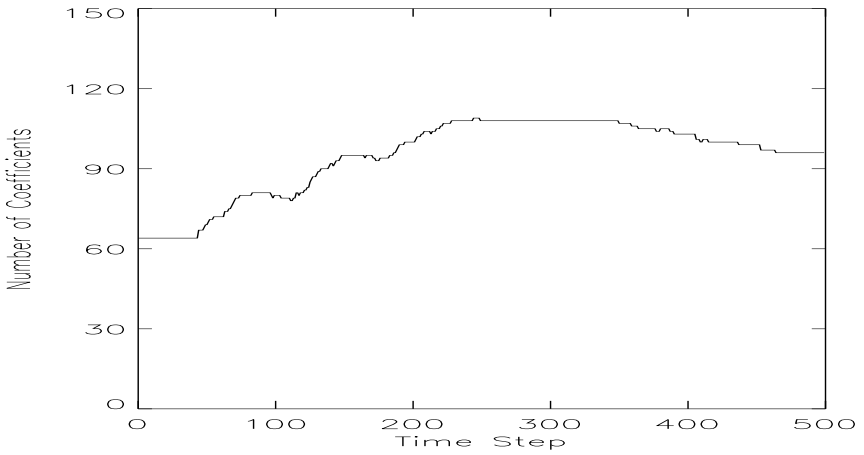

Figure 2.15. The total number of significant wavelet coefficients per time step. This Figure corresponds to Example 2 of the text and Figure 2.13 above.

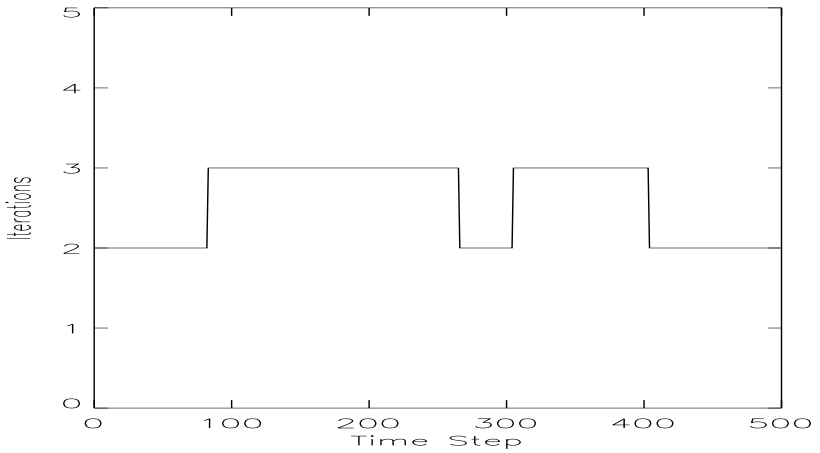

Figure 2.16. The number of iterations per time step needed to satisfy the stopping criterion (2.4.11). This Figure corresponds to Example 2 of the text and Figure 2.13 above.

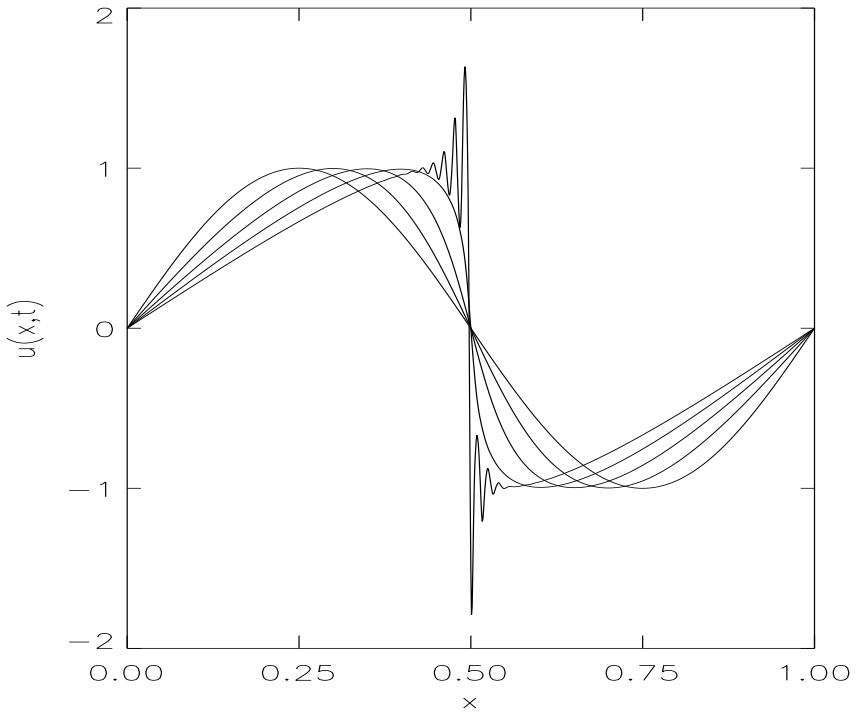

Figure 2.17. The projection on $\mathbf{V}_0$ of the solution of Burgers' equation at various time steps computed via the iteration (2.4.39). In this experiment $N = 2^{10}$, $\nu = 0.001$, $\Delta t = 0.001$, $J = 4$, and $\epsilon = 10^{-6}$. An analogue of the Gibbs phenomenon begins because the shock cannot be accurately represented by $n = 10$ scales. We note that the scheme remains stable in spite of the oscillations. This Figure corresponds to Example 3 of the text and Figures 2.18 and 2.19 below. The shock is resolved in Example 4, below, by introducing more scales.

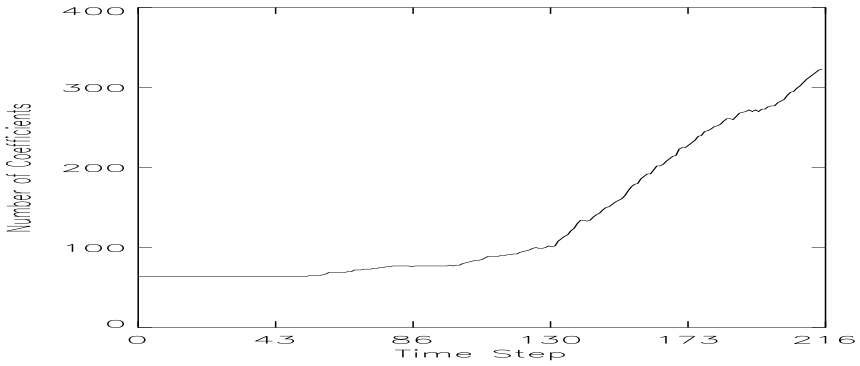

Figure 2.18. The total number of significant wavelet coefficients per time step. This Figure corresponds to Example 3 of the text and Figure 2.17 above.

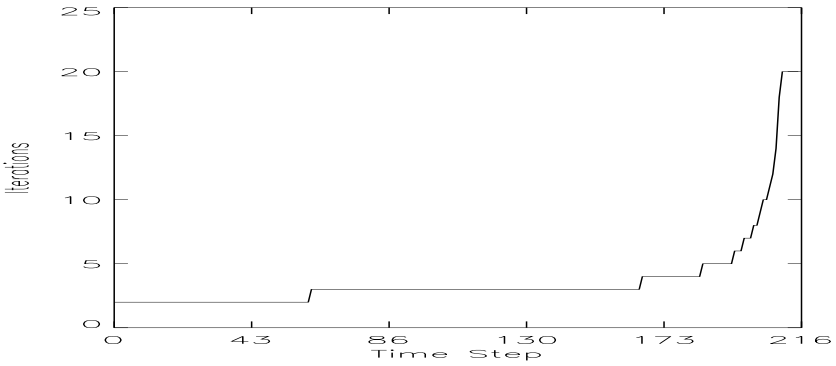

Figure 2.19. The number of iterations per time step needed to satisfy the stopping criterion (2.4.11). We note that as high frequency components appear in the solution the number of iterations rapidly increases. This Figure corresponds to Example 3 of the text and Figure 2.17 above.

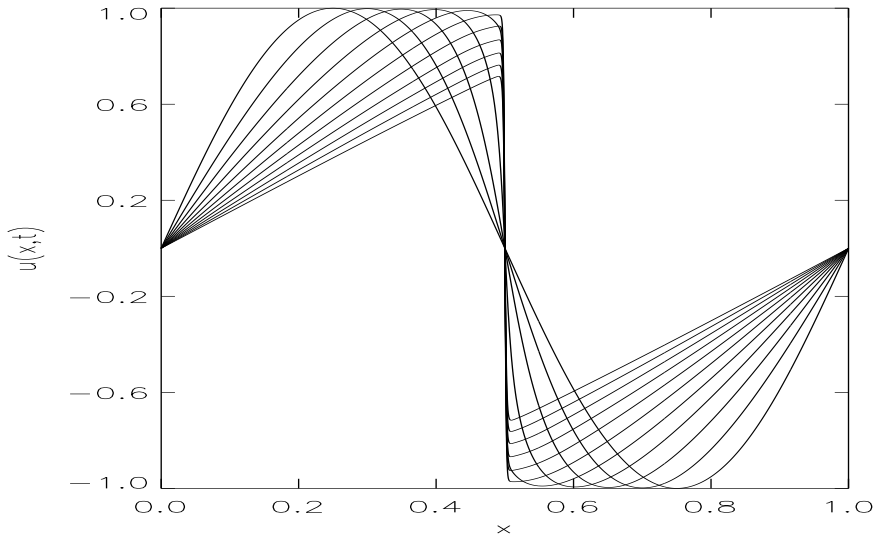

Figure 2.20. Resolution of the shock developed in Example 3. The projection on $\mathbf{V}_0$ of the solution of Burgers' equation at various time steps computed via the iteration (2.4.39). In this experiment $N = 2^{15}$, $\nu = 0.001$, $\Delta t = 0.001$, $J = 9$, and $\epsilon = 10^{-6}$. We note that increasing the number of scales resolves the shock. This Figure corresponds to Example 4 of the text and Figures 2.21, 2.22, and 2.23 below.

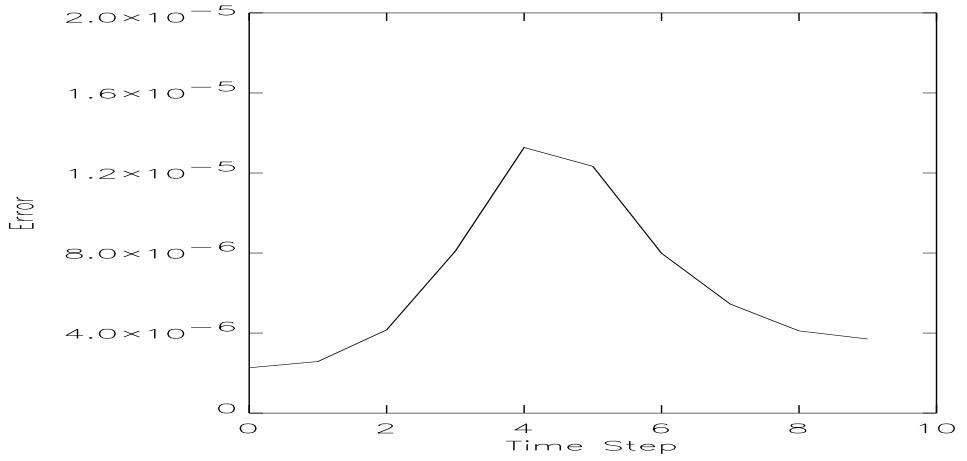

Figure 2.21. The error (2.4.44) per time step in the approximation (2.4.39) as compared with the exact solution (2.4.45). This Figure corresponds to Example 4 of the text and Figure 2.20 above.

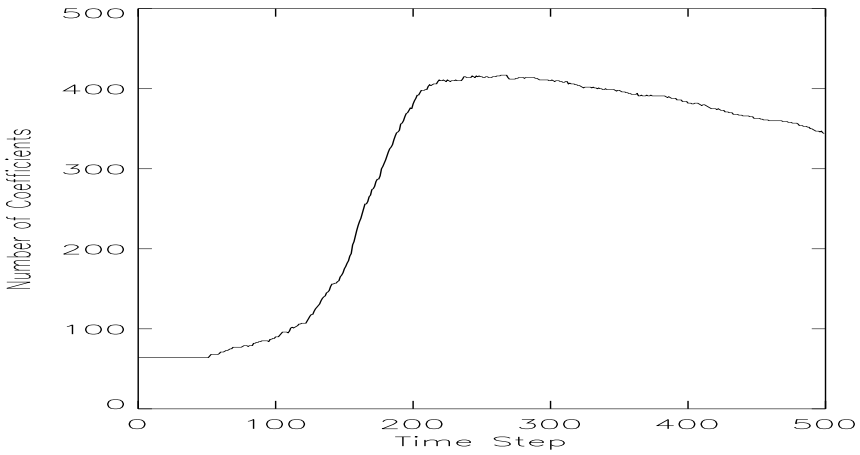

Figure 2.22. The total number of significant wavelet coefficients per time step. This Figure corresponds to Example 4 of the text and Figure 2.20 above.

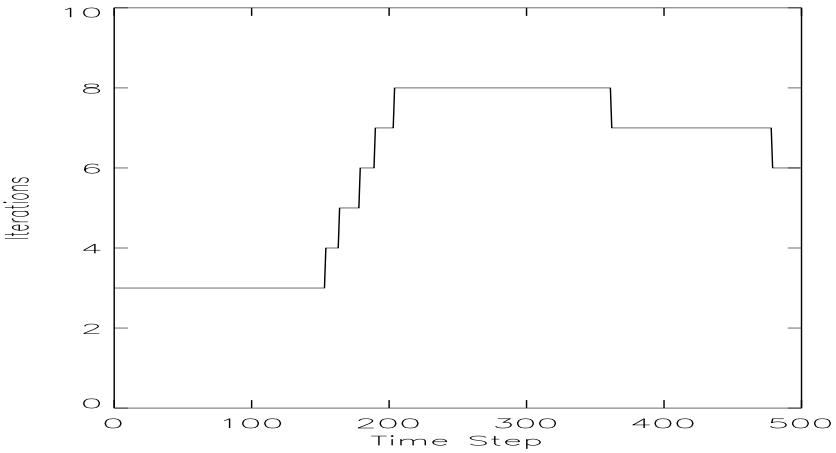

Figure 2.23. The number of iterations per time step needed to satisfy the stopping criterion (2.4.11). This Figure corresponds to Example 1 of the text and Figure 2.9 above.

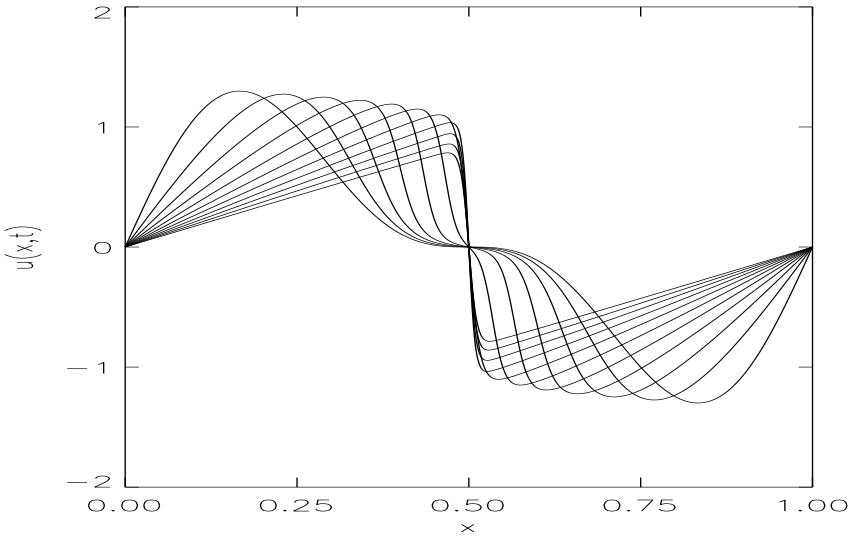

Figure 2.24. The projection on $\mathbf{V}_0$ of the solution of Burgers' equation at various time steps computed via the iteration (2.4.39). In this experiment $N = 2^{12}$, $\nu = 0.005$, $\Delta t = 0.001$, $J = 6$, $\epsilon = 10^{-6}$, and the initial condition is given by $u(x,t) = \sin(2\pi x) + \frac{1}{2}\sin(4\pi x)$. This Figure corresponds to Example 5 of the text and Figures 2.25, 2.26, and 2.27 below.

