

A WAVELET-OPTIMIZED, VERY HIGH ORDER ADAPTIVE GRID AND ORDER NUMERICAL METHOD*

LELAND JAMESON[†]

Abstract. Wavelets detect information at different scales and at different locations throughout a computational domain. Furthermore, wavelets can detect the local polynomial content of computational data. Numerical methods are most efficient when the basis functions of the method are similar to the data present. By designing a numerical scheme in a completely adaptive manner around the data present in a computational domain, one can obtain optimal computational efficiency. This paper extends the numerical wavelet-optimized finite difference (WOFD) method to arbitrarily high order, so that one obtains, in effect, an adaptive grid and adaptive order numerical method which can achieve errors equivalent to errors obtained with a “spectrally accurate” numerical method.

Key words. wavelet, spectral methods, finite difference, adaptive methods, very high order methods

AMS subject classifications. 63N30, 65N13

PII. S1064827596301534

1. Introduction. Wavelet transforms provide information about a function with respect to scale and location, in contrast to Fourier transforms, which provide a one-parameter family of coefficients representing the global frequency content. In numerical computations, one often encounters computational data which have a variety of scales at different locations throughout the computation domain. One might conjecture that such a computational environment could best be computed with a wavelet basis.

Generally, one can classify wavelet methods as either a collocation type or a Galerkin type. One can think of the computational parameters in a wavelet collocation method as the point values of the computational variables in the physical space. Likewise, one can think of the computational parameters in a wavelet Galerkin method as the point values in the transform space, in this case the wavelet transform space. In either case, one is working with N real numbers. The best way to evolve these N real numbers in a wavelet or multiresolution framework is a matter of debate. This paper will introduce a numerical method named the wavelet-optimized finite difference method, version 2 (WOFD2), in which wavelets are used only in grid refinement and order selection. WOFD2 is an extension of WOFD, which was originally introduced in [24] and [28]. The argument supporting WOFD is that one should perform all calculations in the physical space, i.e., use a collocation approach, and should evolve these computational variables using finite difference methods on nonuniform grids.

There is a large variety of wavelet methods now in the literature. Some of these methods will be listed here. In [40] a filter bank approach is introduced in which the

*Received by the editors April 3, 1996; accepted for publication (in revised form) February 9, 1997; published electronically July 21, 1998. This research was conducted in part and supported in part by Mitsubishi Heavy Industries, Yokohama, Japan, while the author was employed there from April, 1995 to July, 1996. The work was also supported in part by National Aeronautics and Space Administration under NASA contract NAS1-19480 while the author was in residence at the Institute for Computer Applications in Science and Engineering (ICASE), NASA Langley Research Center, Hampton, VA 23681.

<http://www.siam.org/journals/sisc/19-6/30153.html>

[†]International Pacific Research Center, University of Hawaii, 2525 Correa Rd., Honolulu, HI 96822 (lameson@soest.hawaii.edu).

computational parameters are point values in the physical space. In [22] an interpolating wavelet method is introduced in which the fundamental computation parameters are point values in the physical space. In [20] and [21] polynomial interpolation errors provide the wavelet coefficients, and this information is used to decide where cells in essentially nonoscillatory schemes should be refined. In [3] an interpolating wavelet transform has been used in a Galerkin method for solving elliptic problems on an interval. In [5] a collocation method is introduced with spline wavelets. In [2] a Galerkin-style method is introduced with attention given to time adaptability. In [33] a Galerkin-style method is applied to Burgers's equation. Let us begin with a review of WOFD.

2. The WOFD method. Explanations of WOFD have appeared in many places (see [24], [28], and [14]) but never in a journal article. This review of WOFD will, therefore, be longer than most.

The first occurrence of the argument that wavelets should be used only to analyze computational data for error detection and grid generation was in [24] and subsequently in [28]. Let us motivate the argument supporting WOFD by addressing a number of relevant points with respect to wavelet numerical methods.

2.1. Data compression. The key strength of wavelet methods is data compression. An efficient basis is one in which a given set of data can be represented with as few basis elements as possible. This is so that a sine wave can be represented, without error, with one basis element if the basis is composed of sine waves, whereas a sine wave would require a large number of wavelets without ever decreasing the error to zero. So, are wavelets better than sine waves for computational fluid dynamics? This depends on the problem. If during a computation one always has smooth and periodic data, then sines and cosines are probably appropriate. On the other hand, if the computational domain contains very fine structure in one part of the domain and smooth structure in the remainder of the domain, then it is possible that wavelets might provide an efficient representation of these data. In other words, one can obtain an efficient representation when the basis elements are similar to the features in a given flow. This similarity can be made precise by simply counting the dimension of the space that one uses with a given error tolerance.

2.2. Daubechies wavelet Galerkin methods. Daubechies wavelets have the remarkable property that they are compactly supported in the physical space and orthogonal under translation and dilation. Assuming that one's initial condition is not a basis function, i.e., a wavelet, then one must employ a quadrature formula. This quadrature formula provides the link between physical-space point values and the wavelet subspace at the finest scale. When one observes the action of an adaptive Daubechies wavelet-based numerical method on these physical-space point values, one finds an adaptive grid finite difference method with an order of accuracy roughly double that of the approximation accuracy for the wavelet at hand, i.e., superconvergence (see [24] and [25]).

2.3. Spline wavelet Galerkin methods. Splines can be constructed from B -splines. It is these underlying B -splines that dictate the properties of spline-based wavelet Galerkin methods. Consider, first of all, the linear B -spline that corresponds to the linear element in finite elements. These linear elements are not orthogonal under translation so that when one builds a differentiation matrix, assuming periodicity, one finds $D = M^{-1}A$, where M and A are banded and circulant where the bands are, respectively, $1/6$, $2/3$, $1/6$ and $-1/2$, 0 , $1/2$. Such a matrix yields fourth-order

differentiation accuracy and corresponds to compact finite difference schemes. Therefore, when one begins adaptation based on wavelet coefficients, one finds that spline wavelet Galerkin methods correspond to an adaptive element finite element method (h -refinement), and to adaptive grid compact finite difference methods; see [24] and [26]. When the order of the B -spline is increased, the same results hold with the order of accuracy increasing.

2.4. Other wavelet Galerkin methods (known and unknown). Let Galerkin methods be summarized as methods in which the degrees of freedom are the expansion coefficients of a set of basis functions. These expansion coefficients are, by definition, not in the physical space. However, partial differential equations (PDEs) are generally specified as equations with boundary conditions in the physical space, so that nonlinearities, etc., when treated in a wavelet subspace, are often unnecessarily complicated. In short, if a quantity is specified in a given space, then implementation is most straightforward in that space. There appears to be no compelling reason to work with Galerkin-style coefficients in a wavelet method, and to try to create a practical Galerkin-style method with wavelets seems to be unnecessarily difficult when an equivalent method can be implemented in the physical space.

2.5. Boundary conditions and nonlinear terms. Nonperiodic boundary conditions are one of the weakest points of wavelet methods. For wavelet Galerkin methods it is unlikely that sufficient accuracy will ever be achieved at the boundary for either Daubechies-based methods (see [27]), or spline-based methods (see [17]). Beyond sufficient accuracy, it is far from a straightforward task to impose even the simplest nonperiodic boundary conditions for wavelet Galerkin methods.

A second obstacle with Galerkin methods is the evaluation of nonlinear terms. That is, if one has the wavelet coefficients of $u(x)$, then how can one easily obtain the coefficients of $u^2(x)$? Furthermore, what if the nonlinearity is $e^{u(x)}$? In short, one need not struggle with all the problems of implementing a Galerkin method if one chooses to work in the physical space from the beginning.

2.6. Wavelet collocation methods (known and unknown). Collocation methods involve numerical operators acting on point values in the physical space. Generally, wavelet collocation methods are created by choosing a wavelet and some kind of computational grid structure which will be dynamically adapted. Recall that the approximation properties of wavelet methods are such that they are constructed in order to reproduce algebraic polynomials perfectly up to a given order. Also recall that finite difference methods are constructed from an underlying algebraic polynomial. When one adds the adaptivity of a wavelet collocation method, one obtains in the physical space operations which are exact for some set of polynomials and which are performed on a nonuniform grid in the physical space. In effect, one obtains finite differencing on nonuniform grids in the physical space. In addition, the operators are far from optimal in obtaining a given accuracy for a given stencil width. The stencil is usually longer than is needed, with the extra degrees of freedom having no specific function.

2.7. Working in the physical space. Ideally, one would like a physical-space implementation to be very similar to a Galerkin implementation. In spectral methods it is the case that collocation methods and Galerkin methods have the same rate of convergence and in practice one sees very little difference between the essence of the two approaches; see [6]. Generally, proofs are easier with Galerkin methods, whereas implementation is more practical with collocation methods.

For wavelets it would also be desirable to have some kind of parallel between Galerkin methods and a physical-space implementation. Generally, a straightforward wavelet collocation method will not have the same rate of convergence for the differentiation operator as the Galerkin method, due to the superconvergence which occurs with a Galerkin approach. This can easily be seen by examining the case of B -splines, where the accuracy of the differentiation matrix found for an n th-order B -spline using a Galerkin method is equal to the accuracy of the differentiation matrix of a $(2n + 1)$ th-order B -spline using a collocation method; see [37]. On the other hand, if one understands the physical-space action of a wavelet Galerkin method, then one can implement directly a numerical method in the physical space which parallels the Galerkin method without all its associated problems. In short, one can take the useful part of wavelet methods, such as multiresolution, and implement it in a viable manner in the physical space.

2.8. Numerical examples of WOFD. The first test case for WOFD came with Burgers's equation in one dimension, given by

$$(2.1) \quad U_t = (U^2)_x + \epsilon U_{xx},$$

with the initial condition

$$(2.2) \quad U(x, 0) = \frac{1}{3} + \frac{2}{3} \sin(x),$$

and with periodic boundary conditions

$$(2.3) \quad U(-\pi, t) = U(\pi, t)$$

fixed at initial condition values

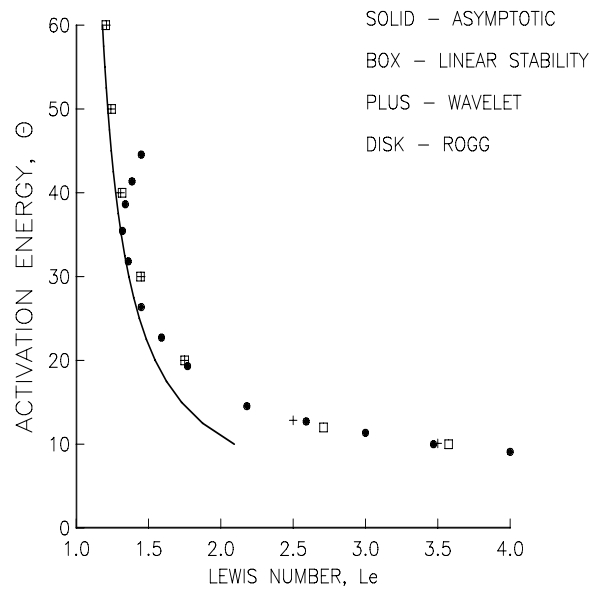
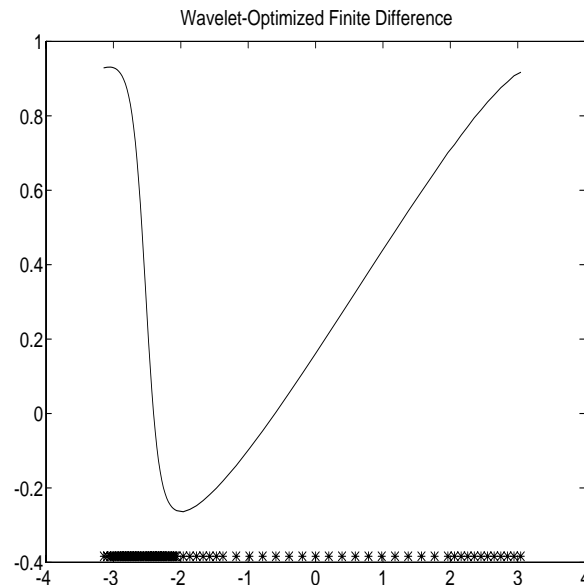
$$(2.4) \quad U(-\pi, t) = U(-\pi, 0), \quad U(\pi, t) = U(\pi, 0).$$

The results are that WOFD can approximate the uniform grid solution as closely as desired with far fewer floating-point operations; see [24] and [28]. See Figures 2.2 and 2.3 for computed solutions.

