Available online at www.sciencedirect.com
SCIENCE DIRECT®
Wave Motion 41 (2005) 263–291
WAVE MOTION
www.elsevier.com/locate/wavemoti

Wave propagation using bases for bandlimited functions

G. Beylkin, K. Sandberg*

Department of Applied Mathematics, University of Colorado at Boulder, 526 UCB, Boulder, CO 80309-0526, USA

Received 7 January 2003; received in revised form 7 May 2004; accepted 10 May 2004

Available online 23 September 2004

Abstract

We develop a two-dimensional solver for the acoustic wave equation with spatially varying coefficients. In what is a new approach, we use a basis of approximate prolate spheroidal wavefunctions and construct derivative operators that incorporate boundary and interface conditions. Writing the wave equation as a first-order system, we evolve the equation in time using the matrix exponential. Computation of the matrix exponential requires efficient representation of operators in two dimensions and for this purpose we use short sums of one-dimensional operators. We also use a partitioned low-rank representation in one dimension to further speed up the algorithm. We demonstrate that the method significantly reduces numerical dispersion and computational time when compared with a fourth-order finite difference scheme in space and an explicit fourth-order Runge–Kutta solver in time.

Keywords: Wave propagation; Prolate spheroidal wave functions; Bandlimited functions; Efficient operator representations; Matrix exponential; Spectral projectors; Acoustic wave equation

1. Introduction

In this paper we demonstrate how to use bases for bandlimited functions in algorithms of wave propagation. Using bandlimited functions allows us to achieve a low sampling rate while significantly reducing numerical dispersion. In addition, we show how to compute and use the matrix exponential as a propagator by employing separated and partitioned low rank representations.

Using bases for bandlimited functions is a significant departure from the usual approach in numerical analysis. For example, the standard notion of the order of approximation is not appropriate in its usual form since in our construction the basis itself is generated for a finite but arbitrary accuracy. We note that the methods we describe in this paper are applicable to many other problems.

* Corresponding author. Tel.: +1 303 492 0593; fax: +1 303 492 4066.
E-mail address: kristian.sandberg@colorado.edu (K. Sandberg).

0165-2125/$ – see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.wavemoti.2004.05.008

The first step in constructing a numerical scheme is to select a basis for representing solutions and operators. Typically, in spectral and pseudo-spectral methods, the trigonometric functions $\{e^{ik\pi x}\}_{k=0}^{N}$ have been used for periodic, and Legendre and/or Chebyshev polynomials for non-periodic problems. Instead, we consider bandlimited functions on an interval. A basis for bandlimited functions, the prolate spheroidal wave functions (PSWFs), was introduced in the 1960s by Slepian et al. in a series of papers [1–5]. Recently the generalized Gaussian quadratures became available in [6,7], making it possible to construct efficient numerical algorithms for such functions.

We review the construction of three bases for bandlimited functions. First we consider bases $\{e^{ic\theta_k x}\}_{k=1}^{N}$ on the interval $[-1, 1]$, where $|\theta_k| < 1$ are the nodes of the generalized Gaussian quadrature constructed for a given precision and bandlimit. We note that these functions are not necessarily periodic. Such bases may not be suitable for some numerical computations (heuristically, they correspond to the basis of monomials). For this reason, we also consider bases of approximate PSWFs and interpolating bases and use them in our computations.

There are at least two deficiencies of orthogonal polynomials in using them for numerical computations. First is the concentration of Gaussian nodes near the end points of the interval. Second is the sampling rate that never approaches, even asymptotically, the rate for periodic functions, namely, π versus two points per wavelength, see e.g. [8]. As it turns out, the nodes of the generalized Gaussian quadratures for exponentials do not concentrate excessively (the rate reported in [6] is in error, see Section 2.2) and the sampling rate asymptotically approaches the rate for periodic functions.

In recent preprints [9,10] the authors present a study of the PSWFs as a tool for solving PDEs. We note that our use of the PSWFs differs in several ways that have a significant impact on the performance. We first select the desired accuracy and then, for a given bandlimit, construct the (nearly) optimal quadratures for these parameters. Alternatively, for a selected accuracy and a given number of nodes, we find the largest possible bandlimit (see discussion in Section 2.2). We note that in [9,10] the number of nodes is selected proportional to the bandlimit, which is not the optimal choice. We also use a different approach to time evolution described below.

An important observation in using the PSWFs is that the norm of the derivative matrix based on bandlimited functions is smaller than that based on polynomials. In constructing derivative operators we incorporate boundary conditions into the derivative matrix. In the case of discontinuous interface conditions, these conditions are also incorporated into the derivative matrix in a way similar to [11]. We also use the spectral projector to remove spurious large eigenvalues and corresponding eigenspaces from the derivative operators, thus further reducing their norm. For time evolution we use a semigroup approach (that involves computing the matrix exponential) and compare it with the standard fourth-order Runge–Kutta method. We note that for time evolution one can also use the approach introduced in [12] or the spectral method in [13]. We will discuss approaches that avoid computing the matrix exponential explicitly elsewhere.

We write the acoustic equation as a first order system [14]. After discretizing the spatial operator, the equation takes the form of the system of linear first order ordinary differential equations:

$$\mathbf{u}_t = L\mathbf{u} + \mathbf{F}(t)$$

with the initial condition $\mathbf{u}(0) = \mathbf{u}_0$. In the case of time independent coefficients, the solution is given by

$$\mathbf{u}(t) = e^{tL}\,\mathbf{u_0} + \int_0^t e^{(t-\tau)L}\,\mathbf{F}(\tau)\,d\tau. \tag{1}$$

Using (1) for time evolution requires computing the matrix exponential $e^{\Delta tL}$ for a time step Δt. The computation of $e^{\Delta tL}$ and applying it to a function is costly in dimensions 2 and higher and, therefore, this approach is rarely used for numerical computations.

We use the separated representation introduced in [15] to represent the operator L for problems in two or higher dimensions. This representation significantly reduces the cost of computing the matrix exponential and matrix–vector multiplications. The separated representation of an operator in two or higher dimensions is given by a sum

of products of operators acting in one dimension. We refer to the number of terms in the separated representation as the separation rank. The separation rank for the matrix exponential $e^{\Delta t L}$ grows with the size of the time step Δt, and we will see that a time step between one and two temporal periods is appropriate to control both the separation rank and the number of time steps. We note a typical time step in problems of wave propagation is a fraction of a temporal period.

We reduce the computational cost further by using the partitioned low-rank (PLR) representation for operators acting in one dimension. This representation is similar to the partitioned singular value decomposition considered in [16,17]. We note that both the separated and PLR representations are interesting on their own, with applications in other areas, e.g., computational quantum mechanics (see [15,18]).

We note that in [19,13] the authors present a spectral method for applying the matrix exponential without constructing such matrix. Our approach is competitive if the problem has to be solved repeatedly for the same model with different initial conditions. We will consider a comparison of the method in [19,13] with our approach separately.

We begin with a review of the bandlimited functions in Section 2 and construct derivative operators incorporating boundary and interface conditions in the following section. In Section 4 we provide several numerical examples demonstrating the accuracy of the derivative matrix based on bandlimited functions and also construct integration operators with respect to bandlimited functions. In the following section we review the separated representation and the PLR representation, and describe linear algebra algorithms for operators in these representation. We also introduce the PLR representation and describe linear algebra algorithms for operators in this representation. Finally, we apply these tools to solve the acoustic equation in two dimensions in Section 6 and give a number of numerical examples and comparisons.