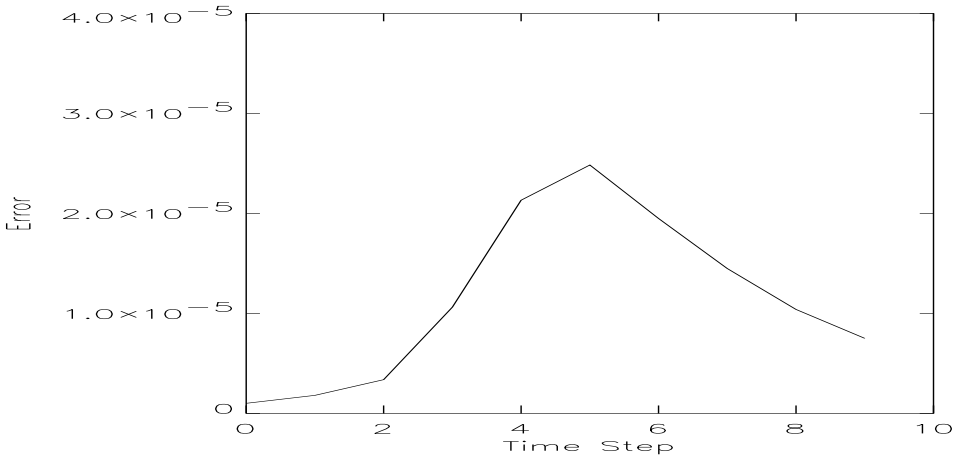

Figure 2.25. The error (2.4.44) per time step in the approximation (2.4.39) as compared with the exact solution (2.4.45). This Figure corresponds to Example 5 of the text and Figure 2.24 above.

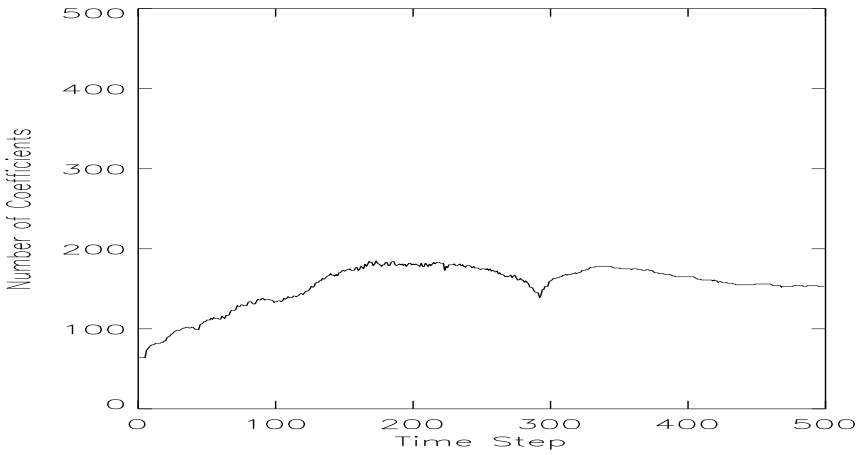

Figure 2.26. The total number of significant wavelet coefficients per time step. This Figure corresponds to Example 5 of the text and Figure 2.24 above.

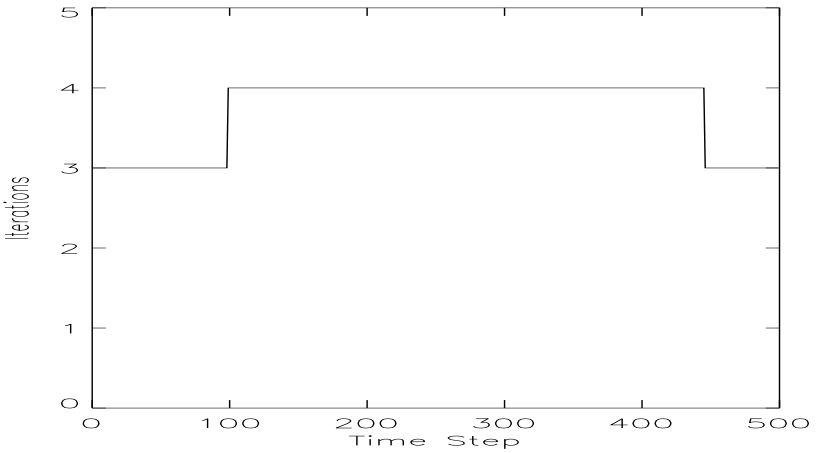

Figure 2.27. The number of iterations per time step needed to satisfy the stopping criterion (2.4.11). This Figure corresponds to Example 5 of the text and Figure 2.24 above.

2.4.3 The Forced Heat Equation Our third example is the numerical calculation of solutions of the so-called forced heat equation

$$u_t = \nu u_{xx} + f(u) \quad 0 \leq x \leq 1 \quad t \geq 0 \tag{2.4.48}$$

for $\nu > 0$, together with an initial condition

$$u(x,0) = u_0(x) \quad 0 \leq x \leq 1 \tag{2.4.49}$$

and periodic boundary conditions

$$u(0,t) = u(1,t) = 0 \quad t \geq 0. \tag{2.4.50}$$

It is well-known that solutions of this type of problem exhibit blowing-up behavior, see e.g. [35, 36] and references therein. For various specific nonlinear functions $f(u) = u^p$, $f(u) = e^u$, etc. the behavior of the solution has been studied in detail and a number of specialized numerical techniques have been developed, see e.g. [37, 38]. which use for example repeated rescaling, However these numerical schemes typically rescale the solution using thresholding mechanisms, which depend on the particular choice of the function $f(u)$ and tend to be rather ad hoc.

In this Section we apply our approach to the simple test case where $f(u) = u^3$, $\nu = 0.001$ and $\Delta t = 0.01$. We fix $n = 12$, $J = 5$, and $\epsilon = 10^{-6}$ and use 'Coiflets' in order to apply the multiplication algorithm described in Section 2.3. We have included this example in order to illustrate the straightforward applicability of our approach: we have simply *used* the algorithms much in the same way one would use a 'canned' FFT subroutine. In each example we have use the stopping criterion (2.4.41).

Figure 2.28 illustrates the evolution of the solution of (2.4.48) with the

initial condition

$$u_0(x) = 1 + e^{-c_1(x-1/2)^2}, \tag{2.4.51}$$

where $c_1 = 500$. We note that one may view the relationship between the subspaces $\mathbf{V}_j$ and $\mathbf{W}_j$ as an analogue of a rescaled coordinate system. Additionally, the only 'algorithmic' parameter introduced by our approach is the cutoff ϵ. Figure 2.29 illustrates the evolution of the solution of (2.4.48) with the initial condition

$$u_0(x) = 1 + \frac{1}{4}e^{-c_1(x-1/4)^2} + e^{-c_1(x-3/4)^2}, \tag{2.4.52}$$

where $c_1 = 500$. We note that our algorithm automatically distinguishes between the humps by placing significant wavelet coefficients in the vicinity of large gradients in the solution.

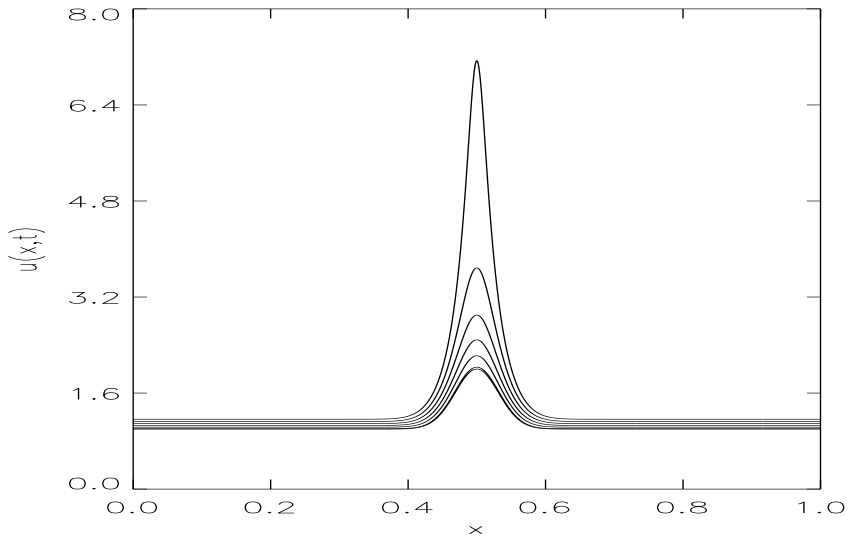

Figure 2.28. The projection on $\mathbf{V}_0$ of the solution of (2.4.48) with $f(u) = u^3$ and using the initial condition (2.4.51). In this experiment $N = 2^{12}$, $J = 5$, $\nu = 0.001$, $\Delta t = 0.01$ and $\epsilon = 10^{-6}$.

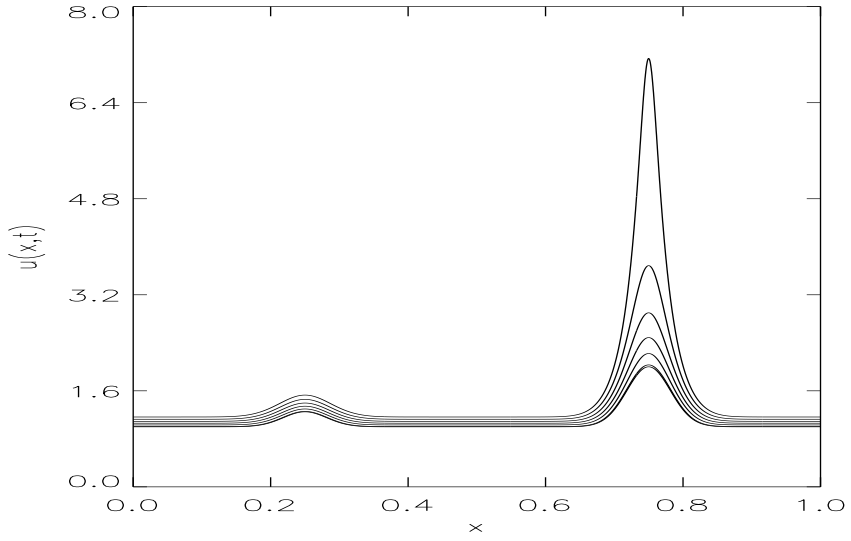

Figure 2.29. The projection on $\mathbf{V}_0$ of the solution of (2.4.48) with $f(u) = u^3$ and using the initial condition (2.4.52). In this experiment $N = 2^{12}$, $J = 5$, $\nu = 0.001$, $\Delta t = 0.01$ and $\epsilon = 10^{-6}$.

2.5 Conclusions

In this Chapter we have introduced new algorithms for the fast, adaptive numerical solution of nonlinear partial differential equations of the form

$$u_t = \mathcal{L}u + \mathcal{N}f(u) \tag{2.5.1}$$

$$u(x,0) = u_0(x) \qquad 0 \leq x \leq 1 \tag{2.5.2}$$

$$u(0,t) = u(1,t) \qquad 0 \leq t \leq T \tag{2.5.3}$$

for the unknown function $u = u(x,t)$. The differential operators $\mathcal{L}$ and $\mathcal{N}$ are assumed to be time-independent and the function $f(u)$ is nonlinear. The solution $u(x,t)$ of (2.5.1) with (2.5.2) and (2.5.3) typically possess smooth and shock-like behavior and we have demonstrated an approach which combines the desirable features of finite difference approaches, spectral methods and front-tracking or adaptive grid approaches usually applied to such problems.

We have introduced two new efficient, generic algorithms which use wavelet expansions of the functions and operators to compute solutions of (2.5.1). These algorithms take advantage of the fact that wavelet expansions may be viewed as a localized Fourier analysis with multiresolution structure which accommodates both smooth and shock-like behavior in the solution. In smooth regions few wavelet coefficients are needed and in singular regions large variations in the solution require more wavelet coefficients. The need for fast, adaptive algorithms for computing solutions of (2.5.1) motivated us to develop the algorithms introduced in this Chapter. These algorithms have complexity proportional to the number of significant coefficients in the wavelet expansions of solutions of (2.5.1).

To summarize, in Section 2.1 we used the semigroup approach to express the solution of (2.5.1) with (2.5.2) and (2.5.3) in terms of the nonlinear integral equation

$$u(x,t) = e^{(t-t_0)\mathcal{L}}u_0(x) + \int_{t_0}^{t} e^{(t-\tau)\mathcal{L}}\mathcal{N}f(u(x,\tau))d\tau. \tag{2.5.4}$$

We introduced a method for approximating the nonlinear integral equation to an arbitrary order of accuracy. The results of Section 2.1 are exemplified by equations (2.1.17)-(2.1.20) which are solutions of (2.5.1) written in terms of operators. The matrices representing these operators have dense representations in traditional approaches (e.g. finite differences) and lead to computationally expensive algorithms. As far as we know a direct efficient or adaptive numerical method for solving (2.5.1) based on the semigroup approach has not been formulated.

In Section 2.2 we compute the nonstandard form representation of the operators appearing in the approximations of (2.5.4). We then prove the vanishing-moment property of the B^j blocks of the NS-form representation of differential operators and the Hilbert transform. Additionally we prove this result for the NS-form representation of the operator functions appearing in the approximation of (2.5.4). This property is the basis for a rapid, adaptive algorithm for applying the NS-form representation of operators to the wavelet expansion of the solution $u(x,t)$ of (2.5.1). Section 2.2.3 and Appendix C provide a description of this algorithm and the corresponding pseudocode. Applying operators to functions via this algorithm is computationally proportional to the number of significant coefficients in the wavelet representation of the function. This complexity is a significant increase the efficiency of similar algorithms in traditional numerical methods.

In Section 2.3 we described an adaptive approach to computing functions $f(u)$ where $u(x,t)$ is expanded in a wavelet basis. In particular we addressed the question of computing the pointwise square, i.e. $f(u) = u^2$. We showed that the coefficients of the projection of $u(x,t)$ onto subspaces $\mathbf{W}_j$ may be used to identify or 'mask' coefficients in subspaces $\mathbf{V}_j$. These masked coefficients are then used in conjunction with the wavelet coefficients on $\mathbf{W}_j$ to evaluate u^2 via the so-called paraproduct, (2.3.17)

$$(P_0 u)^2 = (P_J u)^2 + \sum_{j=j_f}^{J} 2(P_j u)(Q_j u) + (Q_j u)^2. \tag{2.5.5}$$

Once again the computational complexity of computing (2.5.5) is proportional to the number of significant coefficients in the wavelet representation of $u(x,t)$. We then conclude with a discussion of the problem of adaptively computing more general functions $f(u)$ in wavelet bases using a number of operations which is proportional to the number of coefficients in the wavelet representation of $u(x,t)$.

In order to illustrate our algorithms we compute approximations to the solutions of several model equations in Section 2.4. We first illustrated our methodology by approximating the solution of the linear heat equation and comparing our results with the exact solution and the approximation arrived at by the well-known Crank-Nicolson method. The purpose of this comparison is to illustrate that the wavelet representation of operators for diffusion-type equations are sparser for high order methods (for a given accuracy) than the corresponding operators for lower order methods. Additionally we compute approximations to the solutions of Burgers' equation and a version of the forced heat equation. In the case of Burgers' equation we provide a number of examples which illustrate the impact of different viscosities on the parameters associated with the choice

of wavelet basis and the accuracy of the approximation. Decreasing the viscosity results in the formation of a sharper shock. In traditional front-tracking or adaptive grid approaches, decreasing the viscosity requires one to incorporate new grid points near the shock; this may be done in an essentially ad hoc fashion. In traditional our wavelet based approach decreasing the viscosity requires the introduction of a greater number of scales in the wavelet representation of the solution. In our wavelet based approach new basis functions or wavelet coefficients are automatically and adaptively introduced based on the sharpness of the solution. Moreover, the number of calculations required to introduce new basis functions is proportional to the number of coefficients needed to represent the solution at the previous time step. The example of the forced heat equation was included to illustrate the genericity and flexibility of our algorithms and the overall approach.

2.5.1 Future Directions There are several directions for this course of work which we have left for the future. For example, one may consider nonperiodic boundary conditions instead of the periodic boundary condition (2.5.3). We note that variable coefficients in the linear terms of the evolution equation (2.5.1) may be accommodated by computing the NS-form of the corresponding operators. Another direction has to do with the choice of the wavelet basis. One of the conclusions which we have drawn from this study is that there seems to be a number of advantages to using basis functions which are piecewise polynomial. In particular the spline family of bases appears to be attractive as well as multiwavelets. In both cases there are also disadvantages and an additional study might be necessary to understand such a tradeoff. Yet another generalization is to consider multidimensional problems. An example of this

type of problem is the Navier-Stokes equations.

BIBLIOGRAPHY

[1] J. von Neumann. Theory of Self-Reproducing Automata, edited and completed by A. Burks. University of Illinois Press, Champaign, IL, (1966).

[2] B. Madore and W. Freedman. Computer Simulation of the Belousov-Zhabotinskii Reaction. *Science* **222**, p615 (1983).

[3] J. Greenberg and S. Hastings. Spatial Patterns for Discrete Models of Diffusion in Excitable Media. *SIAM J. Appl. Math.* **34**, p515 (1978).

[4] J. Hardy and Y. Pomeau. Thermodynamics and Hydrodynamics for a Modeled Fluid. *J. Math. Phys.* **13**, p1042 (1972).

[5] U. Frisch, B. Hasslacher, and Y. Pomeau. Lattice Gas Automata for the Navier-Stokes Equation. *Phys. Rev. Lett.* **56**, p1505 (1986).

[6] J. Hardy, O. de Pazzis, Y. Pomeau. Time evolution of a two-dimensional model system: I. Invariant states and time correlation functions. *J. Math. Phys.* **14**, p1746 (1973).

[7] H.A. Lim. Lattice Gas Automata of Fluid Dynamics for Unsteady Flows. *Complex Systems* **2**, p45 (1988).

[8] B. Boghosian and C. Levermore. A Cellular Automata for Burgers' Equation. *Complex Systems* **1**, p17 (1987).