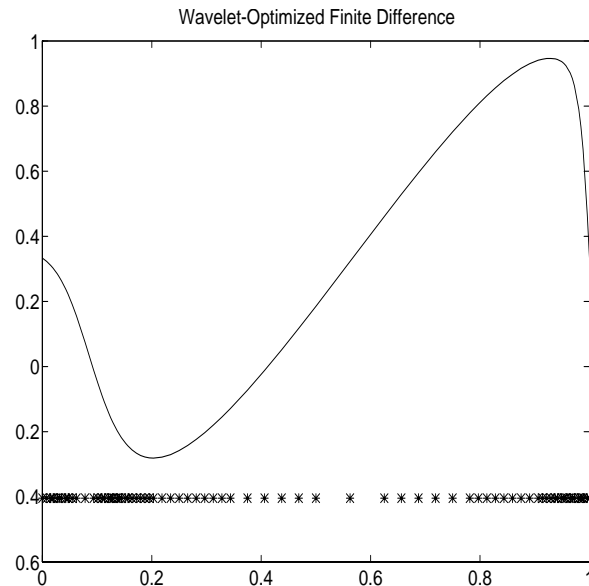
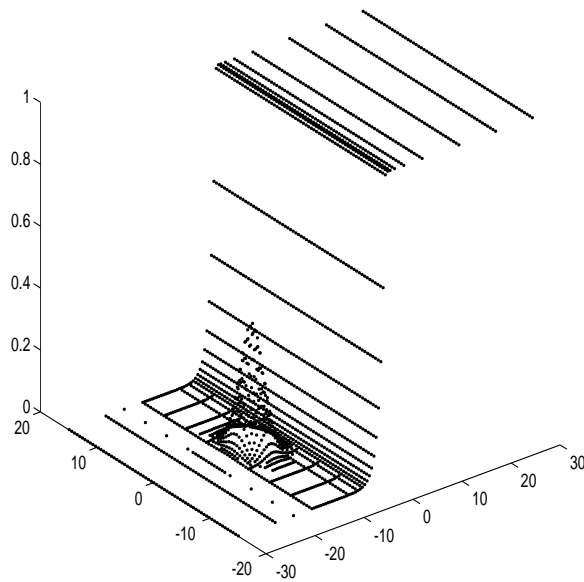
The ultimate goal behind the creation of numerical methods should be to numerically simulate the equations of physics and obtain correct results which cannot be obtained by existing numerical methods. WOFD has already had one such success.

This success came in the area of combustion, in which the goal was to obtain the stability curve in the Lewis number–activation energy plane; see Figure 2.1. Other numerical methods produced results which were inconsistent with theory for large activation energies, whereas WOFD produced results which were consistent with linear stability analysis and with asymptotic results; see [29] and [34]. Note that in this simulation of combustion, WOFD did not require special modification. The equations were simply typed in and the numerical method adapted to the very thin region of the flame front, producing a very convincing answer.

Finally, in two dimensions WOFD appears to work as well as in one dimension. Figures 2.4 and 2.5 show a flame front propagating into a Gaussian pulse and the corresponding computational grid. WOFD is currently being thoroughly tested on the two-dimensional equivalent of the above-mentioned successful application to one-dimensional combustion.

FIG. 2.1. *Stability curve in Lewis number-activation energy plane.*FIG. 2.2. *Burgers's equation with periodic boundary conditions.*

2.9. WOFD2: Spectral accuracy. WOFD2 is the spectrally accurate version of WOFD and is the method introduced in this paper. WOFD2, as in WOFD, uses wavelets for signal analysis of flow features. That is, wavelets provide a perfect mechanism for grid selection where sparse grids are placed in regions of the domain where the flow is “smooth,” and fine grids are placed in regions of the domain where the flow features are “rough” or perhaps highly oscillatory. In addition, WOFD2 uses wavelets to select the appropriate order of a numerical scheme to obtain optimal, and

FIG. 2.3. *Burgers's equation with boundary values fixed.*FIG. 2.4. *Flame front encountering Gaussian pulse.*

adaptive, efficiency during a calculation. In other words, if a portion of a flow is composed of large smooth features, then a high-order, low grid point density is optimal. If another portion of the same flow is composed of very fine scale features, then optimality is obtained by increasing the density of the grid points and decreasing the order. In short, WOFD2 is an adaptive grid and adaptive order numerical method that obtains errors as low as the errors obtained for spectral methods. Note that if a flow is not smooth, that one might obtain, for a fixed grid point density, smaller errors

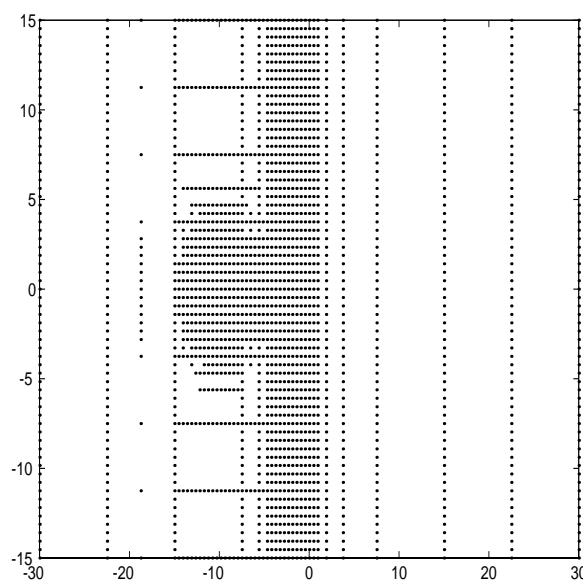


FIG. 2.5. *Computational domain selected by wavelets for flame front encountering Gaussian pulse.*

with a fourth-order method than with a 20th-order method. So, the key point is to “match” the numerical method to the flow. Wavelets provide the necessary machinery to obtain this proper match.

One can argue that high-order numerical methods are appropriate for problems in a) the direct numerical simulation of turbulence (see [31]), b) flows with shocks and nonlinear physics (see [13]), and c) flows with smooth propagating structures such as those encountered in aeroacoustics. Assertion c) is based on convergence properties of the hp -refinement method in finite elements (see [1]), in which convergence is very fast for high-order polynomials as long as the function at hand is smooth. In addition, high-order methods are more efficient for long time integration of unsteady flow problems; see [32].

This paper introduces a numerical method which combines very high order differencing with a wavelet-based grid and order selection mechanism. Here very high order differencing will be schemes of order greater than or equal to 8, i.e., perhaps even 16, 20, or 32. Such high orders of accuracy can produce solutions that are very close to those produced by spectral methods. See [7] for a spectral method on arbitrary grids.

The first issue to address is what is the best way to generate high-order difference operators. This is the subject of the next section.

3. Generating difference equations. Given a vector of N numbers \vec{f} how can one get an approximate value of the derivative \vec{f}' at the i th point and how good will this approximate value be? Generally speaking, the more elements around the i th point of \vec{f} that are used to approximate \vec{f}' , the better the approximation will be. Common finite difference formulas are found by fitting an algebraic polynomial of degree q locally around the i th point of a vector \vec{f} of evenly spaced elements to obtain difference approximations of accuracy $q - 1$. This section will generalize this concept to find the difference equations of arbitrary accuracy on arbitrary grids using algebraic, trigonometric, cosine, and exponential polynomials. As special cases, one

can obtain all the usual finite difference formulas as well as the Fourier collocation and Chebyshev collocation spectral differential matrices.

Two methods of generating the differencing coefficients will be introduced. The first method explains how to set up a system of equations which will have as a solution the differencing coefficients. The second method is the derivation of differencing coefficients by interpolation. It is this second method that is used throughout the paper for the actual generation of difference equations.

3.1. Setting up a linear system. The problem is to find a set of coefficients $\{r_k\}$ which combines the raw data in a vector \vec{f} to provide an approximation to a derivative given by

$$(3.1) \quad f'(x_j) = \sum_{k=left}^{k=right} r_k f(x_k).$$

If we require (3.1) to be exact for algebraic, trigonometric, cosine, or exponential polynomials, then a linear system of equations can be solved to find an appropriate set of differencing coefficients $\{r_k\}$. Let $b(x)$ denote a fundamental basis element from which a basis can be generated by taking powers of $b(x)$, for $b(x) = x$, $b(x) = e^{ix}$, $b(x) = \cos(x)$, or $b(x) = e^x$. That is, we require that the derivative be exact up to a given order N on the numerical grid. The system of equations to be solved for a centered differentiation stencil is

$$(3.2) \quad n(b(x_j))^{n-1} b'(x_j) = \sum_{k=-L}^L r_k (b(x_{j+k}))^n.$$

From this equation one can generate a system of equations with the N requirements that N functions be differentiated exactly, and N degrees of freedom given by the N differencing coefficients r_k . If one is near a boundary, then the stencil is biased. Since this type of system is well known for algebraic polynomials, an example for the less well known trigonometric polynomials will be given.

Consider a trigonometric polynomial on a three-point centered stencil. The first equation simply requires that the derivative of a constant be zero:

$$(3.3) \quad 0 = r_{-1} + r_0 + r_1.$$

Note that this is the same equation as for algebraic polynomials, since $(x)^0 = (e^{ix})^0$. The next two equations come from requiring that the $n = 1$ mode be differentiated exactly:

$$(3.4) \quad ie^{ix_0} = r_{-1}e^{ix_{-1}} + r_0e^{ix_0} + r_1e^{ix_1}.$$

One now obtains the two equations by equating the real and imaginary parts. These three equations can be solved for the three coefficients r_{-1}, r_0, r_1 .

Similarly, one can find the coefficients for higher-order schemes by requiring that more modes be differentiated exactly. Note that no restrictions were placed on the grid. Differencing formulas can be found on arbitrary grids as easily as they can be found on uniform grids. Also, note that the Fourier spectral differentiation matrix can be found from the above procedure by requiring that the grid be uniform and that the differencing formulas have maximum accuracy on a given grid. That is, if one is working on a grid of size 33, then one must require that the first 16 modes and the 0th mode be differentiated exactly.

3.2. Interpolation. A second approach, and the one used in this paper, is to generate differencing coefficients by first interpolating a polynomial through a set of data, followed by differentiation of this polynomial and evaluation at a grid point.

The main reason that differentiation was studied with a variety of types of operators was to find out if there was any advantage to using, say, trigonometric polynomials as opposed to algebraic polynomials when the function to be differentiated was, for example, a Gaussian pulse. It seemed like an appropriate study to undertake given the current research activity in the area of aeroacoustics, where one is often confronted with the need to computationally propagate some type of wave motion. The thought was that perhaps trigonometric polynomials might have some advantage in propagating wave motion over the more common algebraic polynomials. One of the conclusions of this section is that there is no advantage and that one should simply use algebraic polynomials for the generation of differencing equations. In fact, the only important issues involved with obtaining approximate derivatives are the order of the finite difference operator and the density of the numerical grid.

The most important reference for this section is [12]. The following four subsections will cite the interpolation formulas for the four types of interpolation, and hence differentiation, considered in this section.

3.2.1. Algebraic polynomials. Interpolation with algebraic polynomials is probably the most common form, and it is from this type that common uniform grid finite difference methods can be found. Using the following formula one can find the finite difference coefficients for an arbitrary grid of arbitrary order. One simply fits the polynomial to the data, followed by differentiation of the polynomial, and finally one evaluates the polynomial at the point of interest. The well-known Lagrange interpolation formula for algebraic interpolation is

$$(3.5) \quad A_j(x) = \prod_{k=0, k \neq j}^n (x - x_k) \bigg/ \prod_{k=0, k \neq j}^n (x_j - x_k),$$

where $A_j(x_k) = \delta_{jk}$. For given values w_0, w_1, \dots, w_n , the polynomial

$$(3.6) \quad p_n(x) = \sum_{k=0}^n w_k A_k(x)$$

in P_n and takes on the values

$$(3.7) \quad p_n(x_k) = w_k$$

at the points x_i , for $k = 0, 1, \dots, n$.

3.2.2. Trigonometric polynomials. As seen from the previous section, one can also generate difference operators by using trigonometric functions as the fundamental interpolation elements. The following is the appropriate Lagrange-type interpolation formula; see [12].

For $-\pi \leq x_0 < x_1 < \dots < x_{2n} < \pi$,

$$(3.8) \quad T_j(x) = \prod_{k=0, k \neq j}^{2n} \sin \frac{1}{2}(x - x_k) \bigg/ \prod_{k=0, k \neq j}^{2n} \sin \frac{1}{2}(x_j - x_k).$$

The function

$$(3.9) \quad T(x) = \sum_{k=0}^{2n} w_k T_k(x)$$

is the unique solution of the interpolation problem

$$(3.10) \quad T(x_k) = w_k,$$

for $k = 0, 1, \dots, 2n$. Again, one can derive finite difference coefficients by interpolating to a function, followed by differentiation of the interpolation polynomial and evaluation at the point of interest. The following section will prove that such difference equations obey order properties just as the usual difference equations derived from algebraic polynomials do.

