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 _____

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

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

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])

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, [4

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

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.

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.

This Chapter is outlined as follows. In Section 2.1 we use the semigroup 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. In Section 2.2 we write

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 analy

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

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 [illegible] (2.1.5) and onl

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

which is equivalent to the standard trapezoidal rule, or

$$I(t) = {}^{1}$$

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

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 appl

the sequence $f(\lambda_k$

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

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

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

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

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

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

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

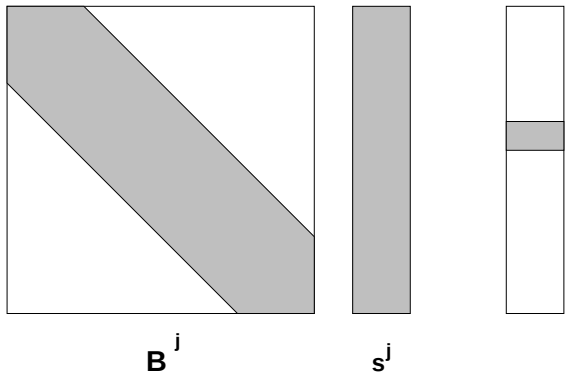
for $j = 1, 2, \ldots, J-1$ and $k \in \mathbb{F}_{2^{n-j}}$ and on the the final, coarse scale

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

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

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.



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

$$||(P_0 f)_\epsilon(x)$$

$\{\varphi_{j,2k+\lambda}(x)\}$ is determined depending on the presence of b

By the vanishing-moment property

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

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 t

Remark:

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

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

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

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

and

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

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

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

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

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

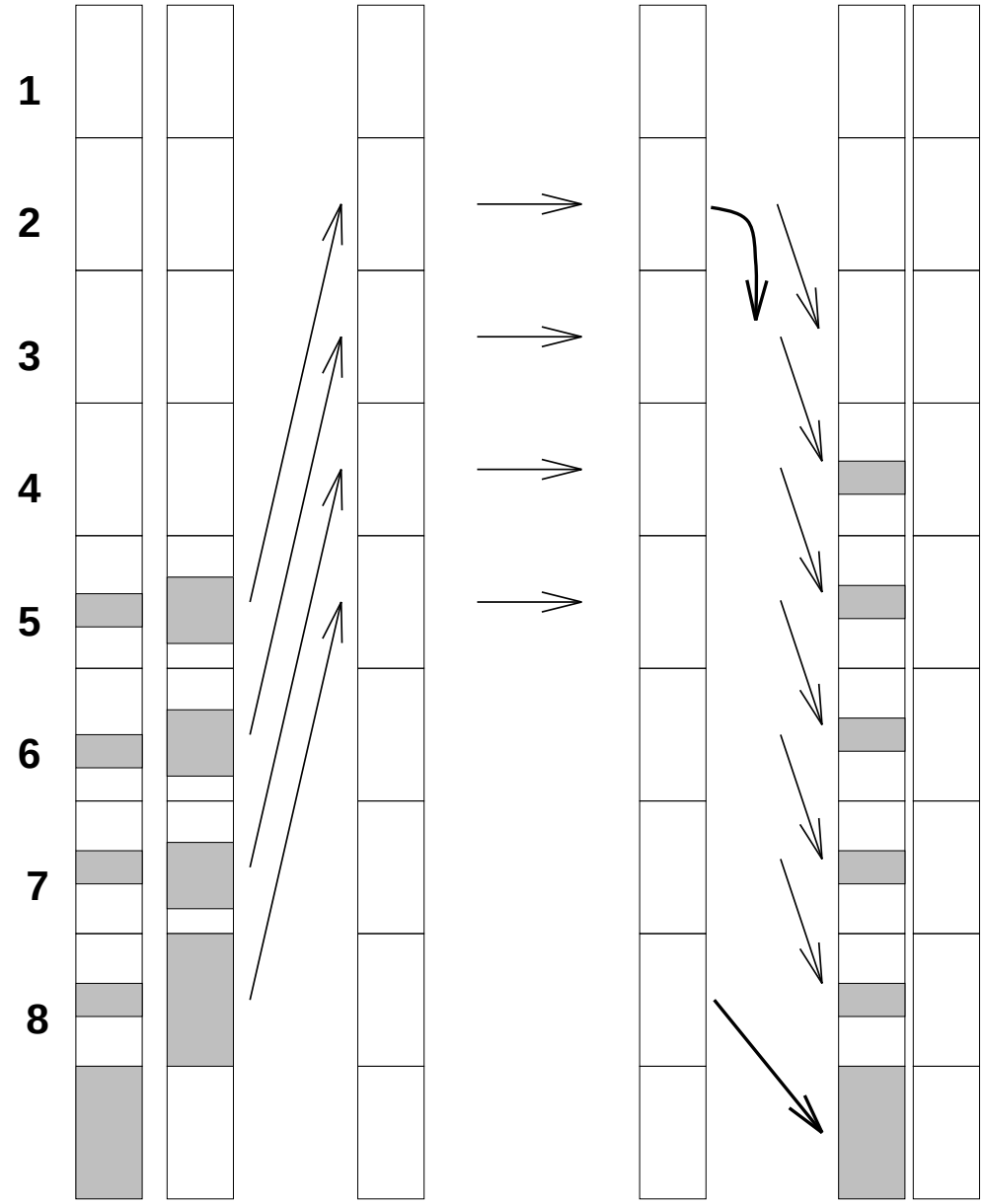
where

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

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

To describe the algorithm H Q V H HYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYYY

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.


1
2
3
4
5
6
7
8

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 $f(P_0 u) \in \mathbf{V}_0$ and expand the projection $f(P_0 u)$ in the 'telescopic' series

$$f(P_0 u) - f(P_J u) = \sum_{j=1}^{J} f(P_{j-1} u) - f(P_j u). \tag{2.3.32}$$

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

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

to arrive at

$$f(P_0 u) = f(P_J u) + \sum_{j=1}^{J} \sum_{n=1}^{N} \frac{f^{(n)}(P_j u)}{n!} (Q_j u)^n + E_{j,N}(f, u). \tag{2.3.34}$$

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.

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 num

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

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

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,

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

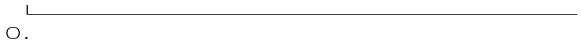
In this example our goal is to use the wavelet representation of the operators appearing in the Crank-Nicolson scheme (2.4.17) to identify a source of error in low order schemes. In the wavelet domain we compare the Crank-Nicolson scheme (which we have taken as an example of a low order scheme) with 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.

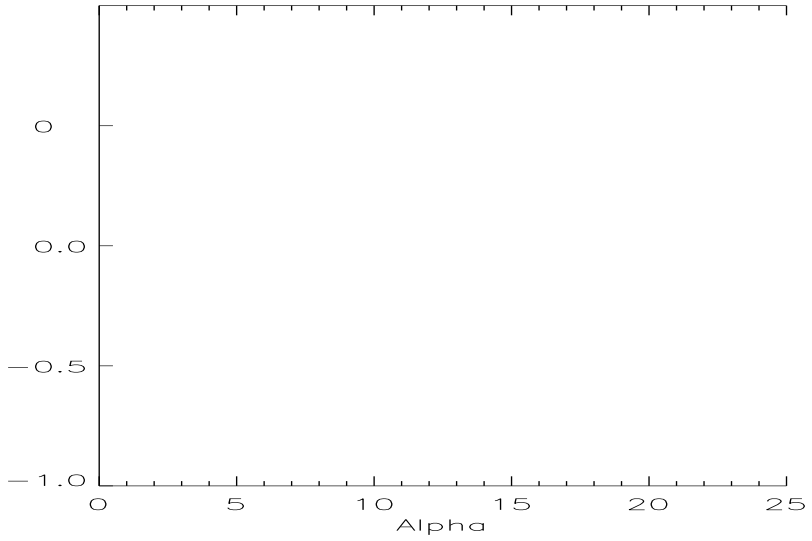
For example, let us consider the following initial condition