[9] J. Lebowitz, E. Orlandi, E. Presutti. Convergence of stochastic cellular automata to Burgers' equation: fluctuations and stability. *Physica D* **33**, p165 (1988).

[10] J. Greenberg and S. Hastings. Spatial Patterns for Discrete Models of Diffusion in Excitable Media. *SIAM J. Appl. Math.* **34**, p515 (1978).

[11] N. Margolus, T. Toffoli, and G. Vichniac, Cellular automata supercomputers for fluid dynamics modeling, *Phys. Rev. Lett.* **56**, p1696 (1986).

[12] S. Wolfram. Theory and Application of Cellular Automata. World Scientific, Singapore (1986).

[13] Cellular Automata: Theory and Experiment. Proceedings of a Workshop Sponsored by the Center for Nonlinear Studies, Los Alamos National Laboratory. *Physica D* **45**, September 1990.

[14] K. Steiglitz, I. Kamal, and A. Watson. Embedding Computation in One-Dimensional Automata by Phase Coding Solitons. *IEEE Trans. on Computers.* **37**, (1988).

[15] J. Park, K. Steiglitz, and W. Thurston. Soliton-Like Behaviour in Automata. *Physica D* **19**, p423 (1986).

[16] M.J. Ablowitz and H. Segur. Solitons and the Inverse Scattering Transform. SIAM, Philadelphia (1981).

[17] L. Faddeev and L.A. Takhtajan. Hamiltonian Methods in the Theory of Solitons. Springer-Verlag (1987).

[18] T. S. Papatheodorou, M. J. Ablowitz, and Y. G. Saridakis. A Rule for the Fast Computation and Analysis of Soliton Automata. *Stud. Appl. Math.* **79**, p173 (1988).

[19] A. S. Fokas, E. Papadopoulou, Y. G. Saridakis, M. J. Ablowitz. *Stud. Appl. Math.* **81**, p153 (1989).

[20] J. M. Keiser. On the Computation of Periodic Particles for Cellular Automata. Masters Thesis, Clarkson University (1989).

[21] M.J. Ablowitz and J.M. Keiser. On Particles and Interaction Properties of the Parity Rule Filter Automata. PAM report 68 (1990).

[22] M.J. Ablowitz, B.M. Herbst, and J.M. Keiser. Solitons, Numerical Chaos and Cellular Automata, in Integrable and Super-Integrable Systems. Ed. B.A. Kuperschmidt. World Scientific (1990).

[23] C. H. Goldberg. Parity Filter Automata. *Complex Systems* **2**, p91 (1988).

[24] M.J. Ablowitz, J.M. Keiser and L.A. Takhtajan. A class of stable multi-state time-reversible cellular automata with rich particle content. *Phys. Rev. A* **44**, p6909 (1991).

[25] R. Lidl and H. Niederreiter. Introduction to Finite Fields and their Applications. Cambridge University Press (1986).

[26] M. Bruschi and P.M. Santini. Cellular automata in 1+1, 2+1 and 3+1 dimensions, constants of motion and coherent structures. *Physica D* **70**, p185 (1994).

[27] M. Bruschi and P.M. Santini. Integrable cellular automata. *Physics letters, A* **169**, p151 (1992).

[28] J. Stoer and R. Burlisch. Introduction to Numerical Analysis. Springer-Verlag, (1980).

[29] G. Dahlquist and A. Björck. Numerical Methods. Prentice-Hall, (1974).

[30] A. Pazy. Semigroups of Linear Operators and Applications to Partial Differential Equations. Springer-Verlag, (1983).

[31] G.B. Whitham. Linear and Nonlinear Waves. Wiley, (1974).

[32] J.M. Burgers. A mathematical model illustrating the theory of turbulence. *Adv. Appl. Mech.* **1**, p171 (1948).

[33] E. Hopf. The Partial Differential Equation $u_t + uu_x = \mu u_{xx}$. *Comm. on Pure and Appl. Math.* **3**, p201, (1950).

[34] J. D. Cole. On a quasilinear parabolic equation occurring in aerodynamics. *Quart. App. Math.* **9**, p225 (1951).

[35] J. Bebernes and A. Lacey. Finite-Time Blowup for a Particular Parabolic System. *SIAM J. Math. Analysis* **21**, p1415 (1990).

[36] J. Bebernes and S. Bricher. Final Time Blowup Profiles for Semilinear Parabolic Equations via Center Manifold Theory. *SIAM J. Math. Analysis* **23**, p852 (1992).

[37] M. Berger and R. V. Kohn. A Rescaling Algorithm for the Numerical Calculation of Blowing-up Solutions. *Comm. Pure and Appl. Math.* **41**, p841 (1988).

[38] W. Huang, Y. Ren, and R. Russell. Moving Mesh Methods Based on Moving Mesh Partial Differential Equations. *J. Comp. Phys.* **113**, p279 (1994).

[39] K. Yosida. Functional Analysis. Springer-Verlag, (1980).

[40] P. Constantin, P.D. Lax, A. Majda. A Simple One-dimensional Model for the Three-dimensional Vorticity Equation. *Comm. Pure and Appl. Math.* **38**, p715 (1985).

[41] S. Mallat. Multiresolution Approximation and Wavelets. Technical Report, GRASP LAB, Department of Computer and Information Science, University of Pennsylvania.

[42] I. Daubechies. Orthonormal bases of compactly supported wavelets. *Comm. Pure and Appl. Math.* **41** p909 (1988).

[43] I. Daubechies. Ten Lectures on Wavelets. CBMS-NSF Series in Applied Mathematics. SIAM (1992).

[44] B. Alpert. Sparse Representation of Smooth Linear Operators. Ph.D. Thesis, Yale University (1990).

[45] G. Beylkin. On the representation of operators in bases of compactly supported wavelets. *SIAM J. Numer. Anal.* (1992).

[46] G. Beylkin. Wavelets and Fast Numerical Algorithms. *Proceedings of Symposia in Applied Mathematics*, **47** (1993).

[47] G. Beylkin. Wavelets, Multiresolution Analysis and Fast Numerical Algorithms. *A draft of INRIA Lecture Notes*, (1991).

[48] G. Beylkin, R. R. Coifman, and V. Rokhlin. Fast wavelet transforms and numerical algorithms I. *Comm. Pure and Appl. Math.*, 44:141–183, 1991. Yale University Technical Report YALEU/DCS/RR-696, August (1989).

[49] G. Beylkin, R. R. Coifman and V. Rokhlin. Wavelets in Numerical Analysis. *Wavelets and Their Applications*, p181 Eds. M.B.Ruskai et. al. Jones and Bartlett, (1992).

[50] J. M. Bony. Calcul symbolique et propagation des singularités pour les équations aux dérivées partielles non-linéaires. *Ann. Scient. E.N.S.*, **14** p209 (1981).

[51] R. R. Coifman and Y. Meyer. Au delà des opérateurs pseudo-différentiels. In *Astérisque, 57, (seconde édition revue et augmentée)*. Société Mathématique de France.

[52] I. Daubechies and J. Lagarius. Two-scale difference equations, I. Global regularity of solutions & II. Local regularity, infinite products of matrices and fractals. *SIAM J. Math. Anal.*, (1991).

[53] Charles K. Chui. An Introduction to Wavelets. Academic Press (1992).

[54] Y. Meyer. Wavelets and operators. *Cambridge studies in advanced mathematics*, **37** (1992).

[55] Y. Meyer. Le Calcul Scientifique, les Ondelettes et les Filtres Miroirs en Quadrature. Centre de Recherche de Mathématiques de la Décision. Report 9007.

[56] M.V. Wickerhauser. Adapted Wavelet Analysis from Theory to Software. A. K. Peters, Ltd. Wellesley, Massachusetts (1994).

[57] Eisenstat, et al. Yale Sparse Matrix Package, I. The Symmetric Codes. Yale Technical Report #112.

[58] H.-O. Kreisss and J. Oliger. Comparison of accurate methods for the integration of hyperbolic equations. *Tellus* **24**, p199 (1972).

[59] B. Fornberg. On a Fourier Method for the Integration of Hyperbolic Equations. *SIAM J. Numer. Anal.* **12**, p509 (1975).

[60] B. Fornberg and G.B. Whitham. A numerical and theoretical study of certain nonlinear wave phenomena. *Phil. Trans. R. Soc. Lond.*, **289**, p373 (1978).

[61] I. Christie, D.F. Griffiths, A.R. Mitchell, and J.M. Sanz-Serna. Product Approximation for Non-linear Problems in the Finite Element Method. *IMA Journal of Numerical Analysis* **1**, p253 (1981).

[62] I. Christie and J.M. Sanz-Serna. Petrov-Galerkin Methods for Nonlinear Dispersive Waves. *Journal of Computational Physics* **39**, p94 (1981).

[63] R.L. Schult and H.W. Wyld. Using Wavelets to Solve the Burgers' Equation: A Comparative Study. *Physical Review A*, **46**, p12 (1992).

[64] J. Liandrat, V. Perrier and Ph. Tchamitchian. Numerical Resolution of Nonlinear Partial Differential Equations using the Wavlet Approach. Wavelets and Their Applications. Eds: M.B. Ruskai, G. Beylkin, R. Coifman, I. Daubechies, S. Mallat, Y. Meyer and L. Raphael. Jones and Bartlett Publishing, Inc. (1992).

APPENDIX A

PRELIMINARIES AND CONVENTIONS OF WAVELET ANALYSIS

In this Appendix we briefly review the notation associated with wavelet basis expansions of functions and operators. We begin in Appendix A.1 by setting the notation associated with multiresolution analysis, which is the framework for wavelet analysis. The representation of functions expanded in wavelet bases is described in Appendix A.2. Expansions of operators in wavelet bases take two natural forms and in Appendix A.3 we describe the construction and properties of the standard and non-standard forms of operators. In Appendix A.4 we review [45] and discuss the construction of the non-standard form of differential operators.

A.1 Multiresolution Analysis

We start with the notion of a multiresolution analysis (MRA). An MRA is a decomposition of a Hilbert space, e.g. $\mathbf{L}^2(\mathbb{R})$, into a chain of closed subspaces

$$\ldots \subset \mathbf{V}_2 \subset \mathbf{V}_1 \subset \mathbf{V}_0 \subset \mathbf{V}_{-1} \subset \mathbf{V}_{-2} \subset \ldots. \tag{A.1.1}$$

that satisfy properties specified in Appendix A. We define an associated sequence of subspaces $\mathbf{W}_j$ as the orthogonal complements of $\mathbf{V}_j$ in $\mathbf{V}_{j-1}$,

$$\mathbf{V}_{j-1} = \mathbf{V}_j \bigoplus \mathbf{W}_j. \tag{A.1.2}$$

Repeatedly using (A.1.2) shows that subspace $\mathbf{V}_j$ can be written as the direct sum of subspaces $\mathbf{W}_{j'}$

$$\mathbf{V}_j = \bigoplus_{j'>j} \mathbf{W}_{j'}. \tag{A.1.3}$$

For functions of one variable, the set of dilations and translations of the scaling function $\varphi(\cdot)$, $\{\varphi_{j,k}(x) = 2^{-j/2}\varphi(2^{-j}x - k)\}_{k\in\mathbb{Z}}$, forms an orthonormal basis of $\mathbf{V}_j$ and the set of dilations and translations of the wavelet $\psi(\cdot)$, $\{\psi_{j,k}(x) = 2^{-j/2}\psi(2^{-j}x - k)\}_{k\in\mathbb{Z}}$, forms an orthonormal basis of $\mathbf{W}_j$. The scaling function $\varphi(x)$ satisfies the two-scale difference equation

$$\varphi(x) = \sqrt{2}\sum_{k=0}^{L-1} h_k\varphi(2x - k) \tag{A.1.4}$$

and the wavelet $\psi(x)$ is defined by

$$\psi(x) = \sqrt{2}\sum_{k=0}^{L-1} g_k\varphi(2x - k). \tag{A.1.5}$$

The sets of coefficients $H = \{h_k\}$ and $G = \{g_k\}$ are called Quadrature Mirror Filters (QMF's) that, once chosen, define a particular wavelet basis. The integer L is the length of the QMF and is related to the number of vanishing moments M of the wavelet $\psi(x)$, e.g. $L = 2M$ for Daubechies wavelets, [42].

Here we have fixed the integer $L < \infty$ which means that we are considering compactly supported wavelets. Although some of our algorithms specifically rely on the finite support of wavelets, similar considerations are applicable to wavelets without compact support.

A.2 Representation of Functions in Wavelet Bases

The projection of a function $f(x)$ onto subspace $\mathbf{V}_j$ is given by

$$(P_j f)(x) = \sum_{k\in\mathbb{Z}} s_k^j\varphi_{j,k}(x) \tag{A.2.6}$$

where P_j denotes the projection operator onto subspace $\mathbf{V}_j$. The set of coefficients $\{s_k^j\}_{k\in\mathbb{Z}}$, which we refer to as 'averages', is computed via the inner product

$$s_k^j = \int_{-\infty}^{+\infty} f(x)\varphi_{j,k}(x)dx. \tag{A.2.7}$$

It follows from (A.1.3) and (A.2.6) that we can also write $(P_j f)(x)$ as a sum of projections of $f(x)$ onto subspaces $\mathbf{W}_{j'}, j' > j$

$$(P_j f)(x) = \sum_{j'>j} \sum_{k \in \mathbb{Z}} d_k^{j'} \psi_{j',k}(x) \tag{A.2.8}$$

where the set of coefficients $\{d_k^j\}_{k \in \mathbb{Z}}$, which we refer to as 'differences', is computed via the inner product

$$d_k^j = \int_{-\infty}^{+\infty} f(x) \psi_{j,k}(x) dx. \tag{A.2.9}$$

The projection of a function on subspace $\mathbf{W}_j$ is denoted $(Q_j f)(x)$, where $Q_j = P_{j-1} - P_j$.

For numerical purposes we define a 'finest' scale, $j = 0$, and a 'coarsest' scale, $j = J$, such that the infinite chain (A.1.1) is restricted to

$$\mathbf{V}_J \subset \mathbf{V}_{J-1} \subset \ldots \subset \mathbf{V}_0. \tag{A.2.10}$$

We will also consider a periodized version of the multiresolution analysis that is obtained if we consider periodic functions. Such a periodization is the simplest, but not the most efficient or elegant, way to consider the multiresolution analysis of a function on an interval. The problem with periodization is that an arbitrary smooth function on an interval is not necessarily periodic and we can therefore introduce an artificial singularity at the boundary. A more elegant approach would use wavelets on the interval, [42], or multiwavelets, [44]. We choose to consider the periodization described here since it is the easiest way to describe the adaptive algorithms we are developing and the approach does not change substantially if we use other bases. We will therefore consider functions of period $N = 2^n$ and their projections on $\mathbf{V}_0$. With a slight abuse of notation we will denote these periodized subspaces also by $\mathbf{V}_j$ and $\mathbf{W}_j$.

We can then view the space $\mathbf{V}_0$ as consisting of 2^n 'samples' or lattice points and each space $\mathbf{V}_j$ and $\mathbf{W}_j$ as consisting of 2^{n-j} lattice points, for $j =$

$1, 2, \ldots, J$. On each subspace $\mathbf{V}_j$ and $\mathbf{W}_j$ the coefficients of the projections satisfy

$$\begin{aligned} s_k^j &= s_{k+2^{n-j}}^j \\ d_k^j &= d_{k+2^{n-j}}^j \end{aligned} \tag{A.2.11}$$

for each $j = 1, 2, \ldots, J$ and $k \in \mathbb{F}_{2^{n-j}} = \mathbb{Z}/2^{n-j}\,\mathbb{Z}$, i.e. $\mathbb{F}_{2^{n-j}}$ is the finite field of 2^{n-j} integers, e.g. the set $\{0, 1, \ldots, 2^{n-j} - 1\}$.

The expansion into the wavelet basis of the projection of a function $f(x)$ on $\mathbf{V}_0$ is given by a sum of successive projections on subspaces $\mathbf{W}_j$, $j = 1, 2, \ldots, J$, and a final 'coarse' scale projection on $\mathbf{V}_J$,

$$(P_0 f)(x) = \sum_{j=1}^{J} \sum_{k \in \mathbb{F}_{2^{n-j}}} d_k^j \psi_{j,k}(x) + \sum_{k \in \mathbb{F}_{2^{n-J}}} s_k^J \varphi_{J,k}(x). \tag{A.2.12}$$

Given the set of coefficients $\{s_k^0\}_{k \in \mathbb{F}_{2^n}}$, i.e. the coefficients of the projection of $f(x)$ on $\mathbf{V}_0$, we use (A.1.4) and (A.1.5) to replace (A.2.7) and (A.2.9) by the following recursive definitions of s_k^j and d_k^j,

$$s_k^j = \sum_{l=1}^{L-1} h_l s_{l+2k+1}^{j-1} \tag{A.2.13}$$

$$d_k^j = \sum_{l=1}^{L-1} g_l s_{l+2k+1}^{j-1}, \tag{A.2.14}$$

where $j = 1, 2, \ldots, J$ and $k \in \mathbb{F}_{2^{n-j}}$.