2. Bandlimited functions and their approximations

In physical phenomena there is always a bound for both the spatial/time extent and the wavenumber/frequency range. However, a function cannot be compactly supported in both the space and the Fourier domain. In order to manage this apparent contradiction, it is natural to consider the basis of eigenfunctions of the space and band limiting operator. This has been the topic of a series of papers by Slepian et al. [1–5], which introduced the prolate spheroidal wave functions (PSWFs) as an eigensystem bandlimited in $[-c, c]$ and maximally concentrated within the space interval $[-1, 1]$.

The bandlimited periodic functions can be expanded into the Fourier basis $\{e^{ik\pi x}\}_{k=0}^{N}$ or, if we consider zero boundary conditions, into the basis $\{\sin k(\pi(x+1))/2\}_{k=1}^{N}$. However, in order to divide the computational domain into subdomains, we need to allow arbitrary boundary conditions on the subdomains, and neither the Fourier nor the sine basis are then acceptable. This motivates the introduction of a basis that can efficiently represent functions of the type e^{ibx} for an arbitrary real value b, such that $|b| < c$, where c is a fixed parameter, the bandlimit.

We note that solutions of equations of mathematical physics behave more like exponentials than polynomials. This provides a naive but compelling motivation for using bandlimited functions rather than polynomials, as a tool for approximating solutions. As we demonstrate, for a given accuracy, computing with bandlimited functions significantly reduces the computational cost.

2.1. The prolate spheroidal wave functions

Let us briefly review the results in [1,2,20] relevant to the purposes of this paper. The PSWFs are constructed for a fixed bandlimit $c > 0$. Consider the operator $F_c : L^2([-1, 1]) \to L^2([-1, 1])$:

$$F_c(\psi)(\omega) = \int_{-1}^{1} e^{icx\omega}\, \psi(x)\, dx \tag{2}$$

and $Q_c = (c/2\pi)F_c^* F_c$:

$$Q_c(\psi)(y) = \frac{1}{\pi}\int_{-1}^{1} \frac{\sin(c(y-x))}{y-x}\psi(x)\,\mathrm{d}x.$$

The PSWFs are the eigenfunctions of the operators Q_c and F_c. The eigenvalues λ of F_c and μ of Q_c are related via

$$\mu = \frac{c}{2\pi}|\lambda|^2. \tag{3}$$

In our notation we may suppress the dependence of the eigenfunctions and eigenvalues on c.

Let us consider the spaces of bandlimited functions,

$$\mathcal{B}_c = \{f \in L^2(\mathbb{R}) | \hat{f}(\omega) = 0 \text{ for } |\omega| \geq c\}.$$

The PSWFs form a complete basis in $L^2([-1,1])$ and $\mathcal{B}_c$ [1]. The eigenfunctions $\psi_j(x)$ are real and orthogonal on both $[-1,1]$ and $\mathbb{R}$:

$$\int_{-1}^{1} \psi_i(x)\psi_j(x)\,\mathrm{d}x = \delta_{ij} \tag{4}$$

and

$$\int_{-\infty}^{\infty} \psi_i(x)\psi_j(x)\,\mathrm{d}x = \frac{1}{\mu_i}\delta_{ij}, \tag{5}$$

where μ_i are eigenvalues of the operator Q_c.

The PSWFs are uniformly bounded on $[-1,1]$, $\|\psi_j\|_{L^\infty([-1,1])} \leq K_c$, for some constant K_c, for all $j = 0, 1, \ldots$ The existence of K_c can be proven by observing that the PSWFs approach the Legendre polynomials for $j \gg c$, although finding tight bounds remains an open problem.

The eigenvalues of Q_c are real and the spectrum is naturally divided into three parts. For large bandlimits c, the first $\approx 2c/\pi$ eigenvalues μ_i of Q_c are close to 1. The next $\approx \log c$ eigenvalues make an exponentially fast transition to zero and the remaining eigenvalues are very close to zero.

We have from (2) the spectral decomposition of the kernel:

$$\mathrm{e}^{\mathrm{ic}\omega\mathrm{x}} = \sum_{\mathrm{j}=0}^{\infty} \lambda_{\mathrm{j}}\psi_{\mathrm{j}}(\omega)\psi_{\mathrm{j}}(\mathrm{x}) \tag{6}$$

for all $x, \omega \in [-1,1]$. This is the most efficient separated representation for $\mathrm{e}^{\mathrm{ic}\omega\mathrm{x}}$, where the series can be truncated for some $j > 2c/\pi$, due to the exponential decay of the eigenvalues λ_j.

For the derivatives of PSWFs we establish the following proposition.

Proposition 1. *On the interval* $[-1,1]$,

$$\left\| \frac{\mathrm{d}\psi_j}{\mathrm{d}x} \right\|_{L^2([-1,1])} \leq c\|\psi_j\|_{L^2(\mathbb{R})} = \frac{c}{\sqrt{\mu_j}}.$$

The proof follows from Bernstein's inequality (see e.g. [21. Ch. 2.5]) and $\|\psi_j\|_{L^2(\mathbb{R})} = 1/\sqrt{\mu_j}$. It is interesting to compare this bound with another version of Bernstein's inequality (see, e.g., [21, Ch. 2.4]), which states that if $p(x)$ is an nth degree polynomial on the interval $[-1,1]$ and $|p(x)| \leq 1$ then, on this interval, $|p'(x)| \leq n^2$.

Recently, the generalized Gaussian quadratures for bandlimited exponentials were developed in [6,7].

Proposition 2. *For $c > 0$ and $\epsilon > 0$, we construct nodes $-1 < \theta_1 < \theta_2 < \cdots < \theta_M < 1$ and weights $w_k > 0$, such that for any $x \in [-1, 1]$:*

$$\left| \int_{-1}^{1} e^{ictx}\, dt - \sum_{k=1}^{M} w_k\, e^{ic\theta_k x} \right| < \epsilon \tag{7}$$

and the number of nodes, M, is (nearly) optimal. The nodes and weights maintain the natural symmetry, $\theta_k = -\theta_{M-k+1}$ and $w_k = w_{M-k+1}$.

Thus, we can integrate all functions e^{ibx} with $|b| < c$ using Proposition 2. The nodes and weights in Proposition 2 are computed as a function of the bandlimit $c > 0$ and the accuracy $\epsilon > 0$ and can be viewed as the generalized Gaussian quadratures for the bandlimited functions. We note that the algorithm in [7] identifies the nodes of the generalized Gaussian quadratures as zeros of the *discrete* prolate spheroidal wave functions (DPSWF) corresponding to small eigenvalues. For a study of DPSWFs we refer to [5].

2.2. On the distribution of nodes for Gaussian quadratures

As it is well known, nodes of Gaussian quadratures (both the usual and generalized) accumulate near the end points as the number of nodes grows. The rate of such accumulation has a critical influence in a variety of applications where quadratures are used either for integration or interpolation.

Although we compute the nodes and weights as in [7] by selecting first the bandlimit, c, and then computing the minimal (or nearly minimal) number of nodes, M, to achieve a given accuracy ϵ, once such quadratures are generated we use the number of nodes as the variable and $c = c(M, \epsilon)$ to study node accumulation.