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

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 $n =$




0
0.0
−0.5
−1.0
0
5
10
15
20
25
Alpha

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

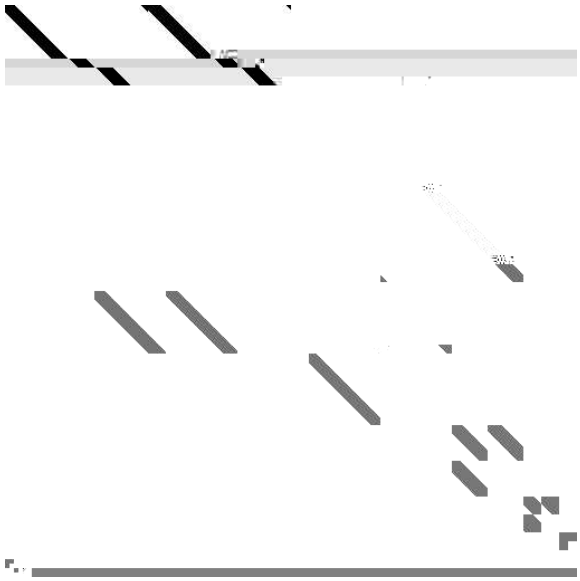


Figure 2.7. NS-form representation of the operator $A^{-1}B$ used in the Crank-Nicolson scheme (2.4.17). Entries of absolute value greater than 10^{-8} are shown in black. The wavelet basis is Daubechies with

$e^{\nu \Delta t \partial_{xx}}$ 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-

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.A in

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)

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

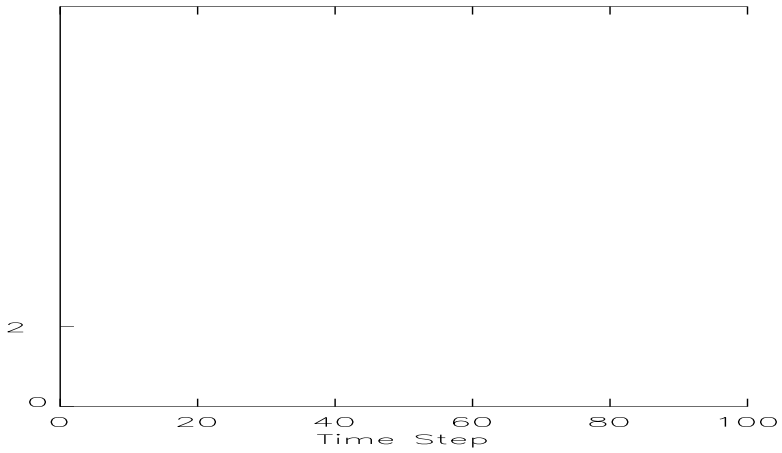
using the pointwise product algorithm described in Section 2.3 and multiply this result by the NS-form representation of $\mathcal{O}_{\mathcal{L},1}$ again using the adaptive mul-

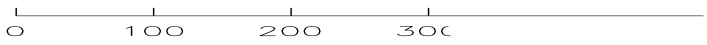
approach by comparing the approximate solution u_w with the exact solution u_e using

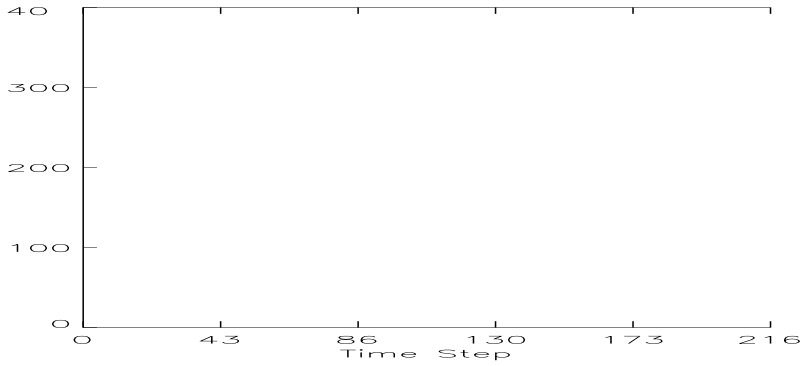
$$\|u_w - u_e\|^2 = 2^{-n} \sum_{w}^{2^n-1}$$