3.2.3. Cosine polynomials. The comparable Lagrange-style interpolation formula for cosine polynomials is the following; see [12].

Given $n + 1$ distinct points $0 \leq x_0 < x_1 < \dots < x_n < \pi$, set

$$(3.11) \quad C_j(x) = \prod_{k=0, k \neq j}^n (\cos x - \cos x_k) \bigg/ \prod_{k=0, k \neq j}^n (\cos x_j - \cos x_k).$$

Then C_j is a cosine polynomial of order less than or equal to n , $C_j(x) = \sum_{k=0}^n a_k \cos(kx)$, for which $C_j(x_k) = \delta_{jk}$. Given $n + 1$ distinct values w_0, w_1, \dots, w_n , there is a unique cosine polynomial of order less than or equal to n , $C(x)$, for which $C(x_k) = w_k$, $k = 0, 1, \dots, n$, given by

$$(3.12) \quad C(x) = \sum_{k=0}^n w_k C_k(x).$$

Note that $\frac{d}{dx} C(x)|_{0,\pi} = 0$ since $\frac{d}{dx} C_k(x)|_{0,\pi} = 0$ for all k . For this reason, difference operators based on cosine polynomials will not be considered in general, but will be compared in a later section to Chebyshev spectral methods. As above, the differencing coefficients are found by first fitting the trigonometric polynomial to the data, followed by differentiation of the polynomial and finally evaluation at the point of interest.

3.2.4. Exponential polynomials. The final polynomial to be tested is the exponential polynomial

$$(3.13) \quad E_j(x) = \prod_{k=0, k \neq j}^n (e^x - e^{x_k}) \bigg/ \prod_{k=0, k \neq j}^n (e^{x_j} - e^{x_k}),$$

where the interpolation polynomial is given by

$$(3.14) \quad E(x) = \sum_{k=0}^n w_k E_k(x),$$

and

$$(3.15) \quad E(x_k) = w_k.$$

3.3. Truncation error and differentiation accuracy. The purpose of this section is to illustrate algebraically that one obtains differentiation order-of-accuracy properties for all four types of differentiation operators, which are similar to the standard order-of-accuracy properties obtained with the usual algebraic interpolation. In short, if one interpolates an N th-order polynomial, then one obtains a reduction in differentiation error of $(\frac{1}{2})^N$ when the density of the grid is doubled. This order of accuracy is obtained regardless of the type of polynomials that are used.

Recall that the remainder for algebraic polynomial interpolation is (see [12])

$$(3.16) \quad f(x) - p_n(x) = \frac{(x - x_0)(x - x_1) \cdots (x - x_n)}{(n + 1)!} f^{(n+1)}(\xi),$$

where ξ lies between the smallest and the largest x_i . The following section will show that a similar expression can be obtained for any polynomial constructed from powers of a given function.

3.3.1. Truncation error for interpolation by powers. There are some subtle issues concerning a general proof of truncation error and accuracy for interpolation by a polynomial constructed from the powers of a general function $g(x)$; see [8]. The following demonstration will illustrate the essential algebraic steps that one follows to obtain accuracy while avoiding the subtle issues. In short, let the polynomial element $g(x)$ and the function to be approximated, $f(x)$, be “well behaved.”

Let

$$p(x) = \sum_{k=0}^n a_k (g(x))^k$$

be the polynomial which interpolates $f(x)$ at x_0, x_1, \dots, x_n , $p(x_i) = f(x_i)$. Then

$$(3.17) \quad f(x) - p(x) = \frac{p^{(n+1)}(\xi) - f^{(n+1)}(\xi)}{\phi^{(n+1)}(\xi)} \phi(x),$$

where

$$(3.18) \quad \phi(x) = (g(x) - g(x_0))(g(x) - g(x_1)) \cdots (g(x) - g(x_n)),$$

and where ξ lies between the smallest and the largest x_i .

Demonstration of truncation error. Note that much of this demonstration is the same as that which can be found in a standard numerical analysis text for the remainder term in algebraic interpolation; see [10].

Define $H(z)$ such that

$$(3.19) \quad H(z) = f(z) - p(z) - R(x)\phi(z),$$

where $R(x)$ is defined such that $H(x) = 0$. Note that $H(x_i) = 0$ for $i = 0, \dots, n$, since $p(x_i) = f(x_i)$ and $\phi(x_i) = 0$. From Rolle’s theorem it follows that there exists a point ξ in the interval defined by the smallest and largest x_i ’s such that $H^{(n+1)}(\xi) = 0$. This implies that

$$(3.20) \quad R(x) = \frac{f^{(n+1)}(\xi) - p^{(n+1)}(\xi)}{\phi^{(n+1)}(\xi)}.$$

Now substitute this into the expression for $H(z)$ and set $z = x$ to get

$$(3.21) \quad f(x) - p(x) = \frac{f^{(n+1)}(\xi) - p^{(n+1)}(\xi)}{\phi^{(n+1)}(\xi)} \phi(x).$$

This is the desired expression.

Note that in the above demonstration, if the polynomial is algebraic then $p^{(n+1)}(z) = 0$ and $\phi^{(n+1)}(z) = (n+1)!$, but for a general $g(x)$ these two functions are just a measure of the smoothness of the basic interpolation element $g(x)$. $f^{(n+1)}(\xi)$ still remains a measure of the smoothness of the function one is interpolating to, and $\phi(x)$ is a function dependent on the grid distribution.

3.3.2. Differentiation accuracy. The primary interest here is to understand the behavior of the derivative operators derived from the various types of interpolation outlined above as the grid is refined. That is,

$$(3.22) \quad f'(x) - p'(x) = Q(\xi)\phi'(x),$$

where $Q(\xi) = \frac{f^{(n+1)}(\xi) - p^{(n+1)}(\xi)}{\phi^{(n+1)}(\xi)}$, and $\phi'(x)$ will dictate the behavior as the grid is refined. It will be shown that the behavior of $\phi'(x)$ is essentially independent of the basic interpolation element $g(x)$ and depends only on the order of the interpolation.

Demonstration of accuracy. Let h denote the smallest spacing in the numerical grid. Then

$$(3.23) \quad \phi'(x_0) = Ch^n + h.o.t.,$$

where n is the highest power in the interpolation polynomial, and the point x_0 is an arbitrary grid point inside the interpolation stencil.

First of all,

$$(3.24) \quad \phi'(x_0) = g'(x_0)(g(x_0) - g(x_1))(g(x_0) - g(x_2)) \cdots (g(x_0) - g(x_n)).$$

Expand about zero the function $g(x)$ such that

$$(3.25) \quad g(x) = g(0) + g'(0)x + g''(0)x^2/2 + \cdots,$$

and examine the difference

$$(3.26) \quad \begin{aligned} g(x_0) - g(x_1) &= g'(0)(x_0 - x_1) + g''(0)/2(x_0^2 - x_1^2) \\ &+ g'''(0)/6(x_0^3 - x_1^3) + \cdots. \end{aligned}$$

Without loss of generality let one of the points be zero, say $x_0 = 0$, to obtain

$$(3.27) \quad g(0) - g(x_1) = g'(0)(-x_1) + g''(0)/2(-x_1^2) + g'''(0)/6(-x_1^3) + \cdots,$$

or

$$(3.28) \quad g(0) - g(x_1) = -x_1 \left(\sum_{m=1}^{\infty} \frac{g^{(m)}(0)}{m!} x_1^{m-1} \right),$$

and one can see that the first term in the difference is linear. If x_0 is not zero, then one obtains the factorization

$$(3.29) \quad x^q - y^q = (x - y) \left(\sum_{i=0}^{q-1} x^i y^{q-1-i} \right),$$

and hence

$$(3.30) \quad g(x_0) - g(x_1) = (x_0 - x_1) \left(\sum_{m=1}^{\infty} \frac{g^{(m)}(0)}{m!} \sum_{k=0}^{m-1} x_0^k x_1^{m-1-k} \right).$$

It should be clear that the first term in the difference $g(x_0) - g(x_1)$ is the linear term, and that doubling the grid such that between every two points another point is placed is, therefore, half the distance to a first order approximation. For each of the differences $x_i - x_j$ there exists a constant $c_{i,j}$ such that the difference

$$(3.31) \quad x_i - x_j = c_{i,j}h$$

can be expressed in terms of the smallest grid spacing h . Let $C = \prod_{j=1}^n c_{0,j}$. Then it is, therefore, clear that

$$(3.32) \quad \phi'(x_0) = Ch^n + h.o.t.$$

Consider the following special cases which include all the polynomial types discussed above:

- $g(x) = x$, $g^{(m)}(0) = 0$ for $m \neq 1$,
- $g(x) = e^x$, $g^{(m)}(0) = 1 \forall m$,
- $g(x) = e^{ix}$, $g^{(m)}(0) = i^m$,
- $g(x) = \cos(x)$, $g^{(m)}(0) = 0$, for m odd, and $g^{(m)}(0) = (-1)^{m/2}$, for m even.

For a simple illustration, consider the following two examples of algebraic and exponential interpolation.

Algebraic interpolation. Consider the simple case of interpolating an algebraic quadratic polynomial $p_2(x)$ to a function $f(x)$ at the grid points $x_0 < x_1 < x_2$: $p_2(x_i) = f(x_i)$, $i = 0, 1, 2$. The remainder term for some ξ , $x_0 \leq \xi \leq x_2$, is

$$(3.33) \quad f(x) - p_2(x) = (x - x_0)(x - x_1)(x - x_2) \frac{1}{3!} f^{(3)}(\xi).$$

Now, differentiate and evaluate at $x = x_1$ to obtain

$$(3.34) \quad f'(x_1) - p_2'(x_1) = (x_1 - x_0)(x_1 - x_2) \frac{1}{3!} f^{(3)}(\xi) = Ch^2 f^{(3)}(\xi),$$

where $h = x_1 - x_0 = x_2 - x_1$. If the grid is evenly spaced, then the differences $(x_i - x_j)$ are some integer multiple of the smallest difference, which one can denote by h . If one doubles the number of grid points, then each of the distances $(x_i - x_j)$ becomes half as large and the accuracy for this quadratic example will be two.

The general statement for algebraic interpolation. In general, one can expect that algebraic polynomial interpolation with a polynomial of order n , $p_n(x)$, will produce a differencing operator also of order n . This can be seen from the portion of the truncation error that depends on the grid distribution, given by

$$(3.35) \quad \prod_{i=0}^n (x - x_i).$$

This product contains $n + 1$ terms. After differentiation and evaluation at a point x_k , the product

$$(3.36) \quad \prod_{i=0}^{n-1} (x_k - x_i)$$

will contain n terms. When the grid density is doubled by adding a point between every two points, then each distance $(x_k - x_i)$ will decrease by a factor of two. The product of these factors will be a multiple of $(\frac{1}{2})^n$, and hence the accuracy of n is achieved.

Exponential interpolation. If, on the other hand, $p_2(x)$ is now an exponential polynomial, then after differentiation and evaluation one obtains

$$(3.37) \quad f'(x_1) - p'_2(x_1) = e^{x_1}(e^{x_1} - e^{x_0})(e^{x_1} - e^{x_2}) \frac{1}{3!} f^{(3)}(\xi).$$

Without loss of generality, let $x_1 = 0$. Then the product

$$(3.38) \quad f'(0) - p'_2(0) = (e^{x_1} - e^{x_0})(e^{x_1} - e^{x_2})$$

becomes

$$(3.39) \quad f'(0) - p'_2(0) = (x_0 + x_0^2/2 + \cdots)(x_2 + x_2^2/2 + \cdots),$$

and one can see that if the distance to zero is halved, and hence x_0 and x_2 are divided by two, then the leading-order terms on the right-hand side dictate that the error be reduced by four to a first order approximation.

Therefore, one can expect the accuracy to behave as it does for algebraic polynomials. That is, doubling the grid points will decrease the error by $(1/2)^2$ to first order.

The general statement for exponential interpolation. In general, one can expect that exponential polynomial interpolation with a polynomial of order n , $p_n(x)$, will produce a differencing operator also of order n , which is the same result as for algebraic interpolation. As can be seen from the above example, differentiation and evaluation at a point x_k will produce a leading-order term that is a product of n terms given by

$$(3.40) \quad \prod_{i=0}^{n-1} (x_k - x_i),$$

and accuracy of order n is achieved.