Let us consider the ratio

$$r(M, \epsilon) = \frac{\theta_2 - \theta_1}{\theta_{\lfloor M/2 \rfloor} - \theta_{\lfloor M/2 \rfloor - 1}}, \tag{8}$$

where "$\lfloor M/2 \rfloor$" denotes least integer part. Observing that the distance between nodes of the Gaussian quadratures changes monotonically from the middle of the interval toward the end points, and that the smallest distance is between the two nodes closest to an end point, this ratio can be used as a measure of node accumulation. For example, the distance between the nodes near the end points of the standard Gaussian quadratures for polynomials decreases as $\mathcal{O}(1/M^2)$, where M is the number of nodes, so that we have $r(M, \epsilon) = \mathcal{O}(1/M)$.

Using the method in [7], we have computed the generalized Gaussian quadratures for different accuracies and observed the rate $r(M, \epsilon)$ at which nodes accumulate near the end points. We illustrate our results for two choices of accuracy, $\epsilon \approx 10^{-7}$ and $\approx 10^{-17}$. The error

$$\epsilon(M) = \max_{x \in [-1,1]} \left| 2\frac{\sin cx}{cx} - \sum_{m=1}^{M} w_m\, e^{ic\theta_m x} \right| \tag{9}$$

was computed by selecting equally spaced points in $[-1, 1]$ (including the end points) with an oversampling factor of 10. Although we attempted to maintain a fixed accuracy, it is changing slightly as M varies and it results in a jittery appearance of graphs in Figs. 1 and 2.

In Fig. 1 we show that the oversampling factor:

$$\alpha(M, \epsilon) = \frac{\pi M}{c(M, \epsilon)} > 1$$

approaches 1 for large M. This factor compares the critical rate of sampling of smooth periodic functions, either for integration or interpolation, to that of smooth (non-periodic) functions on an interval. We recall that in the case of the Gaussian quadratures for polynomials this limit is $\pi/2$ rather than 1 (see e.g. [8]).

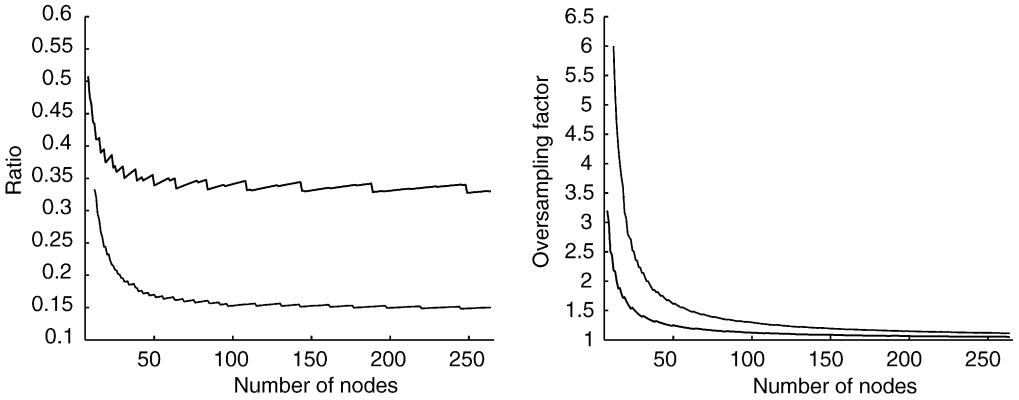

Fig. 1. The ratio $r(M,\epsilon)$ in (8) and the oversampling factor $\alpha(M,\epsilon)$ plotted against the number of nodes for quadratures of accuracy $\epsilon \approx 10^{-7}$ and $\approx 10^{-17}$ (see also Fig. 2).

We note that an erroneous comment was made in [6] about the rate of accumulation of nodes, suggesting that (in our terms) $r(M,\epsilon) = \mathcal{O}(1/\sqrt{M})$. Our results clearly rule out this rate of accumulation, suggesting instead that the ratio approaches a constant, $r(M,\epsilon) = \mathcal{O}(-\log\epsilon)$, although there might be weaker terms not easily observable in our experiments. An asymptotic analysis of DPSWFs and PSWFs should lead to an analytic estimate of $r(M,\epsilon)$.

We also note that there have been attempts to modify the polynomial based quadratures to avoid the problems caused by the accumulation of nodes near the end points [23,24]. However such approach resolves the issues associated with using such quadratures only partially.

2.3. Bases for bandlimited functions on an interval

Following [7], let us define

$$\mathcal{E}_c = \left\{ u \in L^\infty([-1,1]) | u(x) = \sum_{k\in\mathbb{Z}} a_k \, \mathrm{e}^{\mathrm{i}c\mathrm{b}_k x} : \{\mathrm{a}_k\}_{k\in\mathbb{Z}} \in l^1, \mathrm{b}_k \in [-1,1] \right\}.$$

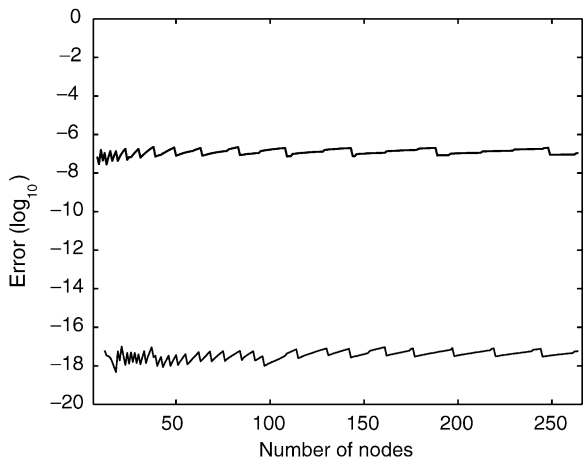

Fig. 2. The accuracy of the quadrature $\epsilon(M)$ in (9) as a function of the number of nodes.

We have $\mathcal{E}_c \subseteq C^\infty([-1,1])$ and prove the following theorem (see Appendix A)

Theorem 3. *For every $\epsilon > 0$ and $u \in \mathcal{B}_c$ there exists a function $\tilde{u} \in \mathcal{E}_c$, such that $\|u - \tilde{u}\|_{L^2([-1,1])} < \epsilon$.*

Any bandlimited function from $\mathcal{E}_c$ can be approximated by a linear combination of a finite number of exponentials in the form $e^{ic\theta_k x}$ where $|\theta_k| \le 1$. The phases θ_k are chosen as nodes of the generalized Gaussian quadratures ([7, Theorem 6.1], see also [6]). Following [7], we use the quadrature nodes and weights to construct bases for $\mathcal{E}_c$.

Theorem 4. *Consider $u \in \mathcal{E}_c$,*

$$u(x) = \sum_{k\in\mathbb{Z}} a_k \, e^{ib_k x}$$

and let $\{\theta_l\}_{l=1}^M$ and $\{w_l\}_{l=1}^M$ be quadrature nodes and weights for the bandlimit $2c$ and accuracy ϵ^2. Then there exist coefficients $\{u_l\}_{l=1}^M$ and a constant A such that

$$\left\| u(x) - \sum_{l=1}^{M} u_l \, e^{ic\theta_l x} \right\|_{L^\infty([-1,1])} \le A \left(\sum_{k\in\mathbb{Z}} |a_k| \right) \epsilon.$$

The set of exponentials $\{e^{ic\theta_k x}\}_{k=1}^M$ may be viewed as a basis for bandlimited functions $\mathcal{E}_c$ with accuracy ϵ. The basis of exponentials has the obvious advantage of being easy to differentiate and integrate but these functions are far from being orthonormal and one must be careful using them for numerical computations. In this respect they are analogous to monomials as a basis for polynomials. In order to construct a basis analogous to the orthogonal polynomials, we turn to the PSWFs.