Given the coefficients $s^0 = P_0 f \in \mathbf{V}_0$ consisting of $N = 2^n$ 'samples' the decomposition of f into the wavelet basis is an order N procedure, i.e. computing the coefficients d_k^j and s_k^j recursively using (A.2.13) and (A.2.14) is an order N algorithm. Computing the J-scale decomposition of f via (A.2.13) and (A.2.14) by the Laplacian pyramid scheme is illustrated in Figure A.1.

Figure A.2 illustrates a typical wavelet representation of a function with $N = 2^n$, $n = 13$ and $J = 7$. Although we have generated this Figure using 'Coiflets', see e.g. [42], with $M = 6$ vanishing moments and an accuracy (cutoff) of $\epsilon = 10^{-6}$, the distribution of significant coefficients is typical for other wavelet

$$\begin{array}{ccccccccccc} \{s_k^0\} & \longrightarrow & \{s_k^1\} & \longrightarrow & \{s_k^2\} & \longrightarrow & \{s_k^3\} & \cdots & \longrightarrow & \{s_k^J\} \\ & \searrow & & \searrow & & \searrow & & & \searrow & \\ & & \{d_k^1\} & & \{d_k^2\} & & \{d_k^3\} & \cdots & & \{d_k^J\}. \end{array}$$

Figure A.1. Projection of the coefficients $\{s_k^0\}$ into the multiresolution analysis via the Laplacian pyramid scheme.

bases. The top Figure is a graph of the projection of the function f on subspace $\mathbf{V}_0$, which we note is a space of dimension 2^{13}. Each of the next $J = 7$ graphs represents the projection of f on subspaces $\mathbf{W}_j$, for $j = 1, 2, \ldots 7$. Each $\mathbf{W}_j$ is a space of dimension 2^{13-j}, i.e. each consists of 2^{13-j} grid points. Even though the width of the graphs is the same, we note that the number of grid points that discretize $\mathbf{W}_j$ is twice the number of grid points that discretize $\mathbf{W}_{j+1}$. Since these graphs illustrate coefficients d_k^j which are above the threshold of accuracy, ϵ, we note that the spaces $\mathbf{W}_1$, $\mathbf{W}_2$, $\mathbf{W}_3$, and $\mathbf{W}_4$ consist of no significant wavelet coefficients. This illustrates the 'compression' property of the wavelet transform: regions where the function (or its projection $(P_0 f) = f_0$) has large gradients are transformed to significant wavelet coefficients. The final (bottom) graph represents the significant coefficients of the projection of f on $\mathbf{V}_J$. This set of coefficients, $\{s_k^J\}_{k \in \mathbb{F}_{2^6}}$, is typically dense and in this example there are 61 significant coefficients.

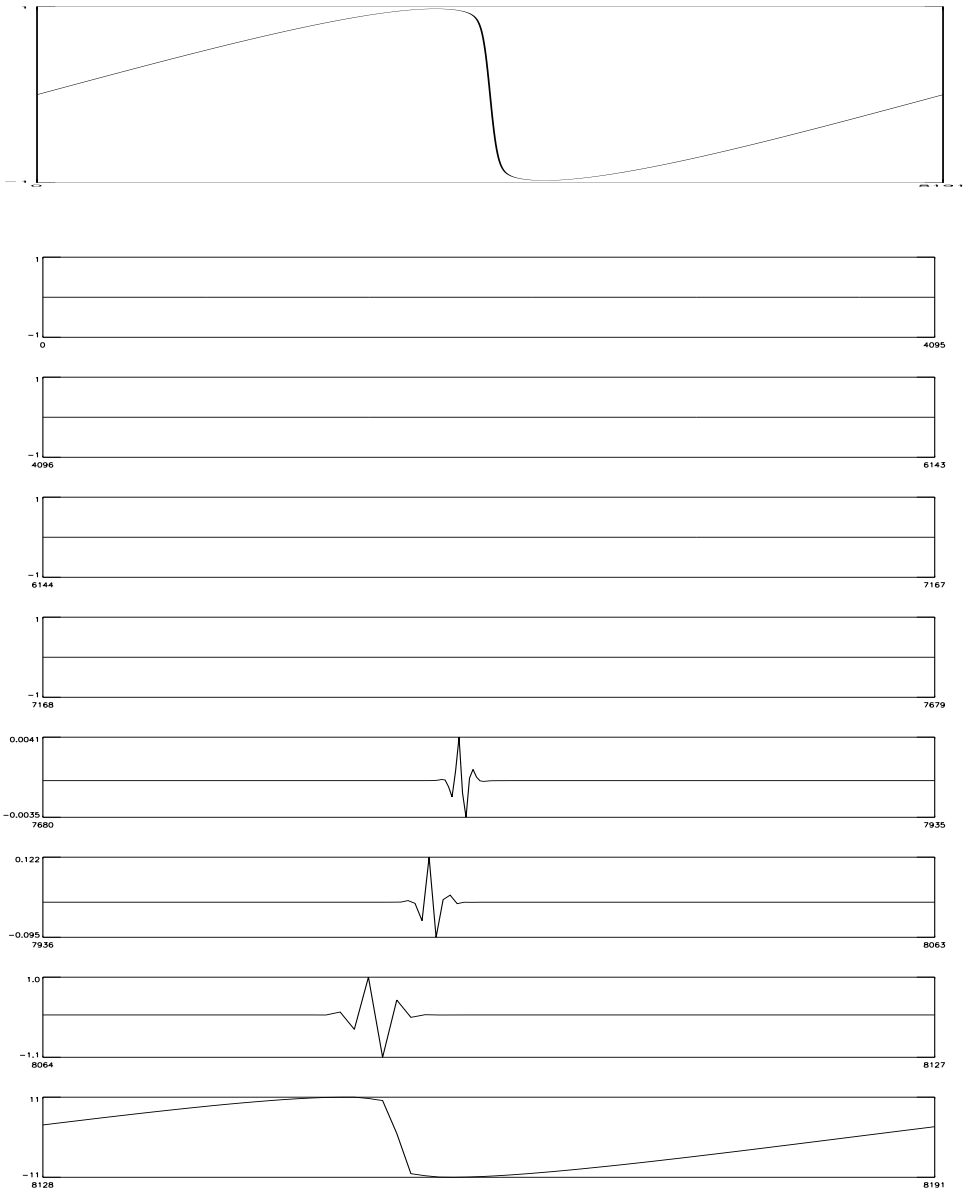

Figure A.2. Graphical representation of a 'sampled' function on $\mathbf{V}_0$ and its projections onto $\mathbf{W}_j$ for $j = 1, 2, \ldots 7$ and $\mathbf{V}_7$. Entries above the threshold of accuracy, $\epsilon = 10^{-6}$, are shown. We refer to the text for a full description of this Figure.

A.3 The Standard and Non-Standard Form of Operators

In order to represent an operator $T : \mathbf{L}^2(\mathbb{R}) \to \mathbf{L}^2(\mathbb{R})$ in the wavelet system of coordinates, we consider two natural ways to define two-dimensional wavelet bases. First, we consider a two-dimensional wavelet basis which is arrived at by computing the tensor product of two one-dimensional wavelet basis functions, e.g.

$$\psi_{j,j',k,k'}(x,y) = \psi_{j,k}(x)\psi_{j',k'}(y) \tag{A.3.15}$$

where $j, j', k, k' \in \mathbb{Z}$. This choice of basis leads to the standard form (S-form) of an operator, [45, 48]. The projection of the operator T into the multiresolution analysis is represented in the S-form by the set of operators

$$T = \{A_j, \{B_j^{j'}\}_{j' \geq j+1}, \{\Gamma_j^{j'}\}_{j' \geq j+1}\}_{j \in \mathbb{Z}}, \tag{A.3.16}$$

where the operators $A_j, B_j^{j'}$, and $\Gamma_j^{j'}$ are projections of the operator T into the multiresolution analysis as follows

$$\left.\begin{array}{rcl} A_j &=& Q_j T Q_j \quad : \quad \mathbf{W}_j \to \mathbf{W}_j \\ B_j^{j'} &=& Q_j T Q_{j'} \quad : \quad \mathbf{W}_{j'} \to \mathbf{W}_j \\ \Gamma_j^{j'} &=& Q_{j'} T Q_j \quad : \quad \mathbf{W}_j \to \mathbf{W}_{j'} \end{array}\right\} \tag{A.3.17}$$

for $j = 1, 2, \ldots, n$ and $j' = j+1, \ldots, n$.

If n is the finite number of scales, as in (A.2.10), then (A.3.16) is restricted to the set of operators

$$T_0 = \{A_j, \{B_j^{j'}\}_{j'=j+1}^{j'=n}, \{\Gamma_j^{j'}\}_{j'=j+1}^{j'=n}, B_j^{n+1}, \Gamma_j^{n+1}, T_n\}_{j=1,\ldots,n}, \tag{A.3.18}$$

where T_0 is the projection of T on $\mathbf{V}_0$. Here the operator T_n is the coarse scale projection of the operator T on $\mathbf{V}_n$,

$$T_n \quad = \quad P_n T P_n : \mathbf{V}_n \to \mathbf{V}_n. \tag{A.3.19}$$

The subspaces $\mathbf{V}_j$ and $\mathbf{W}_j$ appearing in (A.3.17) and (A.3.19) can be periodized in the same fashion as described above.

The operators $A_j, B_j^{j'}, \Gamma_j^{j'}$, and T_j appearing in (A.3.16) and (A.3.18) are represented by matrices $\alpha^j, \beta^{j,j'}, \gamma^{j,j'}$ and s^j with entries defined by

$$\left.\begin{aligned} \alpha^j_{k,k'} &= \int\int \psi_{j,k}(x)K(x,y)\psi_{j,k'}(y)dxdy \\ \beta^{j,j'}_{k,k'} &= \int\int \psi_{j,k}(x)K(x,y)\varphi_{j',k'}(y)dxdy \\ \gamma^{j,j'}_{k,k'} &= \int\int \varphi_{j,k}(x)K(x,y)\psi_{j',k'}(y)dxdy \\ s^j_{k,k'} &= \int\int \varphi_{j,k}(x)K(x,y)\varphi_{j,k'}(y)dxdy \end{aligned}\right\} \tag{A.3.20}$$

where $K(x,y)$ is the kernel of the operator T. The operators in (A.3.18) are organized as blocks of a matrix as shown in Figure A.3.

In [45] it is observed that if the operator T is a Calderón-Zygmund or pseudo-differential operator then, for a fixed accuracy, all the operators in (A.3.16) are banded. In the case of a finite number of scales the operator T_n and possibly some other operators on coarse scales can be dense. As a result the S-form has several 'finger' bands, illustrated in Figure A.4. These 'finger' bands correspond to interactions between different scales. For a large class of operators, e.g. pseudo-differential, the interaction between different scales (characterized by the size of the coefficients in the bands) decays as the distance $|j-j'|$ between the scales increases. Therefore, if the scales j and j' are well separated, then for a given accuracy, the operators $B_j^{j'}$ and $\Gamma_j^{j'}$ can be neglected. For compactly supported wavelets, the distance $|j-j'|$ is quite significant; in a typical example for differential operators $|j-j'|=6$. This is not necessarily the case for other families of wavelets. For example, Meyer's wavelets [54] are characterized by

$$\hat{\psi}(\xi) = \begin{cases} (2\pi)^{-1/2}e^{i\xi/2}\sin(\frac{\pi}{2}\nu(\frac{2}{3\pi}|\xi|-1)) & \frac{2\pi}{3} \le |\xi| \le \frac{4\pi}{3} \\ (2\pi)^{-1/2}e^{i\xi/2}\cos(\frac{\pi}{2}\nu(\frac{2}{3\pi}|\xi|-1)) & \frac{4\pi}{3} \le |\xi| \le \frac{8\pi}{3} \\ 0 & \text{otherwise} \end{cases} \tag{A.3.21}$$

where ν is a C^∞ function satisfying

$$\nu(x) = \begin{cases} 0 & x \le 0 \\ 1 & x \ge 1 \end{cases} \tag{A.3.22}$$

and

$$\nu(x) + \nu(1-x) = 1, \tag{A.3.23}$$

for example

$$\nu(x) = \begin{cases} 0 & x \leq 0 \\ \sin^2(\frac{\pi}{2}x) & 0 \leq x \leq 1 \\ 1 & x \geq 1 \end{cases} \tag{A.3.24}$$

(see e.g. [42]). In this case the interaction between scales for differential operators is restricted to nearest neighbors where $|j - j'| \leq 1$. On the other hand, Meyer's wavelets are not compactly supported in the time domain which means the finger bands will be much wider than in the case of compactly supported wavelets. The control of the interaction between scales is better in the non-standard representation of operators, which we discuss below.

Another property of the S-form which has an impact on numerical applications is due to the fact that the wavelet coefficients are not shift invariant. Even if the operator T is a convolution then the $B_j^{j'}$ and $\Gamma_j^{j'}$ blocks of the S-form are not convolutions. Thus the S-form of a convolution operator is not an efficient representation, especially in multiple dimensions.

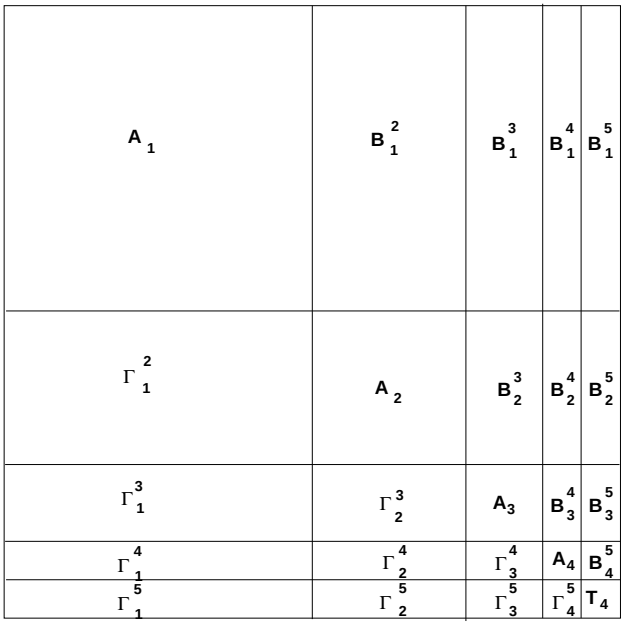

Figure A.3: Organization of the standard form of a matrix.

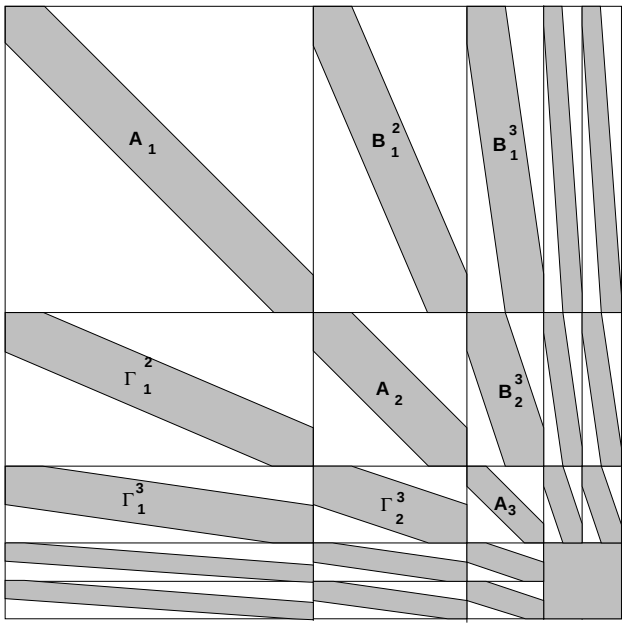

Figure A.4: Schematic illustration of the finger structure of the standard form.

An alternative to forming two-dimensional wavelet basis functions using the tensor product (which led us to the S-form representation of operators) is to consider functions which are combinations of the wavelet, $\psi(\cdot)$, and the scaling function, $\varphi(\cdot)$. We note that such an approach to forming basis elements in higher dimensions is specific to wavelet bases (tensor products as considered above can be used with any basis, e.g. Fourier). The wavelet representation of an operator in the non-standard form (NS-form) is arrived at using bases formed by combinations of wavelet and scaling functions, for example, in $\mathbf{L}^2(\mathbb{R}^2)$

$$\begin{aligned} &\psi_{j,k}(x)\ \psi_{j,k'}(y) \\ &\psi_{j,k}(x)\ \varphi_{j,k'}(y) \\ &\varphi_{j,k}(x)\ \psi_{j,k'}(y) \end{aligned} \tag{A.3.25}$$

where $j, k, k' \in \mathbb{Z}$. The NS-form of an operator T is obtained by expanding T in the 'telescopic' series

$$T = \sum_{j \in \mathbb{Z}} (Q_j T Q_j + Q_j T P_j + P_j T Q_j), \tag{A.3.26}$$

where P_j and Q_j are projectors on subspaces $\mathbf{V}_j$ and $\mathbf{W}_j$, respectively. We observe that in (A.3.26) the scales are decoupled. The expansion of T into the NS-form is thus represented by the set of operators

$$T = \{A_j, B_j, \Gamma_j\}_{j \in \mathbb{Z}}, \tag{A.3.27}$$

where the operators A_j, B_j, and Γ_j act on subspaces $\mathbf{V}_j$ and $\mathbf{W}_j$ as follows

$$\left.\begin{aligned} A_j &= Q_j T Q_j \ : \ \mathbf{W}_j \to \mathbf{W}_j \\ B_j &= Q_j T P_j \ : \ \mathbf{V}_j \to \mathbf{W}_j \\ \Gamma_j &= P_j T Q_j \ : \ \mathbf{W}_j \to \mathbf{V}_j \end{aligned}\right\} \tag{A.3.28}$$

see e.g. [48].