3.4. A numerical check of accuracy. This section will verify that the differentiation operators that are generated from algebraic, trigonometric, and exponential polynomials all exhibit the same order property, which depends only on the order of the polynomial interpolation. The difference operators will be tested on the function

$$(3.41) \quad f(x) = \frac{1}{2 + \cos(2x)}$$

defined on $[0, \pi]$. $f(x)$ is chosen because it is periodic but not exactly a trigonometric or algebraic polynomial. Table 3.1 illustrates the order property. All of the errors are L_2 .

A general question arises. Is there any advantage to using, say, a trigonometric polynomial for the generation of difference equations over, say, the usual algebraic polynomial? From this study the answer appears to be no. The essence depends on the ability of the interpolating polynomial to locally approximate the function at hand. If the function is not exactly a trigonometric or algebraic polynomial, as is

TABLE 3.1
A comparison of errors for various types of differentiation operators.

Grid pts	Alg err	Err ratio	Trig err	Err ratio	Exp err	Err ratio
16	$7.72 * 10^{-4}$		$8.23 * 10^{-4}$		$4.10 * 10^{-3}$	
32	$4.06 * 10^{-6}$	$2^{7.6}$	$3.66 * 10^{-6}$	$2^{7.8}$	$1.25 * 10^{-5}$	$2^{8.4}$
64	$8.02 * 10^{-9}$	$2^{9.0}$	$7.59 * 10^{-9}$	$2^{8.9}$	$1.95 * 10^{-8}$	$2^{9.3}$
128	$9.54 * 10^{-12}$	$2^{9.7}$	$8.75 * 10^{-12}$	$2^{9.8}$	$2.17 * 10^{-11}$	$2^{9.8}$
256	$9.80 * 10^{-15}$	$2^{9.9}$	$9.04 * 10^{-15}$	$2^{9.9}$	$2.20 * 10^{-14}$	$2^{9.9}$
512	$9.70 * 10^{-18}$	$2^{10.0}$	$8.95 * 10^{-18}$	$2^{10.0}$	$2.17 * 10^{-17}$	$2^{10.0}$

likely, then there is no advantage for either approach. Such an issue is important when one is considering the propagation of, say, a pulse in the application of aeroacoustics. A pulse will locally be neither an algebraic or a trigonometric polynomial. In short, a wave, or pulse, cannot be propagated accurately if it cannot first be differentiated accurately, and it cannot be differentiated accurately if it cannot first be approximated accurately.

4. High-order methods. Spectral collocation methods are often given the probable misnomer of “infinitely accurate.” In a manner consistent with finite difference methods, spectral collocation methods will be assigned the accuracy of $N - 1$ when applied on a grid of N points.

This section will begin by connecting spectral collocation methods to finite difference methods. That is, spectral methods will be viewed from the point of view of the maximum finite difference method on a given grid. Following the comments on this connection, a case will be made for applying very high order algebraically generated finite difference operators on Chebyshev grids or, equivalently, applying very high order cosine polynomial generated finite difference operators on a uniform grid. This second process of using cosine polynomials will require a mapping of the independent variable from, say, x to $\cos(x)$, but is exactly equal to applying algebraic polynomials on Chebyshev grids.

4.1. Spectral collocation = maximum order finite difference. On a numerical grid of N points one can fit a polynomial with N degrees of freedom through all of the data. If this polynomial is algebraic and if the grid distribution is Chebyshev, i.e., $x_i = \cos(\frac{i\pi}{N})$, then one can build the Chebyshev collocation differentiation matrix. On the other hand, if the polynomial is trigonometric and if the grid is uniformly distributed, then one can build the Fourier spectral collocation differentiation matrix. One can, therefore, define in the physical space a spectral method to be a method that uses the maximum size polynomial for approximation and differentiation for a given grid size.

Another way to see this is to suppose one has a numerical grid of 16 points and a fourth-order difference operator on a 5-point stencil. Now reduce the number of grid points to 8. The difference operator is still fourth order. Now reduce the number of grid points to 5. The difference operator is still fourth-order accurate. This is spectral accuracy. That is, spectral accuracy of collocation methods on finite grids is $N - 1$ where N is the number of grid points.

4.1.1. A numerical check. Let us consider the above statements numerically. A Fourier collocation spectral method on a grid of nine points is a differencing mechanism with exactly nine degrees of freedom and hence is designed to differentiate nine

TABLE 4.1
Fourier spectral collocation applied to sin's.

Grid pts	sin freq	L_2 error
9	1.0	1.6710^{-28}
9	1.5	8.7110^{-1}
9	2.0	5.6810^{-29}
9	2.5	1.6010^0
9	3.0	2.2410^{-28}
9	3.5	2.7910^0
9	4.0	8.010^{-28}
9	4.5	2.9110^0
9	5.0	3.0510^0

TABLE 4.2
Chebyshev spectral collocation applied to polynomials.

Grid pts	Poly order	L_2 error
9	x^5	2.2910^{-25}
9	$x^{5.5}$	9.2610^{-4}
9	x^6	6.4910^{-25}
9	$x^{6.5}$	2.7710^{-3}
9	x^7	2.3910^{-24}
9	$x^{7.5}$	1.7910^{-2}
9	x^8	6.9610^{-24}
9	$x^{8.5}$	4.2910^{-1}
9	x^9	2.8310^0

functions exactly: 1 , $\cos(kx)$, and $\sin(kx)$ for $k = 1, 2, 3, 4$. Table 4.1 is meant to illustrate two points: i) the maximum frequency that is differentiated exactly is 4.0, and ii) noninteger frequencies perform poorly.

Likewise, a Chebyshev collocation spectral method on a grid of nine points is designed to differentiate nine functions exactly: x^k for $k = 0, \dots, 8$. Table 4.2 is designed to illustrate the same two points as Table 4.1. The table begins with the function x^5 .

Note that if the function being differentiated does not have an integer frequency for e^{ikx} or x^k , then, say for Chebyshev, differentiating $x^{5.5}$ gives no better or worse a result than differentiating $\sin(5.5x)$. The point is that the differentiation accuracy of spectral collocation methods on a finite grid of size N has accuracy $N - 1$, and one cannot expect that a wave-like pulse will be transmitted better with a Fourier spectral method than with a Chebyshev spectral method. The only important issue is the dimension of the space and the boundary conditions: use Chebyshev for nonperiodic boundary conditions and Fourier for periodic boundary conditions.

4.2. Very high order finite differencing. Now suppose one builds a series of algebraically generated finite difference operators of increasing accuracy and tests these difference operators on the function $\sin(2x)$. The grid size will be fixed at 33 points. The first two lines of Table 4.3 illustrate the effect of the Runge phenomenon; see [10]. The change in the error from periodic boundary conditions on a uniform grid to nonperiodic boundary conditions on a uniform grid is from 10^{-27} to 10^{-21} . In addition, note that applying an algebraic polynomial with periodic boundary condi-

TABLE 4.3
Finite difference accuracy approaching spectral accuracy.

Grid pts	Order of acc	L_2 error	Error ratio	Periodic BCs	Grid even or Cheby
33	32	1.5210^{-27}		yes	even
33	32	5.0710^{-21}		no	even
33	18	7.4910^{-17}		no	Cheby
33	20	1.1710^{-18}	64.0	no	Cheby
33	22	1.7310^{-20}	67.6	no	Cheby
33	24	2.4310^{-22}	71.2	no	Cheby
33	26	3.4910^{-24}	69.6	no	Cheby
33	28	5.7510^{-26}	60.7	no	Cheby
33	30	1.3010^{-27}	44.2	no	Cheby
33	32	8.8910^{-28}	1.46	no	Cheby

tions yields a result comparable to applying a trigonometric polynomial, i.e., a Fourier spectral collocation polynomial with periodic boundary conditions. Furthermore, one can observe the Runge phenomenon with trigonometric polynomials just as one observes it with algebraic polynomials, when the boundary conditions are not periodic. That is, the Runge phenomenon occurs due to equally spaced interpolation of high-order algebraic polynomials. This same phenomenon will occur if the interpolation is performed with trigonometric functions. Note that 128-bit arithmetic is being used. From line three to the bottom of the table the order of accuracy is increased from 18 until the maximum, i.e., spectral, accuracy of 32 is obtained. When one tests the accuracy of a finite difference operator one doubles the grid and sees the error decrease as $(\frac{1}{2})^n$, where n is the accuracy of the scheme. This comes from the truncation error, which will produce a factor of the form $(\Delta x)^n$. In Table 4.3, it is the number n that is being increased while Δx remains constant.

Compare Table 4.3 to Table 4.4, in which a Chebyshev collocation method on an increasing grid size is tested on $\sin(2x)$. Note that in Table 4.4, Δx is decreasing and n is increasing in the expression $(\Delta x)^n$ as one proceeds down the table. The final line of Table 4.4 is the Chebyshev method on a grid of 33 points, which produces a result comparable to the result in Table 4.3 on the same size grid. The numbers are not exactly the same because, first of all, all calculations are near machine accuracy, and second, the differencing coefficients are calculated in different ways.

4.3. Chebyshev spectral methods and cosine polynomials. This section will review Chebyshev spectral methods and their equivalency to cosine polynomials. Chebyshev approximation can be seen as approximation by algebraic polynomials

$$\begin{aligned}
 (4.1) \quad T_0(x) &= 1, \\
 T_1(x) &= x, \\
 T_2(x) &= 2x^2 - 1, \\
 T_3(x) &= 4x^3 - 3x,
 \end{aligned}$$

or as approximation by a cosine series (see [12])

$$(4.2) \quad T_n(x) = \cos(n \arccos x) = \cos(n\theta) = \sum_{q=0}^n a_q (\cos \theta)^q = \sum_{q=0}^n a_q x^q,$$

TABLE 4.4
Chebyshev spectral collocation of increasing order.

Grid pts	Alg err	Err ratio
9	3.0010^{-3}	
11	8.6010^{-5}	34.9
13	1.6510^{-6}	52.1
15	2.2910^{-8}	72.1
17	2.3810^{-10}	96.2
19	1.9410^{-12}	122.7
21	1.2710^{-14}	152.8
23	6.8510^{-17}	185.4
25	3.0810^{-19}	222.4
27	1.1710^{-21}	263.2
29	3.8510^{-24}	303.9
31	1.0910^{-26}	353.2
33	2.5410^{-27}	4.3

for some set $\{a_q\}$ and where $x = \cos(\theta)$. If one now chooses a numerical grid defined as $x_j = \cos(\frac{\pi j}{N})$, $j = 0, \dots, N$, then one obtains $T_n(x_j) = \cos(\frac{\pi j n}{N})$ and the pseudospectral Chebyshev method; see [15], [39], and [6].

A Chebyshev spectral method involves approximating a function, $f(x)$, by interpolating an algebraic polynomial to point values $f(x_i)$, where the grid points are given by the Chebyshev grid points $x_i = \cos(\theta_i)$. An equivalent process is to interpolate a cosine polynomial to the evenly spaced point values of the angle θ_i and to consider $f(x)$ to be evaluated on the uniform grid of the angle variable θ_i so that $f(x_i)$ becomes $f(\theta_i)$. That is (see [15]), if the Chebyshev series for $f(x)$ is

$$(4.3) \quad Pf(x) = g(x) = \sum_{k=0}^{\infty} a_k T_k(x),$$

then the expansion coefficients $\{a_k\}$ can be found in two equivalent ways:

$$(4.4) \quad a_k = \frac{2}{\pi c_k} \int_{-1}^1 f(x) T_k(x) (1-x^2)^{-1/2} dx = \frac{2}{\pi c_k} \int_0^\pi f(\cos \theta) \cos k\theta d\theta.$$

By this transformation of the independent variable one can perform Chebyshev spectral methods on a uniform grid or one can build arbitrarily high difference operators on a uniform grid which have stability characteristics equivalent to the usual Chebyshev spectral method.

4.4. High-order differencing on Chebyshev grids. Chebyshev spectral methods work very well for nonperiodic problems precisely because the truncation error for a Chebyshev polynomial is equal ripple.

As shown above, the truncation error for polynomial approximation, $p_n(x)$, of a function $f(x)$ is

$$(4.5) \quad f(x) - p_n(x) = \frac{(x-x_0)(x-x_1)\cdots(x-x_n)}{(n+1)!} f^{n+1}(\xi),$$

where ξ lies between the smallest and the largest x_i . If one wants to minimize the error due to the term

$$(x-x_0)(x-x_1)\cdots(x-x_n),$$

then the $n+1$ sample points should be chosen as the zeros of the Chebyshev polynomial $T_{n+1}(x)$; see [12]. This selection of grid points ensures that the error has the equal-ripple property that is characteristic of Chebyshev polynomials. A Chebyshev spectral method interpolates the highest order polynomial possible onto the $n+1$ degrees of freedom defined by the point values of a function at the $n+1$ zeros of $T_{n+1}(x)$. If the polynomial is of a lower order than n , then the Chebyshev grid is still a good grid based on numerical studies. It appears that the only feature of the Chebyshev grid that is of importance for very high order numerical schemes is the structure of the grid density at the ends of the interval. That is, the grid density gets finer and finer at a rate that keeps the oscillations associated with the Runge phenomenon under control. One hopes it can be proved that the Chebyshev grid will be the best grid even for these lower-order polynomials, but such a proof does not yet exist.