Instead of using the PSWFs directly, we choose to construct their approximations [7], as it is sufficient for our purposes. Given the bandlimit $c > 0$ and accuracy threshold $\epsilon > 0$, let us construct quadrature nodes and weights according to Theorem 4. We then solve the algebraic eigenvalue problem:

$$\sum_{l=1}^{M} w_l \, e^{ic\theta_m\theta_l} \, \Psi_j(\theta_l) = \eta_j \Psi_j(\theta_m) \tag{10}$$

and define the approximate PSWFs on $[-1, 1]$ by

$$\Psi_j(x) = \frac{1}{\eta_j} \sum_{l=1}^{M} w_l \, e^{icx\theta_l} \, \Psi_j(\theta_l), \tag{11}$$

where $\psi(\theta_l)$ are the eigenvectors in (10).

The matrix in (10) does not have zero eigenvalues as can be easily checked numerically although we do not have a proof for this fact. We expect the eigenvalues $\{\eta_j\}_{j=1}^M$ to approximate the first M eigenvalues $\{\lambda_j\}$ and eigenvectors of F_c. This is indeed the case, with the exception of small eigenvalues, where the relative error may be large. Since the absolute values of the first $\simeq 2c/\pi$ eigenvalues in (2) are very close, some of them are numerically indistinguishable (within the machine precision). As a result, we do not construct approximations to the individual PSWFs via (10) but, instead, approximate correctly the subspace spanned by these functions.

Let us consider the inner products of functions in (11):

$$S_{ij} = \int_{-1}^{1} \Psi_i(x)\Psi_j(x) \, dx \tag{12}$$

for $i, j = 1, \ldots, M$. We have the following proposition (see [7], Proposition 8.1).

Proposition 5. *The functions Ψ_m and Ψ_n are nearly orthogonal and the elements of S satisfy*

$$|S_{mn} - \delta_{mn}| \le \begin{cases} \dfrac{\epsilon^2 \sum_{k=1}^{M} w_k}{|\eta_m||\eta_n|} & \text{if } \Psi_m \text{ and } \Psi_n \text{ are both even or both odd} \\ 0 & \text{otherwise} \end{cases}.$$

The matrix S deviates from the identity matrix only when both η_m and η_n are small and close to ϵ. We observe that in our numerical experiments the condition number of S has been less than 3.

In many applications, it is convenient to work with function values as well as with expansion coefficients with respect to a set of basis functions. Following [7], we define the interpolating basis functions for bandlimited functions as

$$R_k(x) = \sum_{l=1}^{M} r_{kl}\, \mathrm{e}^{\mathrm{i}c\theta_l \mathrm{x}} \tag{13}$$

for $k = 1, \ldots, M$, where

$$r_{kl} = \sum_{j=1}^{M} w_k \boldsymbol{\Psi}_j(\theta_k) \frac{1}{\eta_j} \boldsymbol{\Psi}_j(\theta_l) w_l.$$

It is shown in [7] that the functions $R_k(x)$ are interpolating, $R_k(\theta_l) = \delta_{kl}$.

2.4. Examples of approximation by bandlimited functions

The three bases for $\mathcal{E}_c$, the exponential basis, the basis of approximate PSWFs, and the interpolating basis span the same subspace since they are constructed as linear combinations of the eigenvectors in (10). However, it is important to observe that the condition numbers of the transformation matrices for changing bases are drastically different and determine how these bases are used for numerical computations. In Table 1 we display the condition numbers of transformation matrices for two accuracies. In both cases the condition number for transforming between approximate PSWFs and the interpolating basis is small, while the other transformation matrices have very large condition numbers.

This is similar to transformations between bases spanning the subspace of polynomials of degree $\le N$, namely, the monomials, the Legendre polynomials, and the Lagrange interpolating polynomials with the Legendre nodes. The basis of monomials corresponds to the basis of exponentials, while the basis of approximate PSWFs (which are nearly orthonormal) is similar to that of the Legendre polynomials.

Let us provide several examples of approximation by the bandlimited functions. In our examples we sample the function at the quadrature nodes which gives us the coefficients of the interpolating basis. We then find the

Table 1
Condition number for transformation matrices, $c = 8.5\pi$

Transformation matrix	$\epsilon = 10^{-7}$	$\epsilon = 10^{-14}$
Prolate → interpolating	2.7	3.5
Exponential → prolate	1.1×10^8	2.5×10^{14}
Exponential → interpolating	1.2×10^8	3.1×10^{14}

The accuracy $\epsilon = 10^{-7}$ requires 32 nodes and the accuracy $\epsilon = 10^{-14}$ requires 41 nodes.

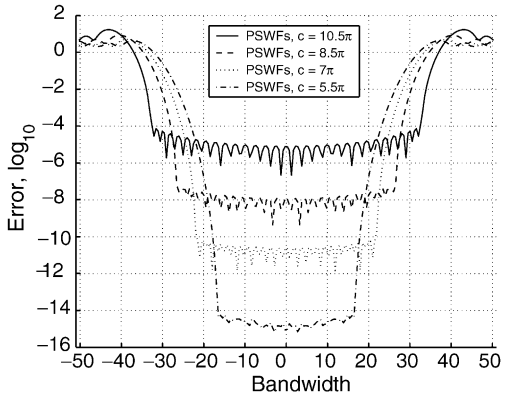

Fig. 3. Absolute error ($\log_{10}$) for approximating the function e^{ibx} in the interval $[-1, 1]$ with $|b| \leq 16\pi$. We use quadratures with 32 nodes and different accuracies.

expansion coefficients β_k with respect to the basis of the approximate PSWFs:

$$f(x) \simeq \sum_{k=1}^{N} \beta_k \Psi_k(x). \tag{14}$$

Expanding each PSWF via exponentials, we obtain the coefficients α_k and

$$f(x) \simeq \sum_{k=1}^{N} \alpha_k \, \mathrm{e}^{\mathrm{i}c\theta_k x}. \tag{15}$$

In Fig. 3 we illustrate the error of approximating the function e^{ibx} in the interval $[-1, 1]$. In Fig. 4 we display the error of approximating the Chebyshev polynomials and an "almost" bandlimited Gaussian $f(x) = \mathrm{e}^{-x^2/2\sigma^2}$ on the interval $[-1, 1]$ with variances $\sigma^2 \in [0.00005, 5]$.

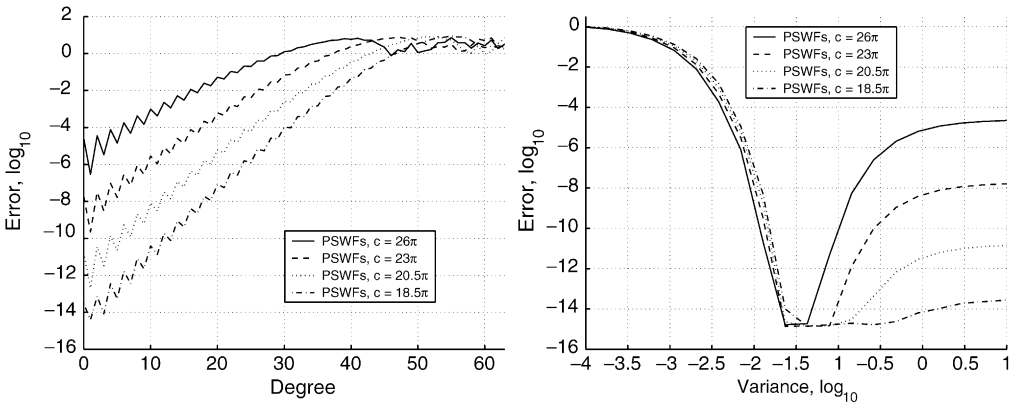

Fig. 4. Absolute error ($\log_{10}$) of approximating the Chebyshev polynomials $T_k(x)$, $k = 0, \ldots, 63$, on $[-1, 1]$ (left) and the Gaussians using approximate PSWFs. We use quadratures with 64 nodes and different accuracies.