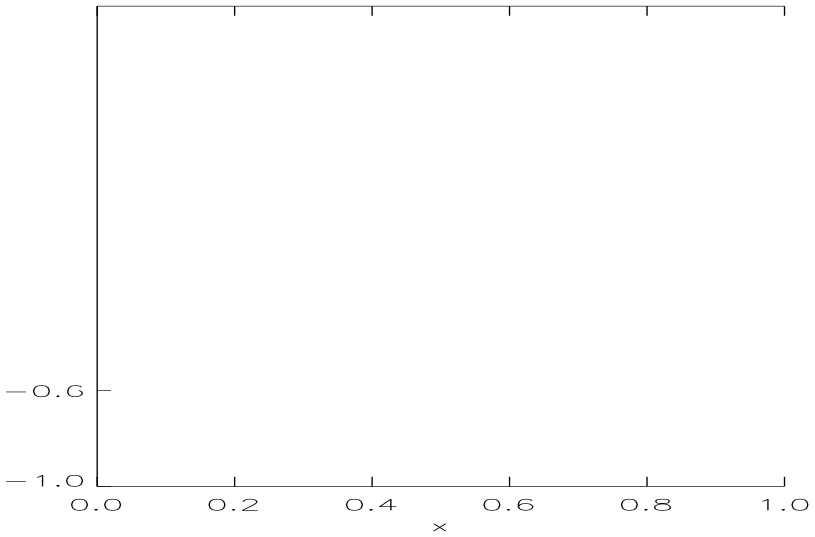
Example 2. In this example $n = 10$, $J = 4$, $\Delta t = 0.001$, and $\nu = 0.01$. We refer to Figures 2.13-2.16. Figure 2.13 illustrates the projection of the solution on $\mathbf{V}_0$ computed using $\epsilon = 10^{-6}$. Figure 2.14 illustrates the error (2.4.44) per time step in the wavelet solution 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

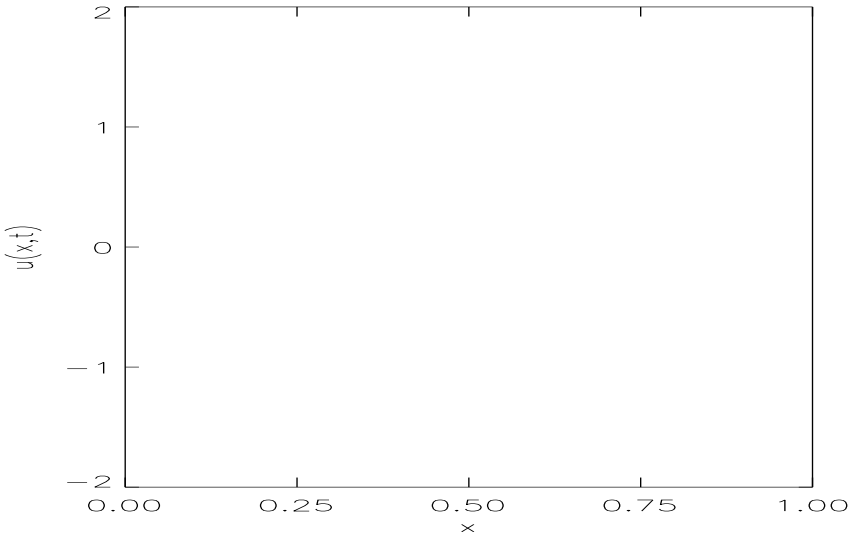
In this example $n = 12$, $\nu = 0.001$, $\Delta t = 0.001$, $J = 6$, and $\epsilon = 10^{-6}$. We refer to Figures 2.24-2.27. Figure 2.24 illustrates the projection of the solution on $\mathbf{V}_0$. Figure 2.25 illustrates the error (2.4.44) per time step in the wavelet solution u_w as compared with the exact solution u_e m b m V Q H We step in th tion e