4.5. A note on ill-conditioned interpolation. All calculations in this paper are done with 128-bit calculations. This was chosen because the same level of calculation accuracy was used in [7]. High-order polynomial interpolation can be an ill-conditioned process, and one can, therefore, expect to encounter roundoff errors. In practice, it is more likely that one would use 64-bit arithmetic. The calculations conducted in this paper have also been done at this lower number of bits without observing roundoff errors but were reported here with 128 bits for comparison with other works, as noted.

Additionally, in this paper polynomials of extremely high order, 16 and 32, are tested to prove that there are no limits to the level of accuracy that one can obtain. In practical calculations, however, the author believes that one need not go higher than perhaps 8 or 10, and at these accuracies, with 64-bit arithmetic, one will not encounter a problem with roundoff error.

5. Wavelet-based grid and order selection. The previous section introduced the idea of building very high order algebraically generated difference operators on Chebyshev grids as a way of obtaining very high accuracy that is almost spectral in nature. This section will explore the idea of performing wavelet-based grid refinement on these Chebyshev grids as a way to obtain the necessary distribution near a boundary, while having the ability to refine the grid away from the boundary for proper physical-space function resolution.

5.1. A short review of wavelets. To define Daubechies-based wavelets, see [11] for the original work and [36] for an introduction to wavelet-based signal processing. Consider the two functions $\phi(x)$, the scaling function, and $\psi(x)$, the wavelet. The scaling function is the solution of the dilation equation

$$(5.1) \quad \phi(x) = \sqrt{2} \sum_{k=0}^{L-1} h_k \phi(2x - k),$$

where $\phi(x)$ is normalized: $\int_{-\infty}^{\infty} \phi(x) dx = 1$, and the wavelet $\psi(x)$ is defined in terms of the scaling function

$$(5.2) \quad \psi(x) = \sqrt{2} \sum_{k=0}^{L-1} g_k \phi(2x - k).$$

One builds an orthonormal basis from $\phi(x)$ and $\psi(x)$ by dilating and translating to get the functions

$$(5.3) \quad \phi_k^j(x) = 2^{-\frac{j}{2}} \phi(2^{-j}x - k)$$

and

$$(5.4) \quad \psi_k^j(x) = 2^{-\frac{j}{2}} \psi(2^{-j}x - k),$$

where $j, k \in Z$. j is the dilation parameter and k is the translation parameter. The coefficients $H = \{h_k\}_{k=0}^{L-1}$ and $G = \{g_k\}_{k=0}^{L-1}$ are related by $g_k = (-1)^k h_{L-k}$ for $k = 0, \dots, L-1$. All wavelet properties are specified through the parameters H and G . If one's data are defined on a continuous domain such as $f(x)$, where $x \in R$ is a real number, then one uses $\phi_k^j(x)$ and $\psi_k^j(x)$ to perform the wavelet analysis. If, on the other hand, one's data are defined on a discrete domain such as $f(i)$, where $i \in Z$ is an integer, then the data are analyzed, or filtered, with the coefficients H and G . In either case, the scaling function $\phi(x)$ and its defining coefficients H detect localized low frequency information, i.e., they are low-pass filters (LPF), and the wavelet $\psi(x)$ and its defining coefficients G detect localized high-frequency information, i.e., they are high-pass filters (HPF). Specifically, H and G are chosen so that dilations and translations of the wavelet, $\psi_k^j(x)$, form an orthonormal basis of $L^2(R)$, and so that $\psi(x)$ has M vanishing moments that determine the accuracy. In other words, $\psi_k^j(x)$ will satisfy

$$(5.5) \quad \delta_{kl} \delta_{jm} = \int_{-\infty}^{\infty} \psi_k^j(x) \psi_l^m(x) dx,$$

where δ_{kl} is the Kronecker delta function, and the accuracy is specified by requiring that $\psi(x) = \psi_0^0(x)$ satisfy

$$(5.6) \quad \int_{-\infty}^{\infty} \psi(x) x^m dx = 0,$$

for $m = 0, \dots, M-1$. Under the conditions of the previous two equations, for any function $f(x) \in L^2(R)$ there exists a set $\{d_{jk}\}$ such that

$$(5.7) \quad f(x) = \sum_{j \in Z} \sum_{k \in Z} d_{jk} \psi_k^j(x),$$

where

$$(5.8) \quad d_{jk} = \int_{-\infty}^{\infty} f(x) \psi_k^j(x) dx.$$

The two sets of coefficients H and G are known as quadrature mirror filters. For Daubechies wavelets the number of coefficients in H and G , or the length of the filters H and G , denoted by L , is related to the number of vanishing moments M by $2M = L$. For example, the famous Haar wavelet is found by defining H as $h_0 = h_1 = 1$. For this filter, H , the solution to the dilation equation (5.1), $\phi(x)$, is the box function given by $\phi(x) = 1$ for $x \in [0, 1]$ and $\phi(x) = 0$ otherwise. The Haar function is very useful as a learning tool, but because of its low order of approximation accuracy and lack of differentiability it is of limited use as a basis set. The coefficients H needed to define compactly supported wavelets with a higher degree of regularity can be found in [11]. As is expected, the regularity increases with the support of the wavelet. The usual notation to denote a Daubechies-based wavelet defined by coefficients H of length L is D_L .

It is usual to let the spaces spanned by $\phi_k^j(x)$ and $\psi_k^j(x)$ over the parameter k , with j fixed, be denoted by V_j and W_j respectively, so that

$$(5.9) \quad V_j = \text{span}_{k \in Z} \phi_k^j(x),$$

$$(5.10) \quad W_j = \text{span}_{k \in Z} \psi_k^j(x).$$

The spaces V_j and W_j are related by

$$(5.11) \quad \cdots \subset V_1 \subset V_0 \subset V_{-1} \subset \cdots$$

and

$$(5.12) \quad V_j = V_{j+1} \oplus W_{j+1},$$

where the notation $V_0 = V_1 \oplus W_1$ indicates that the vectors in V_1 are orthogonal to the vectors in W_1 and the space V_0 is simply decomposed into these two component subspaces.

The previously stated condition that the wavelets form an orthonormal basis of $L^2(R)$ can now be written as

$$(5.13) \quad L^2(R) = \bigoplus_{j \in Z} W_j.$$

Two final properties of the spaces V_j are that

$$(5.14) \quad \bigcap_{j \in Z} V_j = \{0\}$$

and

$$(5.15) \quad \overline{\bigcup_{j \in Z} V_j} = L^2(R).$$

5.2. Wavelet grid refinement on uniform grids. The idea of using wavelets to generate numerical grids began with the observation in [25] that the essence of an adaptive wavelet Galerkin method is nothing more than a finite difference method with grid refinement. So, instead of letting the magnitude of wavelet coefficients choose which basis functions to use in a Galerkin approach, let the same coefficients choose which grid points to use and then think of the wavelet method in a collocation sense.

In other words, suppose a calculation begins with N evenly spaced samples of a function \vec{f} and that some quadrature method produces N scaling function coefficients on the finest scale denoted by V_0 . If the spacing between adjacent values in the vector \vec{f} is Δx , then this is also the physical-space resolution of any calculation done in V_0 . Now, decompose V_0 once to get $V_0 = V_1 \oplus W_1$. Similarly speaking, the physical-space resolution of V_1 is $2\Delta x$ and the refinement from the $2\Delta x$ physical-space resolution to the Δx physical-space resolution is dictated by the wavelet coefficients in W_1 .

5.2.1. An algorithm for wavelet-based grid refinement. Given a vector of evenly spaced data \vec{s} , which will be considered the scaling function coefficients on the finest scale, find the scaling function and wavelet coefficients on the next coarsest scale:

$$(5.16) \quad s_k^j = \sum_{n=1}^{2M} h_n s_{n+2k-2}^{j-1}$$

and

$$(5.17) \quad d_k^j = \sum_{n=1}^{2M} g_n s_{n+2k-2}^{j-1}$$

(see above for notation definitions). One can continue this type of decomposition in order to obtain wavelet coefficients on a number of scales, $\vec{d}_1, \vec{d}_2, \dots$. This is all the information that is necessary in order to choose a numerical grid. That is, if d_k^j is the wavelet coefficient at scale j and location k , then a grid point, or two, is added at location k and scale j . For example, if coefficient $|d_5^2| > \epsilon$, where ϵ is a user-defined sensitivity threshold, then one can add a grid point at location x_{20} , since the wavelet coefficients at scale $j = 2$ represent local high frequencies in the physical space at scale $4\Delta x$, i.e., $5 * 4\Delta x = x_{20}$. x_i represents the numerical value of the i th grid point. Note that one can add a number of grid points in any region around large wavelet coefficients, and it is generally more efficient to do so. In addition, if one is calculating a moving wave structure, then the grid points can be added in front of the wave structure motion. The wave velocity can easily be estimated from the information obtained from wavelet coefficients at two different times.

Note that a Fortran software implementation of this grid refinement algorithm can be found in [30]. This software, and the algorithm defined above, can also be used to refine on Chebyshev grids as described in the next section.

5.3. Wavelet grid refinement on Chebyshev grids. Chebyshev grids are not evenly spaced in physical space, but are evenly spaced in angle. That is, a Chebyshev grid comes from $x_j = \cos(\theta_j)$, $j = 0, \dots, N$, where the angle $\theta_j = \frac{\pi j}{N}$ is evenly spaced. The refinement mechanism described above can now be applied to the uniform angle grid point values to define a new numerical grid. That is, all the above grid refinement machinery can be applied to Chebyshev grids where each subspace V_j will coincide with a uniform angle or usual Chebyshev grid, and each refinement subspace W_j will coincide with additional points being added to the usual Chebyshev grid. It is well known that the Chebyshev grid is the best grid, in terms of minimal error, for algebraic polynomial interpolation. A refinement, W_1 , on a Chebyshev grid, V_1 , to get $V_0 = V_1 \oplus W_1$ is designed to begin with a grid which in one sense is perfect, and to perturb from this grid.

5.4. Wavelet filter order detection. Given a wavelet with m vanishing moments it is straightforward to show that (see [36])

$$(5.18) \quad |d_{jk}| = \left| \int f(x) \psi_k^j(x) dx \right| \leq C 2^{-jm} \|f^m(x)\|.$$

This equation gives all the information that is necessary in order to estimate the local polynomial order of data in a computational domain, and hence fit the numerical

method accordingly. That is, for a fixed scale $j = \text{constant}$, if $f(x) = x^p$ then the coefficients d_{jk} will behave as x^{p-m} with respect to the translation parameter.

This is most easily understood by observing an application of the Haar wavelet to a polynomial. Consider the samples of x^2 ,

$$1, 4, 9, 16, 25, 36, 49, 64,$$

and apply the Haar wavelet filter, 1, -1 to get

$$d_1 = -3, d_2 = -7, d_3 = -11, d_4 = -15.$$

These four numbers represent the wavelet transform of a quadratic but they lie on a line. Now, suppose that our computational domain contains data which are roughly a line, x , in the left half of the domain and a quadratic, x^2 , in the right half of the domain. If we apply Haar to the line data,

$$1, 2, 3, 4, 5, 6, 7, 8,$$

we get

$$-1, -1, -1, -1,$$

and if we apply Haar a second time to these data we get

$$0, 0.$$

That is, two applications of Haar yield the zero vector when applied to linear data. Likewise, apply Haar three times to quadratic data to get the zero vector. In this manner one can detect polynomial structure in the computational domain.

In general, if W is a matrix of a discrete wavelet transform with m vanishing moments, and \vec{f} is a vector of the samples of a polynomial x^p , then $\vec{d} = W\vec{f}$ will be the samples of polynomial cx^{p-m} for some constant c . A second application of an appropriately sized W reduces the order of the polynomial again to $\vec{d}_2 = W\vec{d}_1$, and \vec{d}_2 will be the samples of the polynomial c_2x^{p-2m} for some constant c_2 . This type of mechanism of high-pass filtering data that have already been high-pass filtered is similar to wavelet packets; see [14] and [36]. Based on the data from this high-pass filter analysis, one can obtain an estimate of the local polynomial order of the computational data and thereby choose a numerical method appropriately.