3. Derivative matrices with boundary and interface conditions

In this section we illustrate how to incorporate the boundary and interface conditions into the derivative operator. We follow the method in [11] (essentially the tau method, see e.g. [25,26]) and extend the technique to a non-orthogonal basis since the approximate PSWFs $\{\Psi_i(x)\}_{i=1}^M$ are not orthonormal.

Let S be the M-by-M matrix (12). We consider the derivative operator on an interval subdivided into subintervals and use 2^N subintervals noting that other subdivisions are also possible. We set $\bar{x}_l = -1 + 2^{1-N} l$ for $l = 0, 1, \ldots, 2^N$, and define $\phi_{kl}(x)$ by

$$\phi_{kl}(x) = \begin{cases} \Psi_k(2^N(x - \bar{x}_l) - 1), & x \in [\bar{x}_l, \bar{x}_{l+1}], \\ 0, & x \notin [\bar{x}_l, \bar{x}_{l+1}] \end{cases} \tag{16}$$

for $l = 0, \ldots, 2^N - 1$ and $k = 1, \ldots, M$.

Consider functions $f(x)$ of the form

$$f(x) = \sum_{l=0}^{2^N-1} \sum_{i=1}^{M} s_{il} \phi_{il}(x). \tag{17}$$

Let us represent the derivative $f'(x)$ as

$$\frac{\mathrm{d}f}{\mathrm{d}x} = \sum_{l=0}^{2^N-1} \sum_{i=1}^{M} \tilde{s}_{il} \phi_{il}(x), \tag{18}$$

where the transition matrix between the coefficients s_{il} and $\tilde{s}_{il}$ has the block tridiagonal structure

$$D = \begin{pmatrix} r_0^l & r_{-1} & & & & & r_1^l \\ r_1 & r_0 & r_{-1} & & & & \\ & r_1 & r_0 & r_{-1} & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & \ddots & \ddots & \ddots & \\ & & & & r_1 & r_0 & r_{-1} \\ & & & & & r_1 & r_0 & r_{-1} \\ r_{-1}^r & & & & & & r_1 & r_0^r \end{pmatrix}, \tag{19}$$

where each block is an $M \times M$ matrix.

For each interval let us define $\mathbf{s}_l = [s_{1l}, s_{2l}, \ldots, s_{Ml}]^{\mathrm{T}}$ and $\tilde{\mathbf{s}}_l = [\tilde{s}_{1l}, \tilde{s}_{2l}, \ldots, \tilde{s}_{Ml}]^{\mathrm{T}}$ for $l = 0, \ldots, 2^N - 1$, where the coefficients s_{il} and $\tilde{s}_{il}$ are the expansion coefficients in (17) and (18). Following the derivation in [11], we obtain

$$S\tilde{\mathbf{s}}_l = -bG^*\tilde{\mathbf{s}}_{l-1} + ((1-a)F - (1-b)E - K)\mathbf{s}_l + aG\mathbf{s}_{l+1} \tag{20}$$

for $l = 0, \ldots, 2^N - 1$, where E, F and G are rank one matrices defined by

$$E_{kl} = \overline{\Psi_k(-1)}\Psi_l(-1), \qquad F_{kl} = \overline{\Psi_k(1)}\Psi_l(1) \quad \text{and} \quad G_{kl} = \overline{\phi\Psi_k(1)}\Psi_l(-1),$$

and $0 \le a, b \le 1$ are coupling parameters for the subintervals. For the first subinterval we have

$$S\tilde{\mathbf{s}}_0 = ((1-a)F - E - K)\,\mathbf{s}_0 + aG\mathbf{s}_1 \tag{21}$$

Table 2
The expressions for the blocks in (19)

Stencil type	Expression
Periodic	$r_1 = r_l^l = -\frac{1}{2}(S^{-1}G^*)$
	$r_{-1} = r_{-1}^r = \frac{1}{2}(S^{-1}G)$
	$r_0^l = r_0^r = r_0 = \frac{1}{2}(S^{-1}(K^* - K))$
$f(\pm 1) = 0$	$r_{-1} = \frac{1}{2}(S^{-1}G)$
	$r_1 = -\frac{1}{2}(S^{-1}G^*)$
	$r_0^l = S^{-1}(\frac{1}{2}F - K)$
	$r_0 = \frac{1}{2}(S^{-1}(K^* - K))$
	$r_0^r = S^{-1}(-\frac{1}{2}E - K)$
$f(\pm 1)$ arbitrary	$r_{-1} = \frac{1}{2}(S^{-1}G)$
	$r_1 = -\frac{1}{2}(S^{-1}G^*)$
	$r_0^l = S^{-1}(\frac{1}{2}F - E - K)$
	$r_0 = \frac{1}{2}(S^{-1}(K^* - K))$
	$r_0^r = S^{-1}(F - \frac{1}{2}E - K)$

Only non-zero blocks shown.

and if $f(-1) = 0$, then

$$S\tilde{\mathbf{s}}_0 = ((1-a)F - K)\mathbf{s}_0 + aG\mathbf{s}_1. \tag{22}$$

For the last subinterval we have

$$S\tilde{\mathbf{s}}_{2^N-1} = -bG^*\mathbf{s}_{2^N-2} + (F + (b-1)E - K)\mathbf{s}_{2^N-1} \tag{23}$$

and if $f(\bar{x}_{2^N-1}) = 0$, then we obtain

$$S\tilde{\mathbf{s}}_{2^N-1} = -bG^*\mathbf{s}_{2^N-2} + ((b-1)E - K)\mathbf{s}_{2^N-1}, \tag{24}$$

where G^* denotes the complex transpose.

In order to construct derivative matrices for periodic boundary conditions, we use (20) for the interior subintervals. For the first and the last subintervals we use (20) by identifying $\mathbf{s}_{-1} = \mathbf{s}_{2^N-1}$ and $\mathbf{s}_{2^N} = \mathbf{s}_0$. Using (20)–(24) with $a = b = 1/2$, we obtain expressions for the blocks in (19) as shown in Table 2, where we used $K = F - E - K^*$.

If we have only one interval, then the derivative matrix for arbitrary boundary values is $D = S^{-1}K^*$, the derivative matrix for zero boundary conditions $D_0 = -S^{-1}K$, and that for periodic boundary conditions is given by

$$D_{\text{per}} = \tfrac{1}{2}(S^{-1}(G - G^* - K + K^*)).$$

4. Differentiation and integration of bandlimited functions

Let us compare numerical differentiation and integration of bandlimited functions with finite differences and pseudo-spectral methods. We use the derivative operators constructed in the previous section and demonstrate that using spectral projectors to remove spurious eigenvalues of derivative matrices with boundary conditions improves the accuracy.