If $J \leq n$ is the finite number of scales, as in (A.2.10), then (A.3.26) is truncated to

$$T_0 = \sum_{j=1}^{J} (Q_j T Q_j + Q_j T P_j + P_j T Q_j) + P_J T P_J, \tag{A.3.29}$$

and the set of operators (A.3.27) is restricted to

$$T_0 = \{\{A_j, B_j, \Gamma_j\}_{j=1}^{j=J}, T_J\}, \tag{A.3.30}$$

where T_0 is the projection of the operator on $\mathbf{V}_0$ and T_J is a coarse scale projection of the operator T

$$T_J = P_J T P_J : \mathbf{V}_J \to \mathbf{V}_J \tag{A.3.31}$$

using, e.g. in $\mathbf{L}^2(\mathbb{R}^2)$, the basis functions

$$\varphi_{J,k}(x)\ \varphi_{J,k'}(y), \tag{A.3.32}$$

for $k, k' \in \mathbb{Z}$.

The operators A_j, B_j, Γ_j and T_J appearing in the NS-form are represented by matrices $\alpha^j, \beta^j, \gamma^j$, and s^j with entries defined by

$$\left.\begin{aligned} \alpha^j_{k,k'} &= \int\int K(x,y)\psi_{j,k}(x)\psi_{j,k'}(y)dxdy \\ \beta^j_{k,k'} &= \int\int K(x,y)\psi_{j,k}(x)\varphi_{j,k'}(y)dxdy \\ \gamma^j_{k,k'} &= \int\int K(x,y)\varphi_{j,k}(x)\psi_{j,k'}(y)dxdy \\ s^j_{k,k'} &= \int\int K(x,y)\varphi_{j,k}(x)\varphi_{j,k'}(y)dxdy \end{aligned}\right\} \tag{A.3.33}$$

in $\mathbf{L}^2(\mathbb{R}^2)$. The operators in (A.3.30) are organized as blocks of a matrix as shown in Figure A.5.

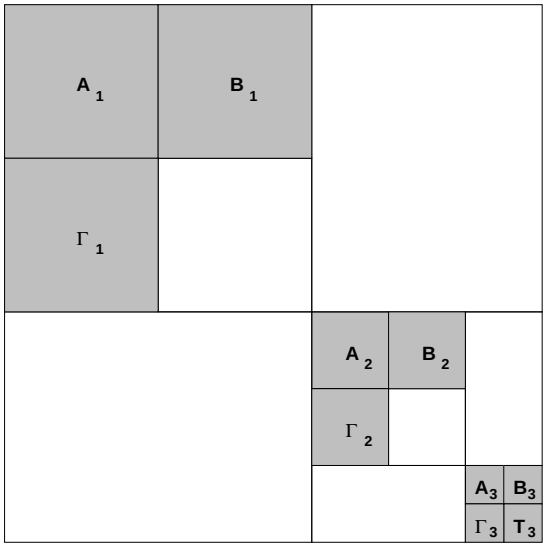

Figure A.5. Organization of the non-standard form of a matrix. A_j, B_j, and Γ_j, $j = 1, 2, 3$, and T_3 are the only non-zero blocks.

The price of uncoupling the scale interactions in (A.3.26) is the need for an additional projection into the wavelet basis of the product of the NS-form and a function. The term non-standard form comes from the fact that the vector to which the NS-form is applied is not a representation of the original vector in the wavelet basis. Referring to Figure A.6 we se that the NS-form is applied to both averages and differences of the wavelet expansion of a function. In this case we can view the multiplication of the NS-form and a function as an embedding of matrix-vector multiplication into a space of dimension

$$M = 2^{n-J}(2^{J+1} - 1) \tag{A.3.34}$$

where n is the number of scales in the wavelet expansion and $J \leq n$ is the depth of the expansion. This result must then be projected back into the original space of dimension $N = 2^n$. We note that in general $M > N$ and for $J = n$ we have $M = 2N - 1$.

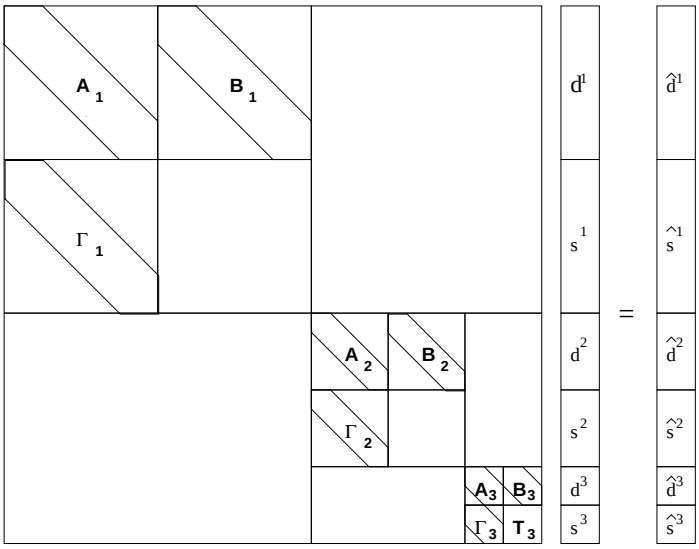

Figure A.6: Application of non-standard form to a vector.

It follows from (A.3.26) that after applying the NS-form to a vector we arrive at the representation

$$(T_0 f_0)(x) = \sum_{j=1}^{J} \sum_{k \in \mathbb{F}_{2^{n-j}}} \hat{d}_k^j \psi_{j,k}(x) + \sum_{j=1}^{J} \sum_{k \in \mathbb{F}_{2^{n-j}}} \hat{s}_k^j \varphi_{j,k}(x). \tag{A.3.35}$$

The representation (A.3.35) consists of both averages and differences on all scales which can either be projected into the wavelet basis or reconstructed to space $\mathbf{V}_0$. In order to project (A.3.35) into the wavelet basis we form the representation, see (A.2.12),

$$(T_0 f_0)(x) = \sum_{j=1}^{J} \sum_{k \in \mathbb{F}_{2^{n-j}}} d_k^j \psi_{j,k}(x) + \sum_{k \in \mathbb{F}_{2^{n-J}}} s_k^J \varphi_{J,k}(x), \tag{A.3.36}$$

using the decomposition algorithm described in Appendix A.2 as follows. Given the coefficients $\{\hat{s}^j\}_{j=1}^J$ and $\{\hat{d}^j\}_{j=1}^J$, we decompose $\{\hat{s}^1\}$ into $\{\tilde{s}^2\}$ and $\{\tilde{d}^2\}$ and form the sums $\{s^2\} = \{\hat{s}^2 + \tilde{s}^2\}$ and $\{d^2\} = \{\hat{d}^2 + \tilde{d}^2\}$. Then on each scale $j = 2, 3, \ldots, J-1$, we decompose $\{s^j\} = \{\hat{s}^j + \tilde{s}^j\}$ into $\{\tilde{s}^{j+1}\}$ and $\{\tilde{d}^{j+1}\}$ and form the sums $\{s^{j+1}\} = \{\hat{s}^{j+1} + \tilde{s}^{j+1}\}$ and $\{d^{j+1}\} = \{\hat{d}^{j+1} + \tilde{d}^{j+1}\}$. The sets $\{s^J\}$ and $\{d^j\}_{j=1}^J$ are the coefficients of the wavelet expansion of $(T_0 f_0)(x)$, i.e. the coefficients appearing in (A.3.36). This procedure is illustrated in Figure A.7.

$$\begin{array}{ccccccccc}
\{\hat{s}^0\} & \rightarrow & \{s^1\} = \{\hat{s}^1 + \tilde{s}^1\} & \rightarrow & \{s^2\} = \{\hat{s}^2 + \tilde{s}^2\} & \cdots & \rightarrow & \{s^J\} = \{\hat{s}^J + \tilde{s}^J\} \\
 & \searrow & & \searrow & & & \searrow & \\
 & & \{d^1\} = \{\hat{d}^1 + \tilde{d}^1\} & & \{d^2\} = \{\hat{d}^2 + \tilde{d}^2\} & \cdots & & \{d^J\} = \{\hat{d}^J + \tilde{d}^J\}
\end{array}$$

Figure A.7. Reprojection of the product of the NS-form and a function into a wavelet basis.

An alternative to projecting the representation (A.3.35) into the wavelet basis is to reconstruct (A.3.35) to space $\mathbf{V}_0$, i.e. form the representation (A.2.6)

$$(P_0 f)(x) = \sum_{k \in \mathbb{Z}} s_k^0 \varphi_{0,k}(x) \tag{A.3.37}$$

using the reconstruction algorithm described in Appendix A.2 as follows. Given the coefficients $\{\hat{s}^j\}_{j=1}^J$ and $\{\hat{d}^j\}_{j=1}^J$, we reconstruct $\{\hat{d}^J\}$ and $\{\hat{s}^J\}$ into $\{\tilde{s}^{J-1}\}$ and form the sum $\{s^{J-1}\} = \{\hat{s}^{J-1} + \tilde{s}^{J-1}\}$. Then on each scale $j = J-1, J-2, \ldots, 1$ we reconstruct $\{\hat{s}^j\}$ and $\{\hat{d}^j\}$ into $\{\tilde{s}^{j-1}\}$ and form the sum $\{s^{j-1}\} = \{\hat{s}^{j-1} + \tilde{s}^{j-1}\}$. The final reconstruction (of $\{d^1\}$ and $\{s^1\}$) forms the coefficients $\{s^0\}$ appearing in (A.3.37). This procedure is illustrated in Figure A.8.

$$\begin{array}{ccccccccc} \{\hat{s}^0\} & \leftarrow & \{s^1\} = \{\hat{s}^1 + \tilde{s}^1\} & \cdots & \leftarrow & \{s^{J-1}\} = \{\hat{s}^{J-1} + \tilde{s}^{J-1}\} & \leftarrow & \{s^J\} \\ & \nwarrow & & & \nwarrow & & \nwarrow & \\ & & \{d^1\} = \{\hat{d}^1 + \tilde{d}^1\} & \cdots & & \{d^{J-1}\} = \{\hat{d}^{J-1} + \tilde{d}^{J-1}\} & & \{d^J\} \end{array}$$

Figure A.8. Reconstruction of the product of the NS-form and a function to space $\mathbf{V}_0$.

In our work we are interested in developing adaptive algorithms, i.e. algorithms such that the number of operations performed is proportional to the number of significant coefficients in the wavelet expansion of solutions of partial differential equations. The S-form has 'built-in' adaptivity, i.e. applying the S-form of an operator to the wavelet expansion of a function, (A.2.8), is a matter of multiplying a sparse vector by a sparse matrix. A simple algorithm for this purpose is described in Appendix C. On the other hand, as we have mentioned before, the S-form is not a very efficient representation of, for example, convolution operators.

In the following Sections we address the issue of adaptively multiplying the NS-form and a vector. Since the NS-form of a convolution operator remains a convolution, the A^j, B^j, and Γ^j blocks may be thought of as being represented by short filters. For example, the NS-form of a differential operator in any dimension requires $O(C)$ coefficients as it would for finite difference schemes. We can exploit the efficient representation afforded us by the NS-form and use the vanishing-moment property of the B^j and Γ^j blocks of the NS-form of differential operators and the Hilbert transform to develop an adaptive algorithm. In

Section 2.2.1 we describe two methods for constructing the NS-form representation of operator functions. In Section 2.2.2 we establish the vanishing-moment property which we later use to develop an adaptive algorithm for multiplying operators and functions expanded in a wavelet basis. Finally, in Section 2.2.3 we present an algorithm for adaptively multiplying the NS-form representation of an operator and a function expanded in the wavelet system of coordinates.

A.4 The Non-Standard Form of Differential Operators

In this Section we review the wavelet representation of differential operators ∂_x^p in the non-standard form (NS-form), and specifically consider the construction and properties of the NS-form of the differential operator ∂_x^p. The rows of the NS-form representation of differential operators may be viewed as finite-difference approximations on subspace $\mathbf{V}_0$ of order $2M-1$, where M is the number of vanishing moments of the wavelet $\psi(x)$. This Appendix is a review of material found in [45].

The NS-form representation of the operator ∂_x^p consists of matrices A^j, B^j, Γ^j, for $j = 0, 1, \ldots, J$ and a final 'coarse scale' approximation T^J. The elements of these matrices are denoted $\alpha_{i,l}^j$, $\beta_{i,l}^j$, and $\gamma_{i,l}^j$, for $j = 0, 1, \ldots, J$, and $s_{i,l}^J$, and are computed using the definitions

$$\left.\begin{array}{rcl} \alpha_{k,k'}^j &=& \int_{-\infty}^{\infty} \psi_{j,k}(x) \frac{d^p}{dx^p} \psi_{j,k'}(x) dx \\ \beta_{k,k'}^j &=& \int_{-\infty}^{\infty} \psi_{j,k}(x) \frac{d^p}{dx^p} \varphi_{j,k'}(x) dx \\ \gamma_{k,k'}^j &=& \int_{-\infty}^{\infty} \varphi_{j,k}(x) \frac{d^p}{dx^p} \psi_{j,k'}(x) dx \\ s_{k,k'}^j &=& \int_{-\infty}^{\infty} \varphi_{j,k}(x) \frac{d^p}{dx^p} \varphi_{j,k'}(x) dx. \end{array}\right\} \tag{A.4.38}$$

Using

$$\left.\begin{array}{rcl} \varphi_{j,k}(x) &=& 2^{-j/2} \varphi(2^{-j}x - k) \\ \psi_{j,k}(x) &=& 2^{-j/2} \varphi(2^{-j}x - k) \end{array}\right\} \tag{A.4.39}$$

and changing variables we can rewrite (A.4.38) as

$$\left.\begin{array}{rcl} \alpha^j_{k,k'} = \alpha^j_{k-k'} & = & \int_{-\infty}^{\infty} \psi_{j,k}(x-(k-k'))\frac{d^p}{dx^p}\psi_{j,k'}(x)dx \\ \beta^j_{k,k'} = \beta^j_{k-k'} & = & \int_{-\infty}^{\infty} \psi_{j,k}(x-(k-k'))\frac{d^p}{dx^p}\varphi_{j,k'}(x)dx \\ \gamma^j_{k,k'} = \gamma^j_{k-k'} & = & \int_{-\infty}^{\infty} \varphi_{j,k}(x-(k-k'))\frac{d^p}{dx^p}\psi_{j,k'}(x)dx \\ s^j_{k,k'} = s^j_{k-k'} & = & \int_{-\infty}^{\infty} \varphi_{j,k}(x-(k-k'))\frac{d^p}{dx^p}\varphi_{j,k'}(x)dx. \end{array}\right\} \quad \text{(A.4.40)}$$

We note that $\alpha^j_{k,k'}$, $\beta^j_{k,k'}$, and $\gamma^j_{k,k'}$ depend on the difference $k-k'$ which we will write as $l = k-k'$. The operator ∂^p_x is homogeneous of degree p and it is therefore sufficient to evaluate (A.4.40) on scale $j=0$ and recursively compute the elements α^j_l, β^j_l, and γ^j_l for scales $j = 1, 2, \ldots J$ via

$$\left.\begin{array}{rcl} \alpha^j_l & = & 2^{-pj}\alpha^0_l \\ \beta^j_l & = & 2^{-pj}\beta^0_l \\ \gamma^j_l & = & 2^{-pj}\gamma^0_l \\ s^j_l & = & 2^{-mj}s^0_l \end{array}\right\} \quad \text{(A.4.41)}$$

We note that if we were to use any other finite-difference representation as coefficients on $\mathbf{V}_0$, the coefficients on $\mathbf{V}_j$ would not be related by scaling and would require individual calculations for each j. We can simplify calculating the coefficients α^j_l, β^j_l, and γ^j_l for scales $j = 1, 2, \ldots J$ using the 2-scale difference equations (A.1.4) and (A.1.5). We are led to

$$\left.\begin{array}{rcl} \alpha^j_l & = & 2\sum_{k=0}^{L-1}\sum_{k'=0}^{L-1} g_k g_{k'} s^{j-1}_{2i+k-k'} \\ \beta^j_l & = & 2\sum_{k=0}^{L-1}\sum_{k'=0}^{L-1} g_k h_{k'} s^{j-1}_{2i+k-k'} \\ \gamma^j_l & = & 2\sum_{k=0}^{L-1}\sum_{k'=0}^{L-1} h_k g_{k'} s^{j-1}_{2i+k-k'}. \end{array}\right\} \quad \text{(A.4.42)}$$

Therefore the representation of ∂^p_x is completely determined by s^0_l in (A.4.40) or in other words, by the representation of ∂^p_x projected on the subspace $\mathbf{V}_0$.