Note that for a given wavelet, the order of the approximation m is fixed. There is a variety of ways that one can use wavelets to detect order and grids. For example, one can perform wavelet analysis with wavelets of different orders, thereby detecting polynomials of comparable orders. WOFD and WOFD2 use a different approach; generally only the D_4 is used. D_4 can approximate 1 and x exactly and, therefore, has a truncation error behaving as x^2 . In short, experience has shown that D_4 is the best choice in the sense that the grid selected is always very good, and work involved in the wavelet decomposition is small.

5.5. Combining wavelet grid and order selection. Let us use the notation introduced earlier in the section for denoting the wavelet low-pass filter and high-pass filter, H and G , respectively. A wavelet decomposition of a vector of data \vec{f} is accomplished by applying H and G to \vec{f} . Let us not worry about denoting matrix size, and simply note that one application of H or G yields a vector half as long. A

wavelet decomposition yields the vectors $H\vec{f}$, $G\vec{f}$, $GH\vec{f}$, $GHH\vec{f}$, etc. This provides the information needed for grid refinement, as described above. A repeated high-pass filter decomposition yields the additional vectors $GG\vec{f}$, $GGG\vec{f}$, etc.

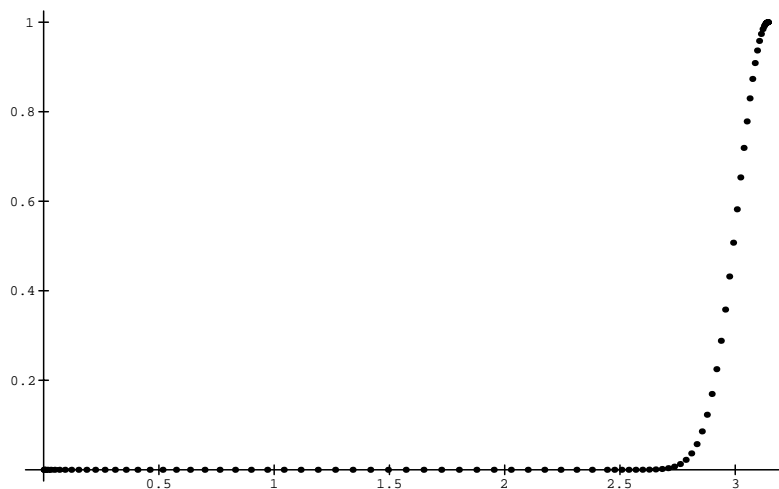
This combination of wavelet filtering and repeated high-pass filtering yields sufficient information for grid and order selection. In practice, one would not want to vary the order at every grid point. The overhead would be more expensive than simply performing a uniform grid calculation. One would, however, like to choose the order to be perhaps 2 or 4 in a “rough” region of the domain, and perhaps 8 or 10 in a “smooth” region of the domain. Within each of these regions one can then perform grid refinement. This type of regional selection of grids and orders should work well in a parallel environment.

6. A new numerical method: WOFD2. This section formally introduces WOFD2. As mentioned above, WOFD2 is an extension of the WOFD idea, but with essentially spectral accuracy. WOFD2 uses wavelets to choose not only a numerical grid but also the order of the difference operator used on this grid. In addition, WOFD2 uses very high order finite difference operators on the order of 8, 16, or even 32. Furthermore, the physical-space grids are no longer evenly spaced at every resolution but are Chebyshev. That is, wavelet-based grid generation (see [30]) requires that a grid be selected from a uniform finest grid. But, high-order polynomials can be highly oscillatory on uniform grids. Therefore, WOFD2 works with Chebyshev grids at each resolution level. Recall that Chebyshev grids $x_i = \cos(\theta_i)$ are not uniform in the physical-space variable x_i but are uniform in the angle variable θ_i . It is in this uniform angle variable θ_i that grid refinement is performed.

Wavelets are very good at finding regions of the domain at which a large numerical error is likely to occur. Numerical error will be determined by the truncation error of a polynomial that is locally interpolated to the data. The truncation error will be the product of intervals and a constant. Imagine the intervals are all a multiple of a smallest interval Δx . Then the key component in the truncation error will be $(\Delta x)^n$. This component can be decreased either by decreasing the size of Δx or by increasing the order of the scheme, i.e., increasing n . Or, one can decrease Δx and increase n simultaneously. The optimal choice of Δx and n is the subject of an ongoing study; see [16] for a similar, but nonwavelet, study applied to spectral multidomain techniques.

6.1. Very high-order boundary conditions. In the above definition of WOFD2 it is noted that the order of accuracy can be as high as 16, 20, or 32. The key to implementing boundary conditions that are of the same order is the Chebyshev grid. As noted, the Chebyshev grid is necessary to control the Runge phenomenon and hence keep high-order polynomials from oscillating with large amplitudes away from the center of the numerical stencil. Note that all nonperiodic boundary conditions in this paper are imposed on biased stencils of compact support.

The next concern is what effect the Chebyshev grid has on the CFL number. Note the grid density in Figure 6.4. This figure shows a Chebyshev grid at two different densities. Away from the center of the domain, a coarse Chebyshev grid is used, and in the center of the domain a more fine Chebyshev grid is used. Note that this figure illustrates the grid density for a variable density Chebyshev grid, but the region near the boundary is a bit wide from the perspective of simply keeping the “tail” of the interpolation polynomial from oscillating. In other words, if the desired boundary accuracy is 16, then a polynomial through the 17 points closest to the boundary generates the differencing coefficients. It is these 17 points that must have the “Chebyshev structure.” On the 18th point from the boundary the grid density

FIG. 6.1. *Initial condition of pulse entering domain.*

can become finer, perhaps a Chebyshev grid at a finer scale or even a uniformly spaced grid that has a grid density somewhere between the finest Δx and coarsest Δx of the Chebyshev grid at the boundary. This type of variable density Chebyshev grid can provide the “Chebyshev structure,” which controls the Runge phenomenon without the severe limitations on the CFL number. The smallest Δx will generally occur near the boundary, but now it will be of the same order as the smallest Δx necessary to resolve the physical structure of the flow features.

6.2. Comparison with *hp*-refinement. In the finite element literature (see [1], [18]), the idea of refining the grid and increasing the polynomial order is known as *hp*-refinement. The theory of *hp*-refinement can certainly be applied to the new method WOFD2, even though WOFD2 works with polynomials only for the generation of finite difference operators to be applied in the physical space. One of the most important results from *hp*-refinement theory from which WOFD2 can benefit is that when the function being differentiated is smooth, the rate of convergence is controlled by the polynomial degree. For the purpose of pulse propagation in aeroacoustics it is apparent that a high-order differentiation, i.e., high-order polynomial interpolation, will propagate the pulse more faithfully than grid refinement on the same pulse, assuming the pulse is smooth.

6.3. Numerical experiments with WOFD2. This section will provide the results from numerous numerical experiments performed with WOFD2. For all the numerical experiments in this section of the paper a Gaussian pulse enters the domain from the right-hand side and travels to the left. The governing equation is the one-dimensional hyperbolic wave equation

$$(6.1) \quad U_t(x, t) = U_x(x, t), \quad U(x, 0) = e^{-c(x-\pi)^2}$$

for some constant c . See Figure 6.1 for a plot of this initial condition. Note that the domain extends from 0 to π . The final time for all simulation is $\pi/2$. The simulation is stopped at this value because at $\pi/2$ the Chebyshev grid has a maximum spacing between grid points and hence a minimum resolution.

TABLE 6.1
WOFD2 of accuracies 8, 16, 32, and 48.

Grid pts	Order of acc	L_2 error	L_∞ error	Final time
64	8	$3.82 * 10^{-3}$	$1.78 * 10^{-2}$	$\pi/2$
64	16	$2.95 * 10^{-4}$	$1.30 * 10^{-3}$	$\pi/2$
64	32	$1.74 * 10^{-5}$	$7.10 * 10^{-5}$	$\pi/2$
64	48	$2.74 * 10^{-6}$	$1.28 * 10^{-5}$	$\pi/2$

TABLE 6.2
WOFD2 adaptive grid, but accuracy fixed at 8.

Grid pts	t_f grid	Order of acc	L_2 error	L_∞ error	Thresh	Final time
128/64	65	8	$3.82 * 10^{-3}$	$1.78 * 10^{-2}$	100.0	$\pi/2$
128/64	77	8	$1.67 * 10^{-3}$	$8.66 * 10^{-3}$	10^{-3}	$\pi/2$
128/64	79	8	$1.63 * 10^{-4}$	$8.76 * 10^{-4}$	10^{-4}	$\pi/2$
128/64	82	8	$8.24 * 10^{-5}$	$3.24 * 10^{-4}$	10^{-5}	$\pi/2$
128/64	84	8	$8.07 * 10^{-5}$	$3.24 * 10^{-4}$	10^{-6}	$\pi/2$
128	129	8	$6.51 * 10^{-5}$	$3.24 * 10^{-4}$	0.0	$\pi/2$

6.3.1. No adaptation. First we consider the case of very high order finite differencing on a Chebyshev grid. The grid size is kept fixed at 64 points, and the order is increased from 8 to 48. The errors decrease in a nice and uniform manner. No unusual numerical oscillations occur (see Table 6.1).

6.3.2. Adapting grid only: Order 8 spatial differencing. In this section the order of the spatial differencing is kept fixed at 8. No results are found for threshold values of 10^{-1} and 10^{-2} . This is because it seems to be a characteristic of adaptive methods that a very rough threshold value can degrade the performance of the method. It is better to start with threshold values less than or equal to 10^{-3} . The first row of Table 6.2 is the worst possible performance where no refinement is done and the grid is 64 points, and the last row of the table is the best possible performance where the grid is 128 points. Note that the software is constructed to work with both periodic and nonperiodic boundary conditions, so that when the boundary conditions are nonperiodic the possible number of points becomes $2^N + 1$, which includes the right-hand boundary point. For periodic boundary conditions the number of grid points is 2^N , since the right-hand boundary point is equal to the first point on the left-hand boundary.

6.3.3. Adapting grid only: Order 16 spatial differencing. Much of what was said for the 8th-order table above can be said here. The second row of Table 6.3 shows how the performance can be slightly degraded for relatively large threshold values. In this case the degradation occurs at the threshold value of 10^{-3} . This is a minor point. Generally speaking, start with a smaller threshold value.

6.3.4. Adapting the grid and order. Table 6.4 is the culmination of the paper and an example of WOFD2 with all options in use. The grid is adjusted between a maximum density of 128 and a minimum density of 64. The order of accuracy is adjusted between a maximum of 16 and a minimum of 8. The error converges in a nice manner toward the minimum error, which occurs at the maximum grid density of 128 and the maximum order of accuracy of 16.

TABLE 6.3
WOFD2 adaptive grid, but accuracy fixed at 16.

Grid pts	t_f grid	Order of acc	L_2 error	L_∞ error	Thresh	Final time
128/64	65	16	$2.95 * 10^{-4}$	$1.30 * 10^{-3}$	100.0	$\pi/2$
128/64	81	16	$1.57 * 10^{-3}$	$7.31 * 10^{-3}$	10^{-3}	$\pi/2$
128/64	87	16	$2.34 * 10^{-4}$	$8.16 * 10^{-4}$	10^{-4}	$\pi/2$
128/64	89	16	$2.43 * 10^{-5}$	$8.09 * 10^{-5}$	10^{-5}	$\pi/2$
128/64	87	16	$2.36 * 10^{-6}$	$9.20 * 10^{-6}$	10^{-6}	$\pi/2$
128/64	91	16	$3.18 * 10^{-7}$	$1.09 * 10^{-6}$	10^{-7}	$\pi/2$
128/64	93	16	$1.05 * 10^{-7}$	$4.66 * 10^{-7}$	10^{-8}	$\pi/2$
128	129	16	$4.26 * 10^{-8}$	$2.24 * 10^{-7}$	0.0	$\pi/2$

TABLE 6.4
WOFD2 with grid and order adaptation.