For the first test let us fix the number of nodes and change the accuracy ϵ, thus obtaining different bandlimits c. Using 32 nodes on the interval $[-1, 1]$, we note that corresponding bandlimit for periodic functions (sampled at the Nyquist frequency) is 16π. We construct 4 derivative matrices using 32 nodes and accuracies ϵ equal to 10^{-13},

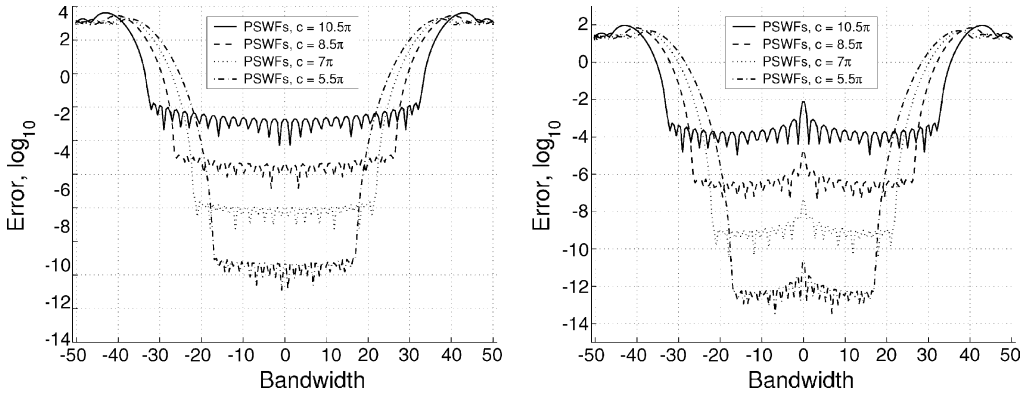

Fig. 5. Absolute (left) and relative (right) errors for the first derivative of the function e^{ibx} in the interval $[-1, 1]$ with $|b| \leq 16\pi$ using a basis of 32 approximate PSWFs.

10^{-10}, 10^{-7}, and 10^{-4}, with the corresponding bandlimits c set to 5.5π, 7π, 8.5π, and 10.5π, respectively. We differentiate the function $f(x) = e^{ibx}$ (not necessarily periodic in $[-1, 1]$) for 200 values of b, $-16\pi \leq b \leq 16\pi$. For each b we differentiate using 32×32 derivative operator and then interpolate the result to 32 equally spaced points (including the end points) in the interval $[-1, 1]$ and compare it with the exact answer. The result is shown in Fig. 5.

We note from Fig. 5 that the error is almost uniform for $|b| \leq c$. It is also clear that a derivative matrix constructed for a lower accuracy gives a good approximation within a larger bandwidth than a derivative matrix constructed for a higher accuracy.

Compared to Fig. 3 we note that we lose 2–3 digits in differentiation. This is expected since according to Proposition 1 the ratio $\|Du\|_2/\|u\|_2$ is approximately bounded by c and, thus, the absolute error may be amplified by a factor $\approx c$. Since the maximum absolute norm of $\mathrm{d}e^{ibx}/\mathrm{d}x$ equals b, dividing the absolute error by b gives us the relative error which is smaller than the absolute error for all but the lowest frequencies (see Fig. 5).

4.1. Comparison with pseudo-spectral methods and finite differences

Let us now compare the accuracy of differentiation using approximate PSWFs to a second order finite-difference and spectral differentiation. For spectral differentiation we use the Chebyshev polynomials. We construct two derivative matrices using approximate PSWFs for the accuracy $\epsilon = 10^{-7}$ and bandlimit $c = 8.5\pi$, and $\epsilon = 10^{-13}$ and bandlimit $c = 5.5\pi$. For comparison, we construct a second-order central finite-difference derivative matrix, using a second order boundary stencil for the first and the last row of the matrix. For the spectral differentiation, we construct a block diagonal derivative matrix where each diagonal block is a derivative matrix with respect to the first eight Chebyshev polynomials constructed using the algorithm in ([25, Appendix C]). Each block is applied to one of the four subintervals $[-1, -1/2]$, $[-1/2, 0]$, $[0, 1/2]$, and $[1/2, 1]$. We use subdivision since the derivative matrices based on Chebyshev polynomials tend to have large norm for high degree polynomials [27]. We differentiate the function $f(x) = \sin(bx)$ for 200 values of b, $-16\pi \leq b \leq 16\pi$. The result is shown in Fig. 6.

We next consider an experiment using 64 nodes. We construct two derivative matrices using PSWFs with the accuracy $\epsilon = 10^{-7}$ and bandlimit $c = 23\pi$, and with $\epsilon = 10^{-13}$ and bandlimit $c = 18.5\pi$. For comparison, we construct a second-order central finite-difference derivative matrix and, for spectral differentiation, a block diagonal spectral derivative matrix where each diagonal block is a derivative matrix with respect to the first 16 Chebyshev polynomials. We differentiate the function $f(x) = \sin(bx)$ for 200 values of b, $-32\pi \leq b \leq 32\pi$. The result is shown in Fig. 6.

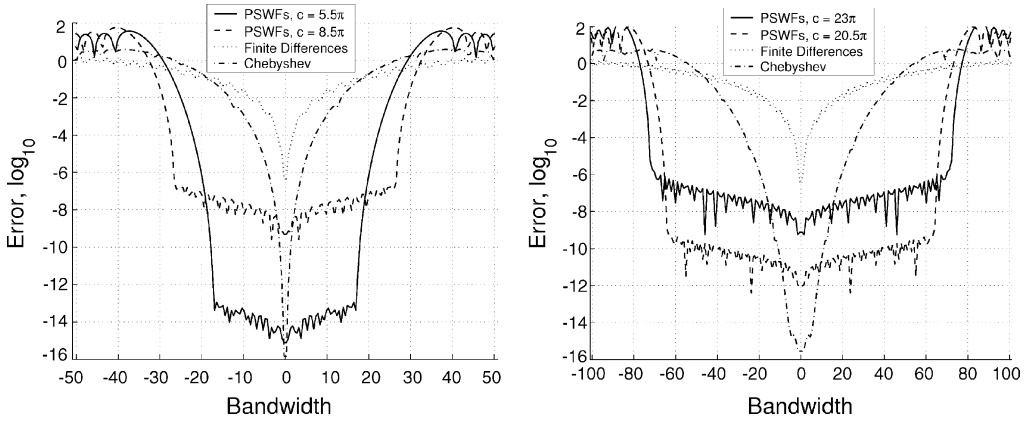

Fig. 6. Comparison of absolute errors for the first derivative of the function $\sin(bx)$ in the interval $[-1, 1]$ with $|b| \leq 16\pi$ and $|b| \leq 32\pi$. The derivative matrices in the basis of approximate PSWFs are constructed using 32 and 64 nodes.

4.2. Using spectral projectors to improve accuracy

The accuracy of the derivative matrix can be increased by projecting out a subspace corresponding to "spurious" eigenvectors. Our approach is similar to filtering of eigenvalues of derivative matrices obtained with polynomial quadratures (see e.g. [28]). We describe such projection for the first derivative operator with the periodic boundary conditions and later use it for the second derivative operator with zero boundary conditions.