To compute the coefficients s^0_l corresponding to the projection of ∂^p_x on $\mathbf{V}_0$ it is sufficient to solve the system of linear algebraic equations

$$s^0_l = 2^p \left[s^0_{2l} + \tfrac{1}{2}\sum_{k=1}^{L/2} a_{2k-1}(s^0_{2l-2k+1} + s^0_{2l+2k-1}) \right], \quad \text{(A.4.43)}$$

and

$$\sum_l l^p \, s_l^0 = (-1)^p p!, \tag{A.4.44}$$

where a_{2k-1} are the autocorrelation coefficients of H defined by

$$a_n = 2 \sum_{i=0}^{L-1-n} h_i \, h_{i+n}, \quad n = 1, \ldots, L-1. \tag{A.4.45}$$

We note that the autocorrelation coefficients a_n with even indices are zero,

$$a_{2k} = 0, \quad k = 1, \ldots, L/2 - 1. \tag{A.4.46}$$

The resulting coefficients s_l^0 corresponding to the projection of the operator ∂_x^p on $\mathbf{V}_0$ may be viewed as a finite-difference approximation of order $2M-1$. The solution of equations (A.4.43) and (A.4.44) has been thoroughly studied and the details can be found in [48].

APPENDIX B

DERIVATION OF QUADRATURE APPROXIMATIONS

In this Appendix we present a detailed description of the steps taken to compute the quadrature approximations (2.1.17)-(2.1.20). These steps consists of a sequence of simple, but tedious, calculations which were performed using *Mathematica* . Section B.1 describes the analytic derivation of approximations of the form (2.1.17) and provides detailed *Mathematica* listings of the programs used for the algebra. Section B.2 discusses approximations of the form (2.1.20) and includes the corresponding *Mathematica* programs.

B.1 Derivation of Approximation – $m = 1$

In the case where $m = 1$, equation (2.1.7) becomes

$$\hat{I}(t) = \mathcal{O}_{\mathcal{L},1} \left(c_{0,0} A_{0,0} + c_{0,1} A_{0,1} + c_{1,0} A_{1,0} + c_{1,1} A_{1,1} \right), \tag{B.1.1}$$

where $A_{i,j} = u_i v_j$ and

$$\mathcal{O}_{\mathcal{L},1} = \left(\mathbf{I} - e^{\Delta t \mathcal{L}} \right) \mathcal{L}^{-1}.$$

The Lagrange based approximation (2.1.13) is given by

$$\overline{I}(t) = \Delta t \left(f_{0,0} A_{0,0} + f_{0,1} A_{0,1} + f_{1,0} A_{1,0} + f_{1,1} A_{1,1} \right), \tag{B.1.2}$$

where

$$f_{i,j} = \int_{t_0}^{t} e^{(t-\tau)\mathcal{L}} L_i(\tau) L_j(\tau) d\tau. \tag{B.1.3}$$

We observe that equation (B.1.1) is an order Δt^2 approximation to $I(x,t)$ and (B.1.2) is an order Δt^3 approximation.

We determine the coefficients $c_{i,j}$ by comparing like coefficients in powers of Δt of the approximations (B.1.1) and (B.1.2) by computing the difference

$$\overline{I}(t) - \hat{I}(t) \quad = \quad \left[\frac{2(A_{0,0} + A_{1,1}) + (A_{1,0} + A_{0,1})}{6} \quad - \right.$$

$$
\begin{aligned}
& (A_{0,0}c_{0,0} + A_{0,1}c_{0,1} + A_{1,0}c_{1,0} + A_{1,1}c_{1,1}) \Big] \Delta t + \\
& \left(\frac{3A_{0,0} + A_{1,0} + A_{0,1} + A_{1,1}}{12} \quad - \right. \\
& \left. \frac{A_{0,0}c_{0,0} + A_{0,1}c_{0,1} + A_{1,0}c_{1,0} + A_{1,1}c_{1,1}}{2} \right) \mathcal{L} \Delta t^2 \\
+ \quad & O(\Delta t^3). \qquad \text{(B.1.4)}
\end{aligned}
$$

In (B.1.4) we have substituted the expressions for the functions $f_{i,j}$ and the Taylor series expansion to order Δt^3 for the exponential function.

In order to simplify this expression, we substitute the order Δt^3 Taylor series expansions for u_0 and v_0 expanded about the point u_1 and v_1, respectively, e.g.

$$u_0 = u_1 - \Delta t u_1' + \frac{\Delta t^2}{2!} u_1'' + O(\Delta t^3),$$

where primes denote differentiation with respect to t. Equation (B.1.4) then becomes

$$
\begin{aligned}
\overline{I}(t) - \hat{I}(t) &= [u_1 v_1 (1 - S_c)] \Delta t \qquad \text{(B.1.5)} \\
&+ \tfrac{1}{2} \left[-u_1' v_1 (1 - 2c_{0,0} - 2c_{0,1}) - v_1' u_1 (1 - 2c_{0,0} - 2c_{1,0}) \right] \Delta t^2 \\
&+ O(\Delta t^3), \qquad \text{(B.1.6)}
\end{aligned}
$$

where $S_c = \sum_{i,j=0}^{m} c_{i,j}$.

Requiring that coefficients of order Δt and Δt^2 be zero implies the following system of equations for the $c_{i,j}$

$$
\begin{aligned}
1 &= c_{0,0} + c_{0,1} + c_{1,0} + c_{1,1} \\
1 &= 2c_{0,0} + 2c_{0,1} \qquad \text{(B.1.7)} \\
1 &= 2c_{0,0} + 2c_{1,0}
\end{aligned}
$$

which has a solution

$$c_{1,1} = c_{0,0}$$

$$\begin{aligned} c_{0,1} &= \frac{1-2c_{0,0}}{2} \\ c_{1,0} &= \frac{1-2c_{0,0}}{2}. \end{aligned} \tag{B.1.8}$$

Substituting these expressions into equation (2.1.7) gives an $O(\Delta t^2)$ approximation to (2.1.16)

$$\hat{I}(t) = \mathcal{O}_{\mathcal{L},1}((\tfrac{1}{2} - c_{0,0})(A_{0,1} + A_{1,0}) + c_{0,0}(A_{0,0} + A_{1,1})), \tag{B.1.9}$$

where $c_{0,0}$ may be chosen arbitrarily. It makes sense to choose $c_{0,0}$ so that the number of terms in the approximation (B.1.9) is minimal. For $c_{0,0} = \frac{1}{2}$ (B.1.9) becomes

$$\hat{I}(t) = \mathcal{O}_{\mathcal{L},1}\tfrac{1}{2}(u_0 v_0 + u_1 v_1) \tag{B.1.10}$$

and for $c_{0,0} = 0$ we have

$$\hat{I}(t) = \mathcal{O}_{\mathcal{L},1}\tfrac{1}{2}(u_0 v_1 + u_1 v_0). \tag{B.1.11}$$

Observe that equation (B.1.10) is analogous to the trapezoidal rule.

B.1.1 *Mathematica* **Programs for** $m = 1$ We begin with the $m = 1$ case by setting the order of the approximation,

```
In[1]:= m = 1
Out[1]= 1
```

We define the n^{th} order Taylor series expansion of e^t about $t = 0$ by

```
In[2]:= n = 10
Out[2]= 10

In[3]:= rexp = {E^(t_) -> Sum[t^j / j!,{j,0,n}]}
           t_
Out[3]= {E   ->
          2    3    4    5     6      7      8        9         10
         t    t    t    t     t      t      t        t         t
1 + t + -- + -- + -- + --- + --- + ---- + ----- + ------ + -------}
         2    6    24   120   720   5040   40320   362880   3628800
```

Finally the n^{th} order Taylor expansions of the functions $u(t)$ and $v(t)$ about the point t_0 are defined by

```
In[4]:= u1[t_,t0_] :=
Normal[Series[u[t1],{t1,t2,n}]] /. {t1 -> t,t2 -> t0}

In[5]:= v1[t_,t0_] :=
Normal[Series[v[t1],{t1,t2,n}]] /. {t1 -> t,t2 -> t0}
```

We note that in what follows $h \equiv \Delta t$, $c[i][j] \equiv c_{i-1,j-1}$, and $f[i][j] = f_{i-1,j-1}$ for $i, j = 0, 1$.

We first construct the approximation $\hat{I}(t)$ given by (2.1.7). In the case where $m = 1$ the operator $\mathcal{O}_{\mathcal{L},m}$ is

```
In[6]:= OLm =  (E^(m L h) - 1)/L
              h L
        -1 + E
Out[6]= ---------
            L
```

Using the Taylor expansions of $u(t)$ and $v(t)$ we compute $\hat{I}(t)$ by

```
In[7]:= Ihat = OLm*Sum[Sum[
c[j][i]*u1[h*(j-1),h]*v1[h*(i-1),h],{i,1,m+1}],{j,1,m+1}]

                h L
Out[7]= ((-1 + E   ) (u[h] v[h] c[2][2] +
                                  2           3  (3)       4  (4)
                                 h  u''[h]   h  u   [h]   h  u   [h]
v[h] c[1][2] (u[h] - h u'[h] + --------- - ---------- + ---------- -
                                    2           6            24
 5  (5)       6  (6)       7  (7)       8  (8)       9  (9)
h  u   [h]   h  u   [h]   h  u   [h]   h  u   [h]   h  u   [h]
---------- + ---------- - ---------- + ---------- - ---------- +
   120          720          5040        40320        362880
 10  (10)
h   u    [h]
------------) + u[h] c[2][1]
  3628800
                  2           3  (3)       4  (4)
                 h  v''[h]   h  v   [h]   h  v   [h]
(v[h] - h v'[h] + --------- - ---------- + ---------- -
                     2           6            24
 5  (5)       6  (6)       7  (7)       8  (8)       9  (9)
h  v   [h]   h  v   [h]   h  v   [h]   h  v   [h]   h  v   [h]
---------- + ---------- - ---------- + ---------- - ---------- +
   120          720          5040        40320        362880
```

```
 10  (10)
h   v    [h]
------------) + c[1][1]
  3628800
                   2           3  (3)       4  (4)
                  h  u''[h]   h  u   [h]   h  u   [h]
(u[h] - h u'[h] + --------- - ---------- + ---------- -
                      2           6            24

 5  (5)       6  (6)       7  (7)       8  (8)       9  (9)
h  u   [h]   h  u   [h]   h  u   [h]   h  u   [h]   h  u   [h]
---------- + ---------- - ---------- + ---------- - ---------- +
   120          720          5040        40320        362880

 10  (10)                        2           3  (3)
h   u    [h]                    h  v''[h]   h  v   [h]
------------) (v[h] - h v'[h] + --------- - ---------- +
  3628800                           2           6

 4  (4)       5  (5)       6  (6)       7  (7)       8  (8)
h  v   [h]   h  v   [h]   h  v   [h]   h  v   [h]   h  v   [h]
---------- - ---------- + ---------- - ---------- + ---------- -
    24          120          720          5040        40320

 9  (9)       10  (10)
h  v   [h]   h   v    [h]
---------- + ------------))) / L
  362880       3628800
```

In order to compute the Lagrange based approximation $\overline{I}(t)$, (B.1.2), we first define the Lagrange polynomials via (2.1.10) using

```
In[8]:= P[t_,i_,m_] :=
Product[(t-k),{k,0,i-1}]*Product[(t-k),{k,i+1,m}]

In[9]:= L[t_,i_,m_] := P[t,i,m]/(P[t,i,m] /. t-> i)
```

The table `ll` defined by

```
In[10]:= ll = Table[l[i] = L[t,i,m],{i,0,m}]
```

consists of Lagrange polynomials. In the case where $m = 1$ the table `ll` consists of the well-known linear interpolants

```
Out[10]= {1 - t, t}
```

The functions $f_{i,j} \equiv f[i+1][j+1], i, j = 0, 1$ defined by (2.1.14) are computed using

```
In[11]:= For[j=1,j<=m+1,++j, For[i=1,i<=m+1,++i, f[i][j] =
Expand[E^(m*L*h)*Integrate[E^(-L*t*h)*ll[[i]]*ll[[j]],{t,0,m}]]]]
```

For $m = 1$ we can enumerate the four $f_{i,j}$ as

```
In[12]:= f[1][1]
                      h L      h L    h L
          -2     2 E      2 E      E
Out[12]= ----- + ------ - ------ + ----
          3  3    3  3     2  2    h L
         h  L    h  L     h  L
```

```
In[13]:= f[1][2]
                      h L            h L
           2     2 E        1     E
Out[13]= ----- - ------ + ----- + -----
          3  3    3  3     2  2    2  2
         h  L    h  L     h  L    h  L
```

```
In[14]:= f[2][1]
                      h L            h L
           2     2 E        1     E
Out[14]= ----- - ------ + ----- + -----
          3  3    3  3     2  2    2  2
         h  L    h  L     h  L    h  L
```

```
In[15]:= f[2][2]
                      h L
          -2     2 E        2      1
Out[15]= ----- + ------ - ----- - ---
          3  3    3  3     2  2   h L
         h  L    h  L     h  L
```

The Lagrange based approximation (2.1.13) is defined by

```
In[16]:= Ibar = h*Sum[Sum[
f[j][i]*u1[h*(j-1),h]*v1[h*(i-1),h],{i,1,m+1}],{j,1,m+1}];
```

The difference $\overline{I}(t) - \hat{I}(t)$ is computed via

```
In[17]:= dif = Expand[(Ibar - Ihat) /. rexp];
```

where we have explicitly used the Taylor series expansion of the exponential function. The coefficients of powers of Δt, `coefhp[j]`, are computed by

```
In[18]:= For[j=1,j<=5,++j,coefhp[j] = Coefficient[dif,h,j-1];]
```

For terms of order 1 we find

```
In[19]:= Simplify[coefhp[1]]
Out[19]= 0
```

The coefficient of order Δt is found to be

```
In[20]:= Simplify[coefhp[2]]

Out[20]= u[h] v[h] (1 - c[1][1] - c[1][2] - c[2][1] - c[2][2])
```

which leads to the first constraint of the set (B.1.7), i.e.

$$\sum_{i,j=0}^{m=1} c_{i,j} = 1.$$

This condition is defined as a rule via, e.g.,

```
In[22]:= rsum = {c[1][1] -> 1 - c[1][2] - c[2][1] - c[2][2]};
```

Using the sum rule, `rsum`, we find at order Δt^2 that

```
In[23]:= Simplify[coefhp[3] /.rsum]

          v[h] u'[h]
Out[23]= ---------- - v[h] c[2][1] u'[h] - v[h] c[2][2] u'[h] +
               2
  u[h] v'[h]
> ---------- -  u[h] c[1][2] v'[h] - u[h] c[2][2] v'[h]
       2
```

Observing that this expression involves only $u(h)v'(h)$ and $u'(h)v(h)$ terms, we first extract the coefficient of the $u(h)v'(h)$ term

```
In[24]:= Simplify[
Coefficient[Coefficient[coefhp[3] /. rsum, u[h],1],v'[h],1]]

          1
Out[24]= - - c[1][2] - c[2][2]
          2
```

Then we extract the coefficient of the $u'(h)v(h)$ term

```
In[25]:= Simplify[
Coefficient[Coefficient[coefhp[3] /. rsum, u'[h],1],v[h],1]]

          1
Out[25]= - - c[2][1] - c[2][2]
          2
```

Outputs Out[20], Out[24], and Out[25] are identified as the constraints on the coefficients $c_{i,j}$, namely equations (B.1.7)

$$\begin{aligned} 1 &= c_{0,0} + c_{0,1} + c_{1,0} + c_{1,1} \\ 1 &= 2c_{0,0} + 2c_{0,1} \\ 1 &= 2c_{0,0} + 2c_{1,0} \end{aligned}$$

We solve this system of equations using the *Mathematica* `Reduce` function

```
In[26]:= Reduce[{c[1][1] + c[1][2] + c[2][1] + c[2][2]==1,
2 c[1][1] + 2 c[1][2] == 1,
2 c[1][1] + 2 c[2][1] == 1}]
```

which yields a solution to this system of constraints on the coefficients $c_{i,j}$,

```
                                                  1 - 2 c[1][1]
Out[26]= c[2][2] == c[1][1] && c[1][2] == ------------- &&
                                                      2

                     1 - 2 c[1][1]
>       c[2][1] == -------------
                           2
```

Output Out[26] is identified as the result (B.1.8),

$$\begin{aligned} c_{1,1} &= c_{0,0} \\ c_{0,1} &= \frac{1 - 2c_{0,0}}{2} \\ c_{1,0} &= \frac{1 - 2c_{0,0}}{2}. \end{aligned}$$