Grid density	t_f grid	Order of acc	L_2 error	L_∞ error	Thresh	Final time
64	65	8	$3.82 * 10^{-3}$	$1.78 * 10^{-2}$	100.0	$\pi/2$
128/64	81	16/8	$1.61 * 10^{-3}$	$7.20 * 10^{-3}$	10^{-3}	$\pi/2$
128/64	80	16/8	$4.42 * 10^{-5}$	$2.68 * 10^{-4}$	10^{-4}	$\pi/2$
128/64	82	16/8	$6.28 * 10^{-6}$	$3.74 * 10^{-5}$	10^{-5}	$\pi/2$
128/64	84	16/8	$4.50 * 10^{-7}$	$2.64 * 10^{-6}$	10^{-6}	$\pi/2$
128/64	86	16/8	$1.23 * 10^{-7}$	$6.49 * 10^{-7}$	10^{-7}	$\pi/2$
128/64	87	16/8	$5.52 * 10^{-8}$	$2.25 * 10^{-7}$	10^{-8}	$\pi/2$
128/64	90	16/8	$5.12 * 10^{-8}$	$2.24 * 10^{-7}$	10^{-9}	$\pi/2$
128	129	16	$4.26 * 10^{-8}$	$2.24 * 10^{-7}$	0.0	$\pi/2$

The usual Chebyshev grid is evenly spaced in angle $\theta_i = i\pi/N$ for $i = 0, \dots, N$. In the physical space the grid distribution is $x_i = \cos(\theta_i)$, which is shaped like a semicircle. When one applies the wavelet grid adaptation to this evenly spaced θ_i , then one obtains in the physical space the distribution $x_i = \cos(\theta_i)$ in the portion of the domain away from the pulse, and the twice-as-dense grid distribution $x_i = \cos(i\pi/(2N))$ in the portion of the domain near the pulse. Note that the grid is the usual Chebyshev grid near the boundary. It is only safely away from the boundary that the grid density makes an abrupt change in density. See Figure 6.2 for an example of an initial grid.

If the numerical scheme is working properly then the pulse will propagate to the middle of the domain and be similar in shape to the initial condition. The best measure of this similarity is the L_∞ error. At the final time the pulse will appear as in Figure 6.3.

The Chebyshev grid is naturally more dense near the boundaries than in the middle of the domain. With the wavelet adaptation of this Chebyshev grid, the grid points can be kept dense while maintaining a Chebyshev distribution throughout most of the domain. Again, the most important region of the domain for a Chebyshev distribution is near the boundary. See Figure 6.4 for the grid distribution at the final time when the pulse has reached the middle of the domain.

Without grid refinement or order refinement the peak numerical error at the final time should be near the peak of the pulse, since it is this portion of the function that is most difficult to represent by polynomial interpolation. See Figure 6.5 for an example of such an error.

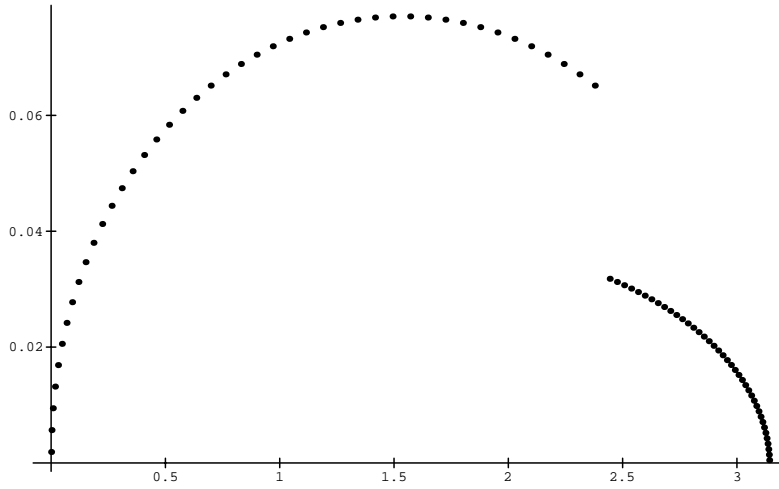


FIG. 6.2. Initial grid density for pulse entering domain.

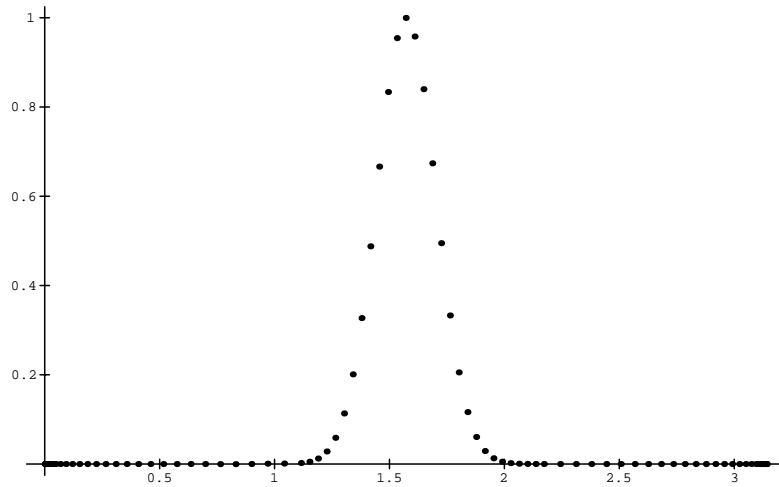
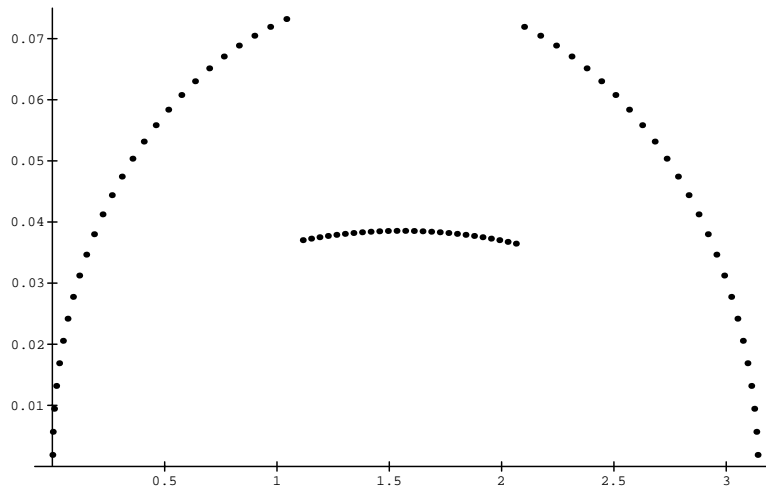
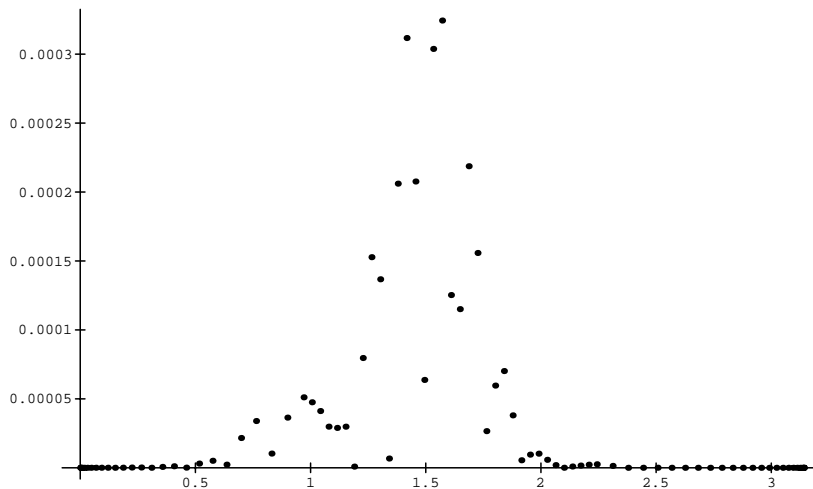


FIG. 6.3. Pulse at final time.

If the wavelet refinement threshold is not sufficiently low then one will see the peak error appear near a region of the domain where there is a grid or stencil discontinuity. A sufficiently low refinement threshold will be on the order of the L_∞ error when no grid or order refinement is executed. In Figure 6.6 noise is amplified at the interface where both the stencil and grid are refined. If the wavelet refinement threshold is adjusted to a smaller value then one can obtain an error profile similar to that in Figure 6.5.

When both the stencil and grid are changed throughout the calculation, one finds a relatively wide stencil near the peak value of the pulse. For the example of a 17-point stencil with accuracy of 16 at the pulse and a 9-point, accuracy 8 stencil away from the pulse see Figure 6.7.

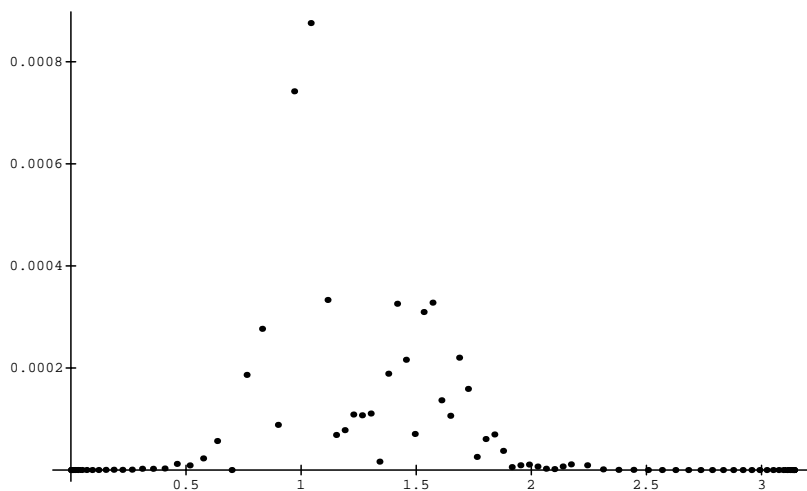
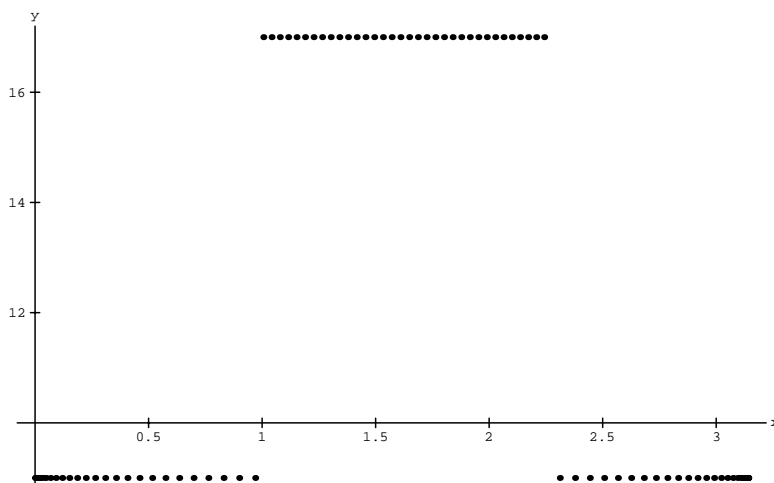
6.4. Two-dimensional examples. Thorough tests of WOFD2 in two dimensions have not yet been conducted, but one can see the grid structure of an adaptive

FIG. 6.4. *Adaptive Chebyshev grid at final time.*FIG. 6.5. *Typical error at pulse peak.*

two-dimensional Chebyshev grid in Figures 6.8 and 6.9. The first figure shows a Gaussian pulse entering the domain from one of the corners, and the second figure shows the corresponding grid. Initial tests on combustion and computational aeroacoustics are promising and will be reported in the future.

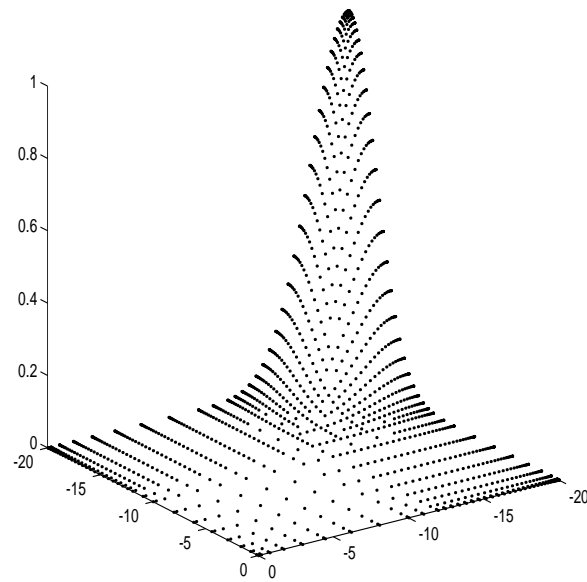
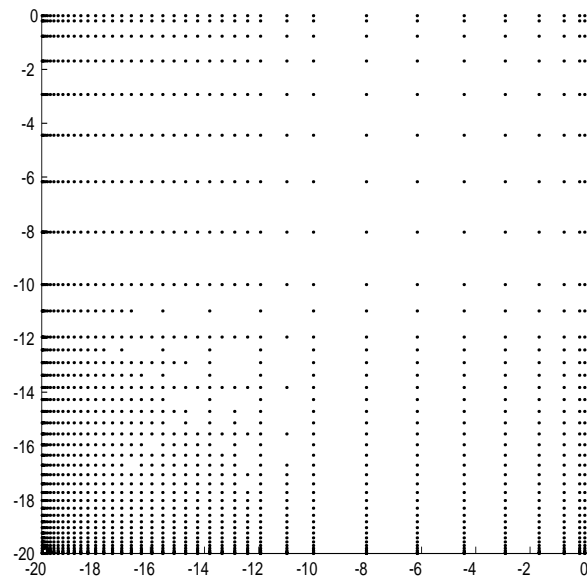
7. Comments on stability of WOFD and WOFD2. For numerical methods such as WOFD and WOFD2 where the computational data and the numerical method are time dependent, traditional stability analysis does not apply. However, the numerical evidence of stability is very strong. A catalog of stable test cases is as follows (some are documented and some are not):

- WOFD tested on $U_t = U_x$ in one dimension with periodic and nonperiodic boundary conditions,
- WOFD tested on $U_t = U_x + U_y$ in two dimensions with periodic and nonperiodic boundary conditions,


 FIG. 6.6. *Typical error at stencil discontinuity.*

 FIG. 6.7. *Width of differencing stencil at final time.*

- WOFD tested on $U_t = (U^2)_x + \epsilon U_{xx}$ in one dimension with periodic and nonperiodic boundary conditions (see [24] and [28]),
- WOFD tested on combustion for the resolution and identification of the velocity of an oscillating flame front in one dimension (see [29]),
- WOFD2 tested in the previous section in $U_t = U_x$ for nonperiodic boundary conditions. The easier case of periodic boundary conditions was, of course, also tested and found to be stable.
- WOFD and WOFD2 are currently undergoing numerical tests on the application to the oscillating flame front problem in two dimensions. This research will be included in a future report.