Consider the eigenvalue problem

$$Du \equiv u' = \lambda u, \quad u(-1) = u(1). \tag{25}$$

It is easily seen that

$$u_k(x) = \mathrm{e}^{\mathrm{i}k\pi x},$$

for $k = 0, 1, \ldots$, are eigenfunctions of (25) with the corresponding eigenvalues $\lambda_k = \mathrm{i}k\pi$. Let us consider a discretization of D obtained by using the approximate PSWFs with the bandlimit c. The eigenfunctions of the discretized problem mimic the eigenfunctions $u_k(x)$ and we obtain a good approximation of $u_k(x)$ for all $k = 0, 1, \ldots$ such that $k\pi \leq c$. For $c/\pi < k \leq N$, the eigenvectors "attempt" to describe the corresponding eigenfunctions $u_k(x)$, but the accuracy of the approximation rapidly decrease with increasing k. We note that the eigenvectors corresponding to eigenfunctions $u_k(x)$ for $k > c/\pi$ are not useful to us, since we seek an approximation of bandlimited functions within the bandlimit c. Hence, the eigenvectors corresponding to frequencies $k > c/\pi$ can be discarded. More formally, let P denote the projector onto the space spanned by all eigenfunctions $u_k(x)$ such that $k \leq c/\pi$. Our goal is then to find a derivative matrix that approximates the operator PDP.

To project the derivative matrix D, we diagonalize D and set the unwanted eigenvalues to zero. Formally, let us denote by $\mathbf{e}_k$ and $\mathbf{f}_k$ the left and the right eigenvectors of D, with the eigenvalue λ_k. We scale $\mathbf{e}_k$ (or $\mathbf{f}_k$) so that $\mathbf{f}_k^{\mathrm{T}}\mathbf{e}_k = 1$ and define $P_k = \mathbf{e}_k\mathbf{f}_k^{\mathrm{T}}$. Since D is an N-by-N diagonalizable matrix, we have $D = \sum_{k=1}^{N} \lambda_k P_k$. Then the projected matrix $\tilde{D}$ is given by $\tilde{D} = \sum_{|\lambda_k| \leq c} \lambda_k P_k$. Alternatively, we can use the sign iteration method described in [17].

The same procedure applies to the second derivative operator with zero boundary conditions. The second derivative is constructed as $L_0 = DD_0$, where D is the first derivative operator without boundary conditions and D_0 is the

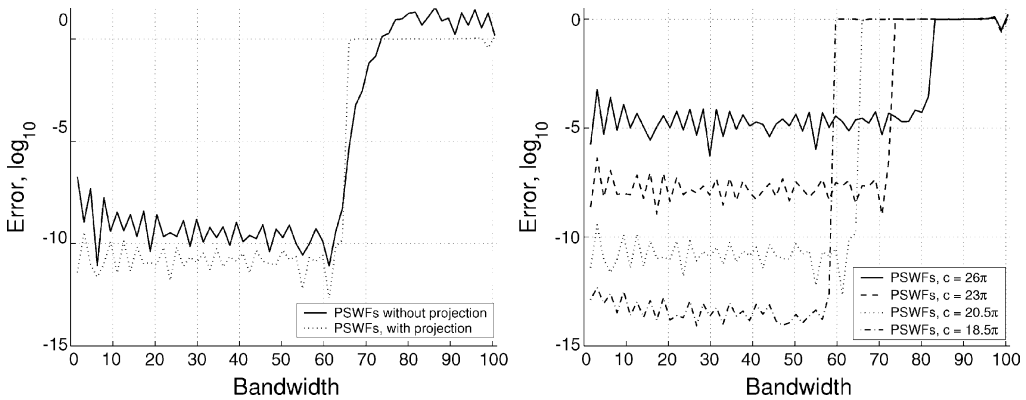

Fig. 7. Comparison of the error for spectrally projected second derivative with zero boundary conditions (left) and the error for different accuracies (right) for the function $\sin b_k x$, where $b_k = k\pi/2$, $k = 1, \ldots, 64$.

first derivative operator with zero boundary conditions. We then apply the spectral projector and arrive at

$$L_{\text{proj}} = \sum_{|\lambda_k| \le c^2} \lambda_k \mathbf{e}_k \mathbf{f}_k^{\mathrm{T}},$$

where $\mathbf{e}_k$ and $\mathbf{f}_k$ are the left and the right eigenvector of L_0, respectively, scaled such that $\mathbf{f}_k^{\mathrm{T}} \mathbf{e}_k = 1$.

Let us demonstrate the impact of using the spectral projector. We construct two second derivative matrices with zero boundary conditions with and without the spectral projector, for the bandlimit $c = 20.5\pi$ and accuracy $\epsilon = 10^{-10}$ using 64 nodes. In all experiments we differentiate the function $f(x) = \sin b_k x$, where $b_k = k\pi/2$, $k = 1, \ldots, 64$. If we use the projected derivative matrix, then the error is smaller within the bandlimit c, as shown in Fig. 7. We attribute reduced error to a smaller norm of the projected derivative matrix (by a factor $\approx$50) and zero eigenvalues for highly oscillatory, spurious eigenfunctions. We further illustrate in Fig. 7 the performance of the projected derivative matrix for different accuracy thresholds and resulting different bandwidths.

4.3. The integration operator

In solving integral equations it is often useful to map a sequence of function values $\{f(\theta_k)\}_{k=1}^N$ to the sequence of integrals $\{\int_{-1}^{\theta_k} f(x)\,\mathrm{d}x\}_{k=1}^N$. Let us construct an integration matrix for bandlimited functions on an interval,

$$T_{lk} = \int_{-1}^{\theta_k} R_l(x)\,\mathrm{d}x,$$

where $R_l(x)$ is a function of the interpolating basis for bandlimited functions. We use the definition of Ψ_l to obtain

$$\int_{-1}^{\theta_k} \Psi_l(x)\,\mathrm{d}x = \frac{1}{\eta_l} \sum_{m=1}^{N} w_m \boldsymbol{\Psi}_l(\theta_m) \frac{\mathrm{e}^{\mathrm{i}c\theta_{\mathrm{m}}\theta_{\mathrm{k}}} - \mathrm{e}^{-\mathrm{i}c\theta_{\mathrm{m}}}}{\mathrm{i}c\theta_m},$$

and proceed by using the definition of $R_l(x)$ in terms of the approximate PSWFs to obtain T_{lk}.

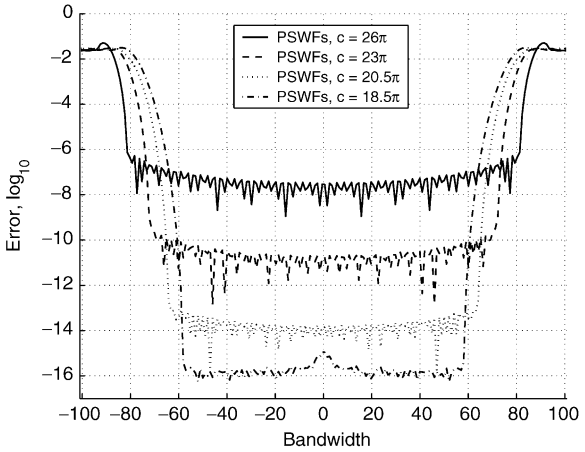

Fig. 8. Absolute error for integrals $\int_{-1}^{\theta_k} e^{ibx}\,\mathrm{d}x$ with $|b| \le 32\pi$, where $\{\theta_k\}_{k=1}^{64}$ are the quadrature nodes.