B.2 Derivation of Approximation – $m = 2$

In this next example we look for an approximation to (2.1.16) which is at least order Δt^3. We find that $\mathcal{O}_{\mathcal{L},m}$ defined by (2.1.19) is insufficient

to guarantee this order. Using our procedure, we identify the source of this inconsistency and illustrate a solution by suitably modifying (2.1.19). As in the previous example, we determine a set of conditions on the coefficients $c_{i,j}$ by comparing like coefficients in powers of Δt. Repeating the same steps as in the previous example, namely computing the difference between $\hat{I}$ and $\overline{I}$, substituting the definition of $f_{i,j}$, substituting the Taylor series expansions of the exponential function, and substituting the expansions of the u_0, u_2 and v_0, v_2 about u_1 and v_1 gives

$$\begin{aligned}
\overline{I}(t) - \hat{I}(t) &= 2u_1v_1(1-S_c)\Delta t \\
&+ 2[u_1v_1\mathcal{L}(1-S_c) + \\
&\quad u_1'v_1(c_{0,0}+c_{0,1}+c_{0,2}-c_{2,0}-c_{2,1}-c_{2,2}) + \\
&\quad v_1'u_1(c_{0,0}+c_{1,0}+c_{2,0}-c_{0,2}-c_{1,2}-c_{2,2})]\Delta t^2 \\
&+ 2[\tfrac{2}{3}u_1v_1\mathcal{L}^2(1-S_c) \\
&\quad \tfrac{1}{3}\mathcal{L}(u_1'v_1(-1+3(c_{0,0}+c_{0,1}+c_{0,2})-3(c_{2,0}+c_{2,2}+c_{2,2})) \\
&\qquad +v_1'u_1(-1+3(c_{0,0}+c_{1,0}+c_{2,0})-3(c_{0,2}+c_{1,2}+c_{2,2}))) \\
&\quad +\tfrac{1}{3}u_1'v_1'(1-3(c_{0,0}-c_{0,2}-c_{2,0}+c_{2,2})) \\
&\quad +\tfrac{1}{6}(u_1''v_1(1-3(c_{0,0}+c_{0,1}+c_{0,2}+c_{2,0}+c_{2,1}+c_{2,2})) \\
&\qquad +u_1v_1''(1-3(c_{0,0}+c_{1,0}+c_{2,0}+c_{0,2}+c_{1,2}+c_{2,2})))]\Delta t^3 \\
&+ O(\Delta t^4). \qquad \text{(B.2.12)}
\end{aligned}$$

We now find relationships between the coefficients $c_{i,j}$ so that coefficients of Δt, Δt^2, and Δt^3 are zero. For terms of order Δt, we find the same condition as in the previous example, namely the sum of the coefficients $c_{i,j}$ must be one,

$$S_c = \sum_{i,j=0}^{m} c_{i,j} = 1. \qquad \text{(B.2.13)}$$

Using (B.2.13) we find the following two conditions for terms of order Δt^2,

$$1 - c_{1,0} - c_{1,1} - c_{1,2} = 2(c_{2,0} + c_{2,1} + c_{2,2}) \qquad \text{(B.2.14)}$$

$$1 - c_{0,1} - c_{0,2} - c_{1,1} = 2(c_{1,2} + c_{2,1} + c_{2,2}). \tag{B.2.15}$$

Requiring the coefficients $c_{i,j}$ to satisfy relations (B.2.13),(B.2.14) and (B.2.15), the difference (B.2.12) is now zero to order Δt^3,

$$\begin{aligned}\overline{I}(t) - \hat{I}(t) &= \Big[-\tfrac{2}{3}\mathcal{L}(u_1'v_1 + v_1'u_1) \\ &\quad +\tfrac{2}{3}u_1'v_1'(1 - 3(c_{0,0} - c_{0,2} - c_{2,0} + c_{2,2})) \\ &\quad +\tfrac{1}{3}(u_1''v_1(1 - 6(c_{2,0} + c_{2,1} + c_{2,2})) \\ &\quad\quad +u_1v_1''(1 - 6(c_{0,2} + c_{1,2} + c_{2,2}))\Big]\Delta t^3 \\ &+ O(\Delta t^4). \end{aligned} \tag{B.2.16}$$

We see that requiring the lower order terms to be zero in equation (B.2.12) has introduced a term which is independent of the coefficients $c_{i,j}$ and is of order Δt^3, i.e.

$$-\tfrac{2}{3}\mathcal{L}(u_1'v_1 + v_1'u_1)\Delta t^3. \tag{B.2.17}$$

Let us first remove the dependence on the coefficients $c_{i,j}$ in (B.2.16) by requiring the following conditions

$$1 = 3(c_{0,0} - c_{0,2} - c_{2,0} + c_{2,2}) \tag{B.2.18}$$

$$1 = 6(c_{2,0} + c_{2,1} + c_{2,2}) \tag{B.2.19}$$

$$1 = 6(c_{0,2} + c_{1,2} + c_{2,2}), \tag{B.2.20}$$

in addition to (B.2.13) through (B.2.15). Relations (B.2.13) through (B.2.15) and (B.2.18) through (B.2.20) define a set of six equations in nine unknowns which has a solution given by

$$\left.\begin{aligned} c_{0,0} &= \tfrac{1}{3}(2 - 3c_{1,2} - 3c_{2,1} - 9c_{2,2}) \\ c_{1,0} &= \tfrac{1}{3}(-2 + 3c_{1,2} + 6c_{2,1} + 12c_{2,2}) \\ c_{0,1} &= \tfrac{1}{3}(-2 + 6c_{1,2} + 3c_{2,1} + 12c_{2,2}) \\ c_{0,2} &= \tfrac{1}{6}(1 - 6c_{1,2} - 6c_{2,2}) \\ c_{2,0} &= \tfrac{1}{6}(1 - 6c_{2,1} - 6c_{2,2}) \\ c_{1,1} &= \tfrac{2}{3}(2 - 3c_{1,2} - 3c_{2,1} - 6c_{2,2}) \end{aligned}\right\} \tag{B.2.21}$$

To specify an approximation we fix four coefficients and use (B.2.21) to define the remaining coefficients.

The difference (B.2.16) is then

$$\overline{I}(t) - \hat{I}(t) = -\tfrac{2}{3}\mathcal{L}(u_1'v_1 + v_1'u_1)\Delta t^3 + O(\Delta t^4), \tag{B.2.22}$$

In order to account for the Δt^3 term, we modify the form of the approximation (2.1.7) by observing that

$$u_1'v_1 + v_1'u_1 = (uv)'|_{t_1}. \tag{B.2.23}$$

We may approximate the derivative $(uv)'|_{t_1}$ in a number of ways. The simplest is to use the forward and backward differences of the product uv,

$$\left.\begin{aligned} (uv)|_{t_2} &= (uv)|_{t_1} + \Delta t(uv)'|_{t_1} + \tfrac{\Delta t^2}{2}(uv)''|_{t_1} + O(\Delta t^3) \\ (uv)|_{t_0} &= (uv)|_{t_1} - \Delta t(uv)'|_{t_1} + \tfrac{\Delta t^2}{2}(uv)''|_{t_1} + O(\Delta t^3) \end{aligned}\right\} \tag{B.2.24}$$

in order to form an order Δt^2 approximation to the derivative of the product,

$$(uv)'|_{t_1} = \frac{u_2v_2 - u_0v_0}{2\Delta t} + O(\Delta t^2). \tag{B.2.25}$$

Using (B.2.25) in the right hand side of (B.2.22) yields

$$-\tfrac{2}{3}\mathcal{L}(u_1'v_1 + v_1'u_1)\Delta t^3 + O(\Delta t^4) = -\tfrac{1}{3}\mathcal{L}(u_2v_2 - u_0v_0)\Delta t^2 + O(\Delta t^4). \tag{B.2.26}$$

Therefore we modify (2.1.7) to include a term of order Δt^2, i.e. we define

$$\hat{I}_{new_1}(t) = \hat{I}(t) + \tfrac{1}{3}\mathcal{L}(u_2v_2 - u_0v_0)\Delta t^2, \tag{B.2.27}$$

which leads to

$$\overline{I}(t) - \hat{I}(t) = O(\Delta t^4). \tag{B.2.28}$$

Note that (B.2.25) is one way to approximate the derivative $(uv)'|_{t_1}$. Another approximation is arrived at by computing products of the Taylor expansions of the individual terms in $(uv)'|_{t_1}$ (B.2.23). For example substituting

$$\left.\begin{aligned} u'_{t_1} &= \tfrac{u_2 - u_0}{2\Delta t} + O(\Delta t^2) \\ v'_{t_1} &= \tfrac{v_2 - v_0}{2\Delta t} + O(\Delta t^2) \end{aligned}\right\} \tag{B.2.29}$$

in equation (B.2.23) yields,

$$(uv)'|_{t_1} = \frac{1}{2\Delta t}(u_2v_1 - u_0v_1 + u_1v_2 - u_1v_0) + O(\Delta t^2). \tag{B.2.30}$$

Using (B.2.30) in the right hand side of (B.2.22) yields

$$-\tfrac{2}{3}\mathcal{L}(u_1'v_1 + v_1'u_1)\Delta t^3 + O(\Delta t^4) = -\tfrac{1}{3}\mathcal{L}(u_2v_1 - u_0v_1 + u_1v_2 - u_1v_0)\Delta t^2 + O(\Delta t^4). \tag{B.2.31}$$

Therefore we modify (2.1.7) to include a term of order Δt^2, i.e. we define

$$\hat{I}_{new_2}(t) = \hat{I}(t) + \tfrac{1}{3}\mathcal{L}(u_2v_1 - u_0v_1 + u_1v_2 - u_1v_0)\Delta t^2, \tag{B.2.32}$$

which also leads to

$$\overline{I}(t) - \hat{I}(t) = O(\Delta t^4). \tag{B.2.33}$$

It makes sense to fix four coefficients in (B.2.21) so that the number of terms in the approximation (2.1.7) is minimal. For example, setting $c_{0,0} = \frac{1}{6}, c_{1,1} = \frac{2}{3}, c_{2,2} = \frac{1}{6}$ and $c_{2,0} = 0$ we arrive at an approximation of the form

$$\begin{aligned} I(t) &= \mathcal{O}_{\mathcal{L},2}\left(\tfrac{1}{6}u_0v_0 + \tfrac{2}{3}u_1v_1 + \tfrac{1}{6}u_2v_2\right) \\ &\quad + \tfrac{1}{3}\mathcal{L}\left(u_2v_2 - u_0v_0\right)\Delta t^2 + O(\Delta t^3), \end{aligned} \tag{B.2.34}$$

which can be recognized as a three-point generalization of the Simpson's rule. Note that in (B.2.34) we have used (B.2.27).

To formulate an alternative to (B.2.34), which is not obtainable by standard quadrature approaches, we may form a linear combination of two 'asymmetric' approximations. One such approximation which is arrived at by fixing $c_{0,0} = c_{1,1} = 0 = c_{2,2} = 0$ and $c_{2,0} = 0$, which yields an approximation of the form

$$\begin{aligned} I_1(t) &= \mathcal{O}_{\mathcal{L},2}\left(\tfrac{1}{2}(u_0v_1 + u_1v_2) + \tfrac{1}{6}(u_1v_0 + u_2v_1) - \tfrac{1}{3}u_0v_2\right) \\ &\quad + \tfrac{1}{3}\mathcal{L}\left(u_2v_2 - u_0v_0\right)\Delta t^2 + O(\Delta t^3). \end{aligned} \tag{B.2.35}$$

Setting $c_{0,0} = c_{1,1} = 0 = c_{2,2} = 0$ and $c_{0,2} = 0$, yields

$$\begin{aligned} I_2(t) &= \mathcal{O}_{\mathcal{L},2}\left(\tfrac{1}{6}(u_0v_1 + u_1v_2) + \tfrac{1}{2}(u_1v_0 + u_2v_1) - \tfrac{1}{3}u_2v_0\right) \\ &\quad + \tfrac{1}{3}\mathcal{L}\left(u_2v_2 - u_0v_0\right)\Delta t^2 + O(\Delta t^3). \end{aligned} \tag{B.2.36}$$

Then an order Δt^3 approximation to $I(x,t)$ is given by a linear combination of (B.2.35) and (B.2.36),

$$\begin{aligned} I(t) &= \tfrac{1}{2}(I_1(x,t) + I_2(x,t)) \\ &= \tfrac{1}{2}\mathcal{O}_{\mathcal{L},2}\left(\tfrac{2}{3}(u_0v_1 + u_1v_0 + u_1v_2 + u_2v_1) - \tfrac{1}{3}(u_0v_2 + u_2v_0)\right) \\ &\quad + \tfrac{1}{3}\mathcal{L}\left(u_2v_2 - u_0v_0\right)\Delta t^2 + O(\Delta t^3). \end{aligned} \tag{B.2.37}$$

Again we note that in (B.2.35), (B.2.36), and (B.2.37) we have used (B.2.27).

If we use (B.2.30) to approximate the derivative $(uv)'|_{t_1}$ for the case where $c_{0,0} = c_{1,1} = 0 = c_{2,2} = 0$ and $c_{0,2} = 0$, we arrive at

$$\begin{aligned} I(t) &= \tfrac{1}{2}\mathcal{O}_{\mathcal{L},2}(\tfrac{2}{3}(u_0v_1 + u_1v_0 + u_1v_2 + u_2v_1) \\ &\quad -\tfrac{1}{3}(u_0v_2 + u_2v_0)) + \tfrac{1}{3}\mathcal{L}(u_2v_1 - u_0v_1 + u_1v_2 - u_1v_0)\Delta t^2 \\ &\quad + O(\Delta t^4). \end{aligned} \tag{B.2.38}$$

We may combine terms and considerably simplify this expression by defining

$$\mathcal{O}^{\pm}_{\mathcal{L},m} = \mathcal{O}_{\mathcal{L},m} \pm \mathcal{L}\Delta t^2. \tag{B.2.39}$$

Then equation (B.2.38) becomes

$$I(t) = \tfrac{1}{3}\mathcal{O}^{+}_{\mathcal{L},2}(u_0v_1 + u_1v_0) + \tfrac{1}{3}\mathcal{O}^{-}_{\mathcal{L},2}(u_1v_2 + u_2v_1) - \tfrac{1}{6}\mathcal{O}_{\mathcal{L},2}(u_0v_2 + u_2v_0) + O(\Delta t^3). \tag{B.2.40}$$

B.2.1 *Mathematica* **Programs for** $m = 2$ We now detail the *Mathematica* calculations for the case where $m = 2$. Due to the lengthier outputs from *Mathematica* for this example, we will suppress all but the most illustrative output. Again, we begin with the definitions of the order of approximation, $m = 2$, the order of the Taylor series expansions, $n = 10$, and the

definitions of the series expansions of the exponential function and unknown functions, $u(t)$ and $v(t)$.

```
In[1]:= m = 2;
In[2]:= n = 10;
In[3]:= rexp = {E^(t_) -> Sum[t^j / j!,{j,0,n}]};
In[4]:= u1[t_,t0_] :=
Normal[Series[u[t1],{t1,t2,n}]] /. {t1 -> t,t2 -> t0};
In[5]:= v1[t_,t0_] :=
Normal[Series[v[t1],{t1,t2,n}]] /. {t1 -> t,t2 -> t0};
```

The quadrature approximation $\hat{I}(t)$, equation (2.1.7), is defined by

```
In[6]:= OLm =  (E^(m L h) - 1)/L;
In[7]:= Ihat = OLm*Sum[Sum[
c[j][i]*u1[h*(j-1),h]*v1[h*(i-1),h],{i,1,m+1}],{j,1,m+1}];
```

The approximation $\overline{I}(t)$, equation (2.1.13) is defined in terms of the Lagrange polynomials

```
In[8]:= P[t_,i_,m_] :=
Product[(t-k),{k,0,i-1}]*Product[(t-k),{k,i+1,m}];

In[9]:= L[t_,i_,m_] := P[t,i,m]/(P[t,i,m] /. t-> i);
```

which are placed in the lookup table `ll`,

```
In[10]:= ll = Table[l[i] = L[t,i,m],{i,0,m}];
```

The functions $f_{i,j} \equiv f[i+1][j+1], i, j = 0, 1, 2$, defined by (2.1.14), are computed using

```
In[11]:= For[j=1,j<=m+1,++j, For[i=1,i<=m+1,++i, f[i][j] =
Expand[E^(m*L*h)*Integrate[E^(-L*t*h)*ll[[i]]*ll[[j]],{t,0,m}]]]];
```

The approximation $\overline{I}(t)$, equation (2.1.13) is given by

```
In[12]:= Ibar = h*Sum[Sum[
f[j][i]*u1[h*(j-1),h]*v1[h*(i-1),h],{i,1,m+1}],{j,1,m+1}];
```

The difference $\overline{I}(t) - \hat{I}(t)$ is computed via

```
In[13]:= dif = Expand[(Ibar - Ihat) /. rexp];
```

where we have explicitly used the Taylor series expansion of the exponential function. The coefficients in powers of Δt, `coefhp[j]`, are computed by

```
In[14]:= For[j=1,j<=5,++j,coefhp[j] = Coefficient[dif,h,j-1];]
```

For terms of order 1 we find

```
In[15]:= Simplify[coefhp[1]]
Out[15]= 0
```

The coefficient of order Δt is found to be

```
In[16]:= Simplify[coefhp[2]]

Out[16]= 2 u[h] v[h] (1 - c[1][1] - c[1][2] - c[1][3] - c[2][1] -
>      c[2][2] - c[2][3] - c[3][1] - c[3][2] - c[3][3])
```

which is the same condition as in the $m = 1$ case, namely

$$\sum_{i,j=0}^{m=2} c_{i,j} = 1.$$

In order to simplify further calculations, let us recompute the coefficients of powers of Δt subject to the rule that the sum of the coefficients is 1. We first define the rule via, e.g.,

```
In[17]:= rsum = {c[1][1] ->  1 - c[1][2] - c[1][3] - c[2][1] -
c[2][2] - c[2][3] - c[3][1] - c[3][2] - c[3][3]}
```

Then we define a new variable `coefhp2[j]` via

```
In[18]:= For[j=1,j<=5,++j,coefhp2[j] = Simplify[coefhp[j] /. rsum];]
```