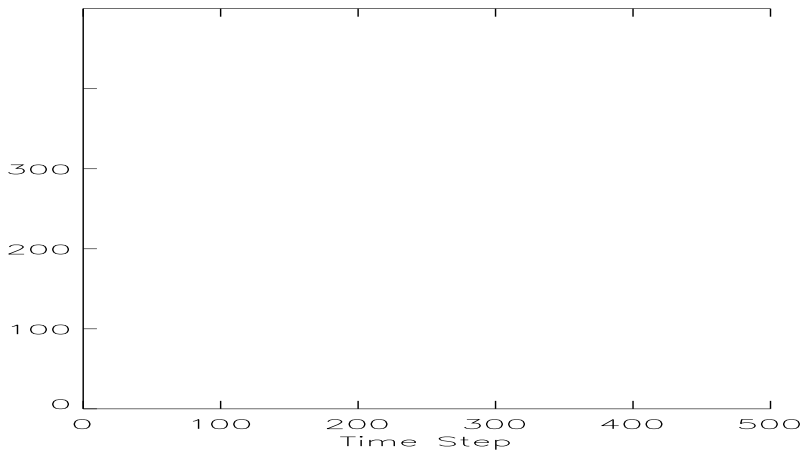

2
0
0
20
40
60
80
100
Time Step


0
100
200
300


400
300
200
100
0
0
43
86
130
173
216
Time Step


−0.6
−1.0
0.0
0.2
0.4
0.6
0.8
1.0
x


2
1
0
−1
−2
u(x,t)
0.00
0.25
0.50
0.75
1.00
x


300
200
100
0
0
100
200
300
400
500
Time Step

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

$$u_t = \nu u_{xx} + f(u) \quad 0 \leq$$

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.


0.0
0.2
0.4

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

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

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

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

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 de [illegible] 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. $f(u) = u$

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 adap i

HWavehe s Vrequire f V QQVas V HHt ire H H en bs QpQare au Hal t f f a p. T V VHH V ro V H H

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. *SIA*

[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.]

[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 Problem

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

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

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

$$\begin{aligned} s_k^j &= s_{k+2^{n-j}}^j \\ d_k^j &= d_{k+2^{n-j}}^j \end{aligned} \qquad (
$$