Of all the above tests, the most convincing evidence of stability occurs with the tests of combustion. In these test cases as many as 500,000 time steps have been taken with as many as 5000 grid updates, and the solution conforms to the theory; see [29].

FIG. 6.8. *Pulse on two-dimensional adaptive Chebyshev grid.*FIG. 6.9. *Two-dimensional grid for adaptive Chebyshev method.*

Numerical instability has yet to be observed in multiple test cases, which began in 1991.

Given that one cannot predict the numerical data or the numerical operators a priori, a proof of stability would have to proceed along the lines of a statistical argument and is a topic of current research.

8. Conclusion.

8.1. A summary of WOFD and WOFD2 features. Let us conclude with a brief summary of the key features of both WOFD and WOFD2.

- *Accuracy.* WOFD is fourth order and the order of WOFD2 varies depending on the computational data, but is generally as high as 8 or 10.
- *Stability.* WOFD and WOFD2 are stable in practice. In fact, in applications to combustion WOFD has been run for as many as 500,000 time steps in one and two dimensions. WOFD2 is newer and not as thoroughly tested, but to date has been stable in practice. A proof of stability is desirable and is the subject of current research.
- *Boundary conditions.* Boundary conditions for both WOFD and WOFD2 are imposed as they would be for finite difference methods. This allows us to draw on the considerable existing technology on finite difference boundary conditions.
- *Errors.* The errors for both WOFD and WOFD2 are on the order of the wavelet refinement parameter and are essentially uniform throughout the domain.
- *Memory usage.* For WOFD, the memory usage will be on the order of the number of grid points on the compressed grid. That is, computational variables are stored at each point on the compressed grid. Furthermore, operators such as differentiation are stored where the grid changes density. That is, the majority of the domain of the grid is uniform at some density. In these regions the operator is simply the uniform version of the fourth order operator. In regions where the grid density changes, the operator is stored. This requires a very small amount of additional storage. For WOFD2, the differentiation operators are stored everywhere. This leads to additional memory needs, but is acceptable for some problems, and efforts to reduce this requirement are underway.
- *Treatment of nonlinearities.* All equations are treated pointwise, therefore, nonlinearities are treated trivially.

This paper has covered many topics related to the construction of a very high order adaptive order and adaptive grid numerical method WOFD2. First it was necessary to review the closely related method WOFD, which never appeared as a journal article. Next, it was necessary to explore the various ways in which difference operators can be constructed. This included a comparison of difference operators generated from algebraic, trigonometric, exponential, and cosine polynomials. Next, which type of polynomial would be best for the construction of very high order numerical differencing was explored. The conclusion, not surprisingly, is that one should use algebraic polynomials on Chebyshev grids. The next step was to apply wavelet grid and order adaptation in order to be able to reduce errors throughout the domain by either increasing the order of the numerical method or by increasing the grid density in the appropriate region. The results of the numerical tests were very positive and it appears that WOFD2 will be applicable to a large range of numerical problems. The version of WOFD2 which has been presented here has been tweaked very little. That is, it worked essentially the first time it was tried. This is encouraging because most high-order spectrally accurate numerical methods require some kind of filtering or other refinement. Note that in the numerical testing conducted here, only one level of refinement was performed, but still the work was reduced by about a factor of two without increasing the L_∞ error. More levels of refinement are possible with further work reductions.

REFERENCES

- [1] B. SZABO AND I. BABUSKA, *Finite Element Analysis*, Wiley-Interscience, New York, 1991.
- [2] E. BACRY, S. MALLAT, AND G. PAPANICOLAOU, *A wavelet based space-time adaptive numerical method for partial differential equations*, Math. Model. Numer. Anal., 26 (1992), pp. 703–834.
- [3] S. BERTOLUZZA, G. NALDI, AND J. C. RAVEL, *Wavelet methods for the numerical solution of boundary value problems on the interval*, in Wavelets: Theory, Algorithms and Applications, C. Chui, L. Montefusco, and L. Puccio, eds., Academic Press, New York, 1994, pp. 425–448.
- [4] G. BEYLKIN, R. COIFMAN, AND V. ROKHLIN, *Fast wavelet transforms and numerical algorithms I*, Comm. Pure Appl. Math., 64 (1991), pp. 141–184.
- [5] W. CAI AND J. Z. WANG, *Adaptive wavelet collocation methods for initial value boundary problems of nonlinear PDE*, SIAM J. Numer. Anal., 33 (1996), pp. 937–970.
- [6] C. CANUTO, M. Y. HUSSAINI, A. QUARTERONI, AND T.A. ZANG, *Spectral Methods in Fluid Dynamics*, Springer-Verlag, New York, 1988.
- [7] M. CARPENTER AND D. GOTTLIEB, *Spectral Methods on Arbitrary Grids*, ICASE Report No. 95-37, NASA CR-198158, 1995.
- [8] S. N. CHRISTOFI, *The Study of Building Blocks for Essentially Non-oscillatory (ENO) Schemes*, Ph.D. thesis, Division of Applied Mathematics, Brown University, Providence, RI, 1996.
- [9] C. CHUI, *Wavelets: A Tutorial in Theory and Applications*, Vol. 2, Academic Press, New York, 1992, pp. 217–236.
- [10] G. DAHLQUIST AND A. BJÖRCK, *Numerical Methods*, Prentice-Hall, Englewood Cliffs, NJ, 1974.
- [11] I. DAUBECHIES, *Orthonormal basis of compactly supported wavelets*, Comm. Pure Appl. Math., 41 (1988), pp. 909–996.
- [12] P. J. DAVIS, *Interpolation and Approximation*, Dover, New York, 1975.
- [13] W. S. DON AND C. B. QUILEN, *Numerical simulation of shock-cylinder interactions*, J. Comput. Phys., 122 (1995), pp. 244–265.
- [14] G. ERLEBACHER, M. Y. HUSSAINI, AND L. JAMESON, *Wavelets: Theory and Applications*, Oxford University Press, London, 1996.
- [15] D. GOTTLIEB AND S. A. ORSZAG, *Numerical Analysis of Spectral Methods: Theory and Applications*, SIAM, Philadelphia, 1977.
- [16] D. GOTTLIEB AND C. E. WASBERG, *Optimal strategy in domain decomposition spectral methods for wave-like phenomena*, in Proc. Multidomain Methods, Bergen, Norway, 1996.
- [17] D. GOTTLIEB, B. GUSTAFSSON, P. OLSSON, AND B. STRAND, *On the superconvergence of Galerkin methods for hyperbolic IBVP*, SIAM J. Numer. Anal., 33 (1996), pp. 1778–1796.
- [18] W. GUI AND I. BABUSKA, *The h -, p - and hp -versions of the finite element method in one dimension. Part I: The error analysis of the p -version. Part II: The error analysis of the h and hp -versions. Part III: The adaptive hp -version*, Numer. Math., 49 (1986), pp. 577–612, 613–657, 659–683.
- [19] B. ENGQUIST, S. OSHER, AND S. ZHONG, *Fast Wavelet Algorithms for Linear Evolution Equations*, ICASE Report 92-14, 1992.
- [20] A. HARTEN, *Multiresolution Analysis for ENO Schemes*, ICASE Report 91-77, NASA Contractor Report 189546, 1991.
- [21] A. HARTEN, *Multiresolution Representation and Numerical Algorithms: A Brief Review*, ICASE Report 94-59, NASA Contractor Report 194949, 1994.
- [22] M. HOLMSTRÖM, *Solving Hyperbolic PDE's Using Interpolating Wavelets*, Report 189/1996, Department of Scientific Computing, Uppsala University, Uppsala, Sweden, 1996.
- [23] M. HOLMSTRÖM AND J. WALDÉN, *Adaptive wavelet methods for hyperbolic PDEs*, J. Sci. Comput., to appear.
- [24] L. JAMESON, *Wavelets and Numerical Methods*, Ph.D. thesis, Division of Applied Mathematics, Brown University, Providence, RI, 1993.
- [25] L. JAMESON, *On the wavelet based differentiation matrix*, J. Sci. Comput., 8 (1993), pp. 267–305.
- [26] L. JAMESON, *On the spline-based wavelet differentiation matrix*, Appl. Numer. Math., 17 (1995), pp. 33–45.
- [27] L. JAMESON, *On the differentiation matrix for Daubechies-based wavelets on an interval*, SIAM J. Sci. Comput., 17 (1996), pp. 498–516.
- [28] L. JAMESON, *On the Wavelet-Optimized Finite Difference Method*, ICASE Report 94-9, NASA CR-191601, 1994.
- [29] L. JAMESON, T. L. JACKSON, AND D. G. LASSEIGNE, *Wavelets as a numerical tool*, in Proc. Joint US-Japan Workshop on Combustion, J. Buckmaster and T. Takeno, eds., Springer-Verlag, New York, 1993.

- [30] L. JAMESON, *Wavelet-based grid generation*, Appl. Numer. Anal., to appear.
- [31] G. E. KARNIADAKIS AND S. A. ORSZAG, *Nodes, modes and flow codes*, Physics Today, 46 (1993), pp. 34–42.
- [32] H. O. KREISS, Technical report, Uppsala University, Sweden, 1978.
- [33] J. LIANDRAT AND P. TCHAMITCHIAN, *Resolution of the 1D Regularized Burgers Equation Using a Spatial Wavelet Approximation Algorithm and Numerical Results*, ICASE Report 90-83, 1990.
- [34] D. G. LASSEIGNE, T. L. JACKSON, AND C. E. GROSCH, *Stability of Freely Propagating Flames Revisited*, in preparation.
- [35] Y. MEYER, *Ondelettes et Operators*, Hermann, Paris, 1990.
- [36] G. STRANG AND T. NGUYEN, *Wavelets and Filter Banks*, Wellesley-Cambridge Press, Wellesley, MA, 1996.
- [37] B. SWARTZ AND B. WENDROFF, *The relation between the Galerkin and collocation methods using smooth splines*, SIAM J. Numer. Anal., 11 (1974), pp. 994–996.
- [38] O. V. VASILYEV AND S. PAOLUCCI, *A dynamically adaptive multilevel wavelet collocation method for solving partial differential equations in a finite domain*, J. Comput. Phys., 125 (1996), pp. 498–512.
- [39] R. G. VOIGT, D. GOTTLIEB, AND Y. HUSSAINI, *Spectral Methods for Partial Differential Equations*, SIAM, Philadelphia, PA, 1984.
- [40] J. WALDÉN, *Wavelet Solvers for Hyperbolic PDE's*, Acta Univ. Ups. Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology 240, Uppsala, 1996.
- [41] J. WALDÉN, *Orthonormal compactly supported wavelets for solving hyperbolic PDEs*, Report No. 170, Department of Scientific Computing, Uppsala University, Sweden, 1995.

Copyright of SIAM Journal on Scientific Computing is the property of Society for Industrial and Applied Mathematics and its content may not be copied or emailed to multiple sites or posted to a listserv without the copyright holder's express written permission. However, users may print, download, or email articles for individual use.

Copyright of SIAM Journal on Scientific Computing is the property of Society for Industrial and Applied Mathematics and its content may not be copied or emailed to multiple sites or posted to a listserv without the copyright holder's express written permission. However, users may print, download, or email articles for individual use.