We note that the order 1 coefficient remains zero,

```
In[19]:= Simplify[coefhp2[1]]
Out[19]= 0
```

However, due to the sum rule, `rsum`, the order Δt coefficient is now zero

```
In[20]:= Simplify[coefhp2[2]]
Out[20]= 0
```

Using the variable `coefhp2[j]`, we continue with terms of order Δt^2,

```
In[21]:= Simplify[coefhp2[3]]
```

```
Out[21]= 2 (v[h] u'[h] - v[h] c[2][1] u'[h] - v[h] c[2][2] u'[h] -
v[h] c[2][3] u'[h] - 2 v[h] c[3][1] u'[h] - 2 v[h] c[3][2] u'[h] -
2 v[h] c[3][3] u'[h] + u[h] v'[h] - u[h] c[1][2] v'[h] -
2 u[h] c[1][3] v'[h] - u[h] c[2][2] v'[h] - 2 u[h] c[2][3] v'[h] -
u[h] c[3][2] v'[h] - 2 u[h] c[3][3] v'[h])
```

Observing that only $u(h)v'(h)$ and $u'(h)v(h)$ terms are present, we collect coefficients of these terms via

```
In[22]:= Simplify[
Coefficient[Coefficient[coefhp2[3],u'[h],1],v[h],1] ]

Out[22]= 2 (1 - c[2][1] - c[2][2] - c[2][3] - 2 c[3][1] -
2 c[3][2] - 2 c[3][3])

In[23]:= Simplify[
Coefficient[Coefficient[coefhp2[3],u[h],1],v'[h],1] ]

Out[23]= 2 (1 - c[1][2] - 2 c[1][3] - c[2][2] - 2 c[2][3] -
c[3][2] - 2 c[3][3])
```

Outputs Out[22] and Out[23] identify two additional constraints on the coefficients, namely those of equations (B.2.14) and (B.2.15),

$$\begin{aligned} 1 - c_{1,0} - c_{1,1} - c_{1,2} &= 2(c_{2,0} + c_{2,1} + c_{2,2}) \\ 1 - c_{0,1} - c_{0,2} - c_{1,1} &= 2(c_{1,2} + c_{2,1} + c_{2,2}) \end{aligned}$$

We define two new rules based on these conditions by

```
In[24]:= r21 = {c[2][1] -> 1 - c[2][2] - c[2][3] - 2 c[3][1] -
2 c[3][2] - 2 c[3][3]};

In[25]:= r12 = {c[1][2] -> 1 - 2 c[1][3] - c[2][2] - 2 c[2][3] -
c[3][2] - 2 c[3][3]};
```

Once again, we simplify the coefficients `coefhp2[j]` subject to these two new rules via

```
In[26]:= For[j=1,j<=5,++j,
coefhp3[j] = Simplify[coefhp2[j] /. Flatten[Join[r21,r22]]];]
```

We now see that coefficients through order Δt^2 are zero

```
In[27]:= coefhp3[1]          (* At order 1 *)
Out[27]= 0

In[28]:= coefhp3[2]          (* At order h *)
Out[28]= 0

In[29]:= coefhp3[3]          (* At order h^2 *)
Out[29]= 0
```

For terms of order Δt^3 we find

```
In[30]:= coefhp3[4]

Out[30]= (-2 L v[h] u'[h] - 2 L u[h] v'[h] + 8 u'[h] v'[h] -
6 c[2][2] u'[h] v'[h] - 12 c[2][3] u'[h] v'[h] -
12 c[3][2] u'[h] v'[h] - 24 c[3][3] u'[h] v'[h] + v[h] u''[h] -
6 v[h] c[3][1] u''[h] - 6 v[h] c[3][2] u''[h] -
6 v[h] c[3][3] u''[h] + u[h] v''[h] - 6 u[h] c[1][3] v''[h] -
6 u[h] c[2][3] v''[h] - 6 u[h] c[3][3] v''[h]) / 3
```

We extract the coefficients of combinations of the functions u, v and their derivatives using

```
In[31]:= Coefficient[Coefficient[coefhp3[4],u[h],1],v'[h],1]
         -2 L
Out[31]= ----
          3

In[32]:= Coefficient[Coefficient[coefhp3[4],u'[h],1],v[h],1]
         -2 L
Out[32]= ----
          3

In[33]:= Coefficient[Coefficient[coefhp3[4],u'[h],1],v'[h],1]
         8
Out[33]= - - 2 c[2][2] - 4 c[2][3] - 4 c[3][2] - 8 c[3][3]
         3

In[34]:= Coefficient[Coefficient[coefhp3[4],u[h],1],v''[h],1]
         1
Out[34]= - - 2 c[1][3] - 2 c[2][3] - 2 c[3][3]
         3

In[35]:= Coefficient[Coefficient[coefhp3[4],u''[h],1],v[h],1]
         1
Out[35]= - - 2 c[3][1] - 2 c[3][2] - 2 c[3][3]
         3
```

Outputs Out[31] and Out[32] contribute to the order Δt^3 term which is independent of the coefficients $c_{i.j}$, see (B.2.17). Outputs Out[33], Out[34], and Out[35] identify three additional conditions on the coefficients.

We now have a set of six equations in nine unknowns defined by the sum rule, line In[17], and the conditions appearing in outputs Out[22], Out[23], Out[33], Out[34], and Out[35]. These equations may be written in terms of the following variables,

```
In[36]:= rel1 = c[1][1] + c[1][2] + c[1][3] + c[2][1] +
c[2][2] + c[2][3] + c[3][1] + c[3][2] + c[3][3];

In[37]:= rel2 = 2 (1 - c[2][1] - c[2][2] - c[2][3] -
2 c[3][1] - 2 c[3][2] - 2 c[3][3]);

In[38]:= rel3 = 2 (1 - c[1][2] - 2 c[1][3] - c[2][2] -
2 c[2][3] - c[3][2] - 2 c[3][3]);

In[39]:= rel4 = 8/3 - 2 c[2][2] - 4 c[2][3] - 4 c[3][2]
- 8 c[3][3];

In[40]:= rel5 = 1/3 - 2 c[1][3] - 2 c[2][3] - 2 c[3][3];

In[41]:= rel6 = 1/3- 2 c[3][1] - 2 c[3][2] - 2 c[3][3];
```

The *Mathematica* `Reduce` function is now used to solve these equations,

```
In[42]:= Reduce[{rel1==1,rel2==0,rel3==0,rel4==0,rel5==0,rel6==0}]

                     2 - 3 c[2][3] - 3 c[3][2] - 9 c[3][3]
Out[42]= c[1][1] == --------------------------------------- &&
                                    3

                  -2 + 3 c[2][3] + 6 c[3][2] + 12 c[3][3]
>     c[2][1] == ---------------------------------------- &&
                                  3

                  -2 + 6 c[2][3] + 3 c[3][2] + 12 c[3][3]
>     c[1][2] == ---------------------------------------- &&
                                  3

                  1 - 6 c[2][3] - 6 c[3][3]
>     c[1][3] == ------------------------- &&
                              6
```

```
                  1 - 6 c[3][2] - 6 c[3][3]
>     c[3][1] == ------------------------- &&
                             6

                  2 (2 - 3 c[2][3] - 3 c[3][2] - 6 c[3][3])
>     c[2][2] == -----------------------------------------
                                    3
```

which are the solutions appearing in equation (B.2.21).

APPENDIX C

PSEUDOCODE LISTINGS

In this Appendix we provide pseudocode which describes the adaptive algorithms discussed in Chapter 2. Appendix C.1 contains pseudocode which describes the multiplication of the NS-form representation of an operator and a function expanded in a wavelet basis. Appendix C.2 contains pseudocode which describes the pointwise square of a function expanded in a wavelet basis. Appendix C.3 discusses the sparse data structures used to program our algorithms and provides, as an illustration, a relatively simple program for computing the pointwise product of two sparse vectors. In this Appendix we assume that parameters which describe the multiresolution analysis and numerical experiment, e.g. J, ϵ, n, the quadrature mirror filters, etc., have been specified. The format of our pseudocode closely follows the pseudocode described in [56].

C.1 Pseudocode for Multiplying Operators and Functions

The first algorithm we describe is for multiplying the NS-form representation of an operator and a function expanded in a wavelet basis. The processing of this algorithm consists of evaluating equations (2.2.39) and (2.2.40), namely

$$\hat{d}^j = A^j d^j + B^j \tilde{s}^j \tag{C.1.1}$$

$$\hat{s}^j = \Gamma^j d^j, \tag{C.1.2}$$

for $j = 1, 2, \ldots, J-1$, and

$$\hat{d}^J = A^J d^J + B^J \tilde{s}^J \tag{C.1.3}$$

$$\hat{s}^J = \Gamma^J d^J + T^J \tilde{s}^J, \tag{C.1.4}$$

for $j = J$. The algorithm uses as input the masked averages $\{\tilde{s}^j\}$ and the differences $\{d^j\}$, for each scale $j = 1, 2, \ldots, J$. Additionally we use the 'filters' corresponding to the A^j, B^j, and Γ^j blocks for each scale $j = 1, 2, \ldots, J$ and the final scale average T^J. The coefficients $\{\hat{s}^j\}$ and $\{\hat{d}^j\}$ are then reprojected into the wavelet basis using the successive reconstruction/summation procedure illustrated by Figure A.8.

The algorithm for applying the NS-form of an operator to a function expanded in a wavelet basis may be described by the pseudocode shown in Figure C.1. The 'auxiliary functions' listed in the pseudocode shown in Figure C.1 are described as follows. The function `Convolve(x,f,lf)` computes the periodized convolution of the vector `x` with the filter `f` of length `lf`. The function `Reconstruct(x(j),y(j),m,z(j-m))` uses the averages `x` and differences `y` on a given subspace j to reconstruct the averages `z` on subspace $j - $ `m`. The function `Cutoff(x,epsilon)` removes all elements from vector `x` which have absolute value less than `epsilon`. The function `DecomposeAndSum(x,a,d)` successively projects the vector `x` into the multiresolution analysis calculating the averages `a` and differences `d`. On each scale the newly computed averages and differences are added to those already present.

C.2 Pseudocode for Computing the Pointwise Square of a Function

The algorithm for computing the pointwise square of a function evaluates equation (2.3.17),

$$(P_0 u)^2 = (P_J u)^2 + \sum_{j=j_f}^{J} 2(P_j u)(Q_j u) + (Q_j u)^2. \tag{C.2.5}$$

The algorithm uses as input the masked averages $\{\tilde{s}^j\}$ and the differences $\{d^j\}$ for each scale $j = 1, 2, \ldots, J$. The coefficients $\{(\hat{s}^j)^2\}$ which are present on the left hand side of (C.2.5) are then successively projected into the wavelet basis and added as illustrated by Figure 2.3.1.

The algorithm for calculating the pointwise square may be described

```
Let danswer(J) = Convolve{d(J),alpha(J),lalpha(J)} +
                 Convolve{sbar(J),beta(J),lbeta(J)}

Let sanswer(J) = Convolve{d(J),gamma(J),lgamma(J)} +
                 Convolve{sbar(J),tfinal(J),lfinal(J)}

For j = J - 1 To jfirst Step -1

     Let danswer(j) = Convolve{d(j),alpha(j),lalpha(j)} +
                      Convolve{sbar(j),beta(j),lbeta(j)}

     Let stilde(j) = Convolve{d(j),gamma(j),lgamma(j)} +
                     Convolve{sbar(j),tfinal(j),lfinal(j)}

     Reconstruct(stilde(j+1),danswer(j+1),1,sanswer(j))

     Let sanswer(j) = sanswer(j) + stilde(j)

     Cutoff(sanswer(j),epsilon)
     Cutoff(danswer(j),epsilon)

Next j

DecomposeAndSum(sanswer,sanswer,danswer)
```

Figure C.1. Pseudocode for the multiplication of the NS-form representation of an operator and a function expanded in a wavelet basis.

by the pseudocode shown in Figure C.2. The 'auxiliary functions' listed in the pseudocode shown in Figure C.2 are described as follows. The function `SparseVectorProduct(x,y,z)` computes the pointwise product of two sparse vectors `x` and `y` and places the result in another sparse vector `z`. The function `SparseVectorScale(x,c)` scales each element of the sparse vector `x` by the amount `c`. The function `SparseVectorSum(x,y,z)` computes the pointwise sum of two sparse vectors `x` and `y` and places the result in another sparse vector `z`.

C.3 Sparse Data Structures

A dense vector of length N may be considered to be sparse if the number of significant (non-zero) values N_s present in the vector is much less than N. A sparse vector may then be uniquely described by the values and the positions in the dense vector of each of the N_s significant elements. A sparse vector s may then be represented by two significantly shorter vectors; one sv for the values and one si for the corresponding indices. A sparse vector may then be represented by two vectors sv and si each of length N_s. The memory needed to represent the vector s is $2N_s + 1$ as compared with N. This representation of a sparse vector, which we use in the implementations of our algorithms, is based on the format described in [57].

With the dramatic savings in memory comes programmatic difficulties; programming with sparse data structures may be tedious. We now provide an example of a subroutine which computes the pointwise product of two sparse vectors and places the result in another sparse vector. The inputs are x, ix, nx and y, iy, ny, the parameter $eps = \epsilon$ is the minimum value of any element, and the output is another sparse vector z, iz, nz.

In this example we first initialize three pointers into the vectors x, y, and z. The pointers `ixpr` and `iypr` tell us where we currently are in the corresponding vector, and `izpr` points to the next possible valid position for a product of x and y elements. The main body of the subroutine is defined by the

```
For j = jfirst To J

     Reconstruct(sbar(j),zero(j),j0,stemp(j-j0))

     Reconstruct(zero(j),d(j),j0,dtemp(j-j0))

     SparseVectorProduct(stemp(j-j0),dtemp(j-j0),stilde(j-j0))

     SparseVectorScale(stilde(j-j0),2)

     SparseVectorSquare(dtemp(j-j0),dtilde(j-j0))

     SparseVectorSum(stilde(j-j0),dtilde(j-j0),ssum(j-j0))

     Cutoff(ssum(j-j0),epsilon)

Next j

SparseVectorSquare(stemp(J-j0),stemp(J-j0))

SparseVectorSum(ssum((J-j0),stemp(J-j0),ssum(J-j0))

Cutoff(ssum(J-j0),epsilon)

DecomposeAndSum(ssum,sanswer,danswer)

Cutoff(sanswer,epsilon)
Cutoff(danswer,epsilon)
```

Figure C.2. Pseudocode for the adaptive pointwise square of a function expanded in the wavelet basis.

loop `do 10 k = 1, nx + ny`. The upper limit of the `do` loop is `nx + ny` because we must consider products involving any possible pairs of x and y elements. If we have exhausted either x or y we know that any new products would involve a zero term and we therefore exit the loop. If there are elements remaining in both x and y we then determine if the current x and y indices coincide. If the indices do not coincide we increment the pointer corresponding to the lesser index and go on to the next product. However, if the indices coincide then we compute the product and place its value in the current position in the z vector.

We then test whether or not this newly computed value is greater than the cutoff `eps`. If it is then the newly computed value is a valid element of the sparse vector and the index corresponding to this product is set via `iz(izpr) = ix(ixpr)` and the pointer into the resultant is incremented. If the newly computed value has absolute value less than `eps` then nothing else is done to the resultant vector. In either case the pointers into x and y are incremented and the loop is repeated. We note that if the newly computed value has absolute value less than `eps` then the element in the z vector is overwritten the next time two x and y indices coincide.

It is clear that if one where to implement the pointwise product of two vectors of length n in the dense format the result would be However for the purposes of Chapter 2 the number of elements n would be on the order of 2^{16}, see e.g. the discussion on Burgers equation in Section 2.4.2, and simply multiplying two vectors requires $O(2^{16})$ operations.

```
      subroutine sparse_prodv(x,ix,nx,y,iy,ny,z,iz,nz,eps)
      implicit real*8 (a-h,o-z)
      real*8   x(*),  y(*),  z(*)
      integer ix(*), iy(*), iz(*)

      ixpr = 1
      iypr = 1
      izpr = 1

      do 10 k = 1, nx + ny

           if ( (ixpr .gt. nx) .or. (iypr .gt. ny) ) goto 20

           if (ix(ixpr) .eq. iy(iypr)) then
                z(izpr) = x(ixpr)*y(iypr)

                if (abs(z(izpr)) .gt. eps) then
                     iz(izpr) = ix(ixpr)
                     izpr = izpr+1
                endif

                ixpr = ixpr + 1
                iypr = iypr + 1

           else if (ix(ixpr) .lt. iy(iypr)) then
                ixpr = ixpr + 1
           else
                iypr = iypr + 1
           endif

10    continue

20    continue

      nz = izpr - 1

      return
      end
```

Figure C.3. FORTRAN subroutine for multiplying two sparse vectors and putting the result in a third sparse vector.

```
      subroutine dense_prodv(x,y,z,n,eps)
      implicit real*8 (a-h,o-z)
      real*8   x(*),  y(*),  z(*)

      do 10 k = 1, n

           z(k) = x(k)*y(k)

           if (abs(z(k)) .lt. eps) z(k) = 0.0

10    continue

      return
      end
```

Figure C.4. FORTRAN subroutine for multiplying two dense vectors and putting the result in a third dense vector. Compare with Figure C.3.