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

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 V

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 t

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

$$\alpha^j_{k,k'} \quad = \quad \int\int$$

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 on

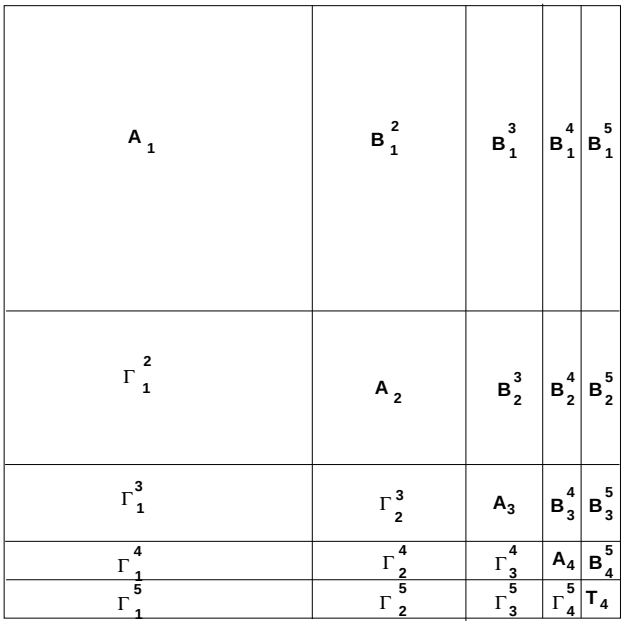
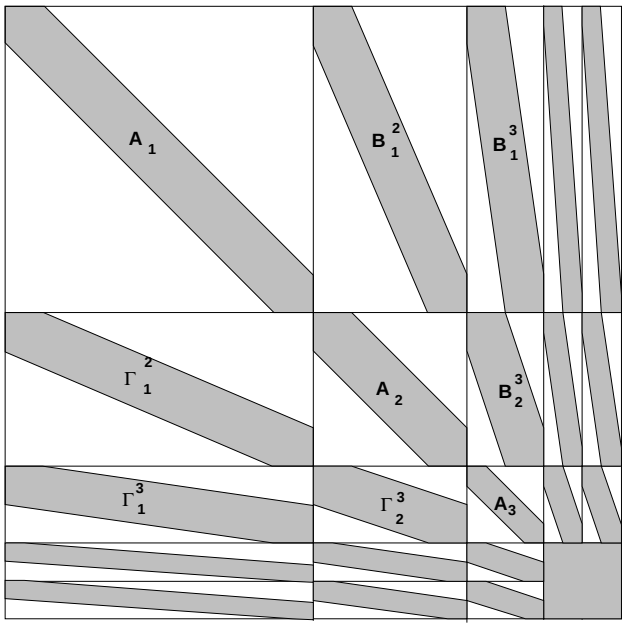


Figure A.3: Organization of the standard form of a matrix.


A 1
B 2 1
B 3 1
Γ 2 1
A 2
B 3 2
Γ 3 1
Γ 3 2
A 3

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

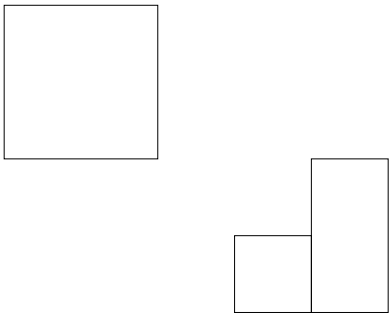
$$\varphi_{J,k}(x)\ \varphi_{J,k'}(y), \tag{A.3.32}$$

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.



The price of uncoupling the scale interactions in (A.3.26) is the need for an

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

using the reconstruction algorithm described in Appendix A.2 as follo

Section 2.2.1 we descriN

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

$$\alpha^j_{k,k'} \quad = \quad \alpha^j_{k-k'} \quad = \quad \int_{-\infty}^{\infty} \psi_{j,k}(x-(k-k'))\frac{d^p}{dx^p}\psi_{j,k'}(x)dx$$

β

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*

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

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,t1,n}]] /. {t1 -> t,t1 -> t0}

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

We note that in what follows $h \equiv \Delta t$ c

```
 10  (10)
h   v    [h]
------------) + c[1][1]
  362880

                   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] + --------- - ---------- +
  362880                            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       362880
```

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)
```

Then, ta

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[1
```

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]
```

```
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 (

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

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

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.[1] For example, setting $c_{0,0} = \frac{1}{6}, c_{1,1} = \frac{2}{3}, c_{2,2} = \frac{1}{6}$ and $c_{2,} + O$,

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

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
```

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 sam

```
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

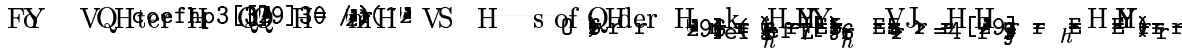
```
In[26]:= For[j=1,j<=5,++j,
coefhp3[j] = Simplify[coefhp2[j] /. Flatten[Join[r21,r22]]];]
```

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
```



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

sum

sum

sum

sum

```
                  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 $\{\tilde{s}^j\}$

```
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)` compute

```
For j = jfirst To J

     Reconstruc
```

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 ▪ 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 ▪ 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.