Let us illustrate the accuracy of integration using the bandlimited functions. We use 64 nodes and construct 4 derivative matrices using the same setting for the bandlimits and the accuracies as before. Let $\{\theta_k\}_{k=1}^{64}$ be a set of the generalized Gaussian quadrature nodes for bandlimited functions. We compute the integrals $\int_{-1}^{\theta_k} e^{ibx}\,\mathrm{d}x$ for 200 values of $-32\pi \le b \le 32\pi$. The results are shown in Fig. 8.

5. Separated and partitioned low rank representations of operators

Using the matrix exponential to solve the wave equation allows us to take large time steps while controlling the accuracy. However, computing the matrix exponential directly in two and higher dimensions becomes prohibitively expensive even for moderate matrix sizes since the computational cost of the matrix-matrix multiplication is $O(N^{3d})$ in d-dimensions. In order to overcome the prohibitive cost of computing and using the matrix exponential, we need an efficient operator representation. We use separated representations that have been introduced in [15]. In this paper we consider only the two-dimensional case, and note that our approach generalizes to higher dimensions using the algorithms in [15]. For operators in each separated direction we also use the so-called partitioned low rank (PLR) representation, a simplification of the partitioned singular value decomposition (PSVD) used in [16,29,30,17].

5.1. The separated representation

Let us consider a linear operator L acting on functions of two variables. We represent L by a matrix $L(j_1, j_1', j_2, j_2')$, where the indices (j_1', j_2') denote the input and (j_1, j_2) the output variables. For simplicity we assume the same range for all indices, $j_1, j_1', j_2, j_2' = 1, \ldots, N$.

Definition 6. For a given ϵ, we represent the matrix $L(j_1, j_1', j_2, j_2')$, $L : \mathbb{C}^{N^2} \to \mathbb{C}^{N^2}$, as

$$\sum_{k=1}^{r} s_k A_k(j_1, j_1') \otimes B_k(j_2, j_2'),$$

where $\otimes$ denotes the Kronecker product, $\{A_k(j_1, j_1')\}_{k=1}^r$ and $\{B_k(j_2, j_2')\}_{k=1}^r$ are $N \times N$ matrices, $\|A_k\| = 1$, $\|B_k\| = 1$, $s_k > 0$, and

$$\left\| L(j_1, j_1', j_2, j_2') - \sum_{k=1}^{r} s_k A_k(j_1, j_1') \otimes B_k(j_2, j_2') \right\| \leq \epsilon.$$

The number of terms in the representation, r, is the separation rank of L for accuracy ϵ.

We note that the separation rank differs from the operator rank, a similar representation that splits the input and output variables. We note that (only in two dimensions) the separated representation can be computed using the singular value decomposition (SVD). However, even in that case we use a simpler algorithm described below. In higher dimension algorithms for computing separated representations can be found in [15].

If $u \in \mathbb{C}^{N^2}$ is a vector in two dimensions, stored as a two-dimensional array, then the matrix–vector product can be computed by

$$\sum_{k=1}^{r} s_k A_k u B_k^{\mathrm{T}}, \tag{26}$$

where the matrix A_k acts upon the columns of u, and B_k acts upon the rows of u. The computational cost for a matrix–vector multiplication is then given by $O(2rN^3)$ provided that no additional representations are employed. Linear algebra operations in separated representations are easily accomplished but the separation rank of the result will grow. For example, the matrix product of the two separated representations L_1 and L_2 is computed as

$$L_1 L_2 = \sum_{k=1}^{r_1} \sum_{l=1}^{r_2} s_k^{(1)} s_l^{(2)} (A_k^{(1)} A_l^{(2)}) \otimes (B_k^{(1)} B_l^{(2)}), \tag{27}$$

yielding the separation rank $r_1 r_2$. To reduce the separation rank in (27) while maintaining accuracy ϵ, we use the algorithm described in Section 5.3. In most cases, the resulting rank $\tilde{r}$ is significantly less than $r_1 r_2$.

As an example of a separated representation, consider an operator with variable coefficients,

$$L = \frac{1}{\kappa(x, y)} \frac{\partial}{\partial x} \left(\sigma(x, y) \frac{\partial}{\partial x} \right) + \frac{1}{\kappa(x, y)} \frac{\partial}{\partial y} \left(\sigma(x, y) \frac{\partial}{\partial y} \right)$$

in the acoustic equation $u_{tt} = Lu$. We construct

$$\left\| \sigma(x, y) - \sum_{l=1}^{r_\sigma} s_l^\sigma \hat{\sigma}_l(x) \tilde{\sigma}_l(y) \right\| \leq \epsilon, \qquad \left\| \frac{1}{\kappa(x, y)} - \sum_{l'=1}^{r_\kappa} s_{l'}^\kappa \frac{1}{\hat{\kappa}_{l'}(x) \tilde{\kappa}_{l'}(y)} \right\| \leq \epsilon,$$

and obtain an approximation to L,

$$\sum_{l'=1}^{r_\kappa} \sum_{l=1}^{r_\sigma} s_l^\sigma s_{l'}^\kappa \left[\frac{1}{\hat{\kappa}_{l'}(x)} \frac{\partial}{\partial x} \left(\hat{\sigma}_l(x) \frac{\partial}{\partial x} \right) \frac{\tilde{\sigma}_l(y)}{\tilde{\kappa}_{l'}(y)} + \frac{\hat{\sigma}_l(x)}{\hat{\kappa}_{l'}(x)} \frac{1}{\tilde{\kappa}_{l'}(y)} \frac{\partial}{\partial y} \left(\tilde{\sigma}_l(y) \frac{\partial}{\partial y} \right) \right],$$

which we may reduce further. This computation is performed using a discrete representation of all coefficients and operators and typically results in a very low separation rank.

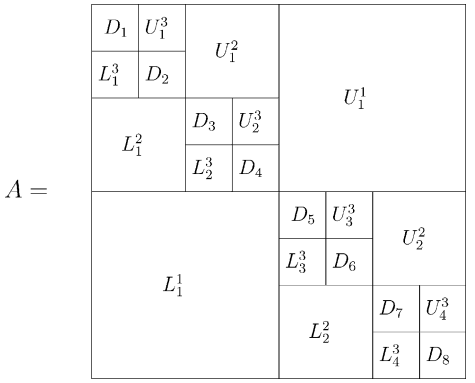

Fig. 9. Matrix subdivision for the 3-level PLR representation. The diagonal blocks are stored as full matrices and whereas the off-diagonal blocks are of low rank and are represented accordingly.

5.2. The partitioned low rank (PLR) representation

The partitioned low rank (PLR) representation is a simplification of the partitioned singular value decomposition (PSVD) introduced in [16,29,30], and used for spectral projectors in [17]. The PSVD is simplified by dropping the requirement of orthogonality between vectors as implied by the SVD and using a much simpler algorithm for rank reduction. The PSVD and PLR are more flexible than wavelet decompositions and are applicable to a wider class of matrices. In particular, the exponential of a matrix with pure imaginary spectrum and the bandlimited derivative matrix constructed in Section 4 are of high rank, dense, non-Toeplitz, with entries oscillatory as functions of indices. Unlike operators with real, negative spectrum, exponentials of such operators are not necessarily compressible via the wavelet transform while the PLR representation is efficient even when wavelet or multiwavelet transforms are dense. In Section 6 we apply PLR representation to exponentials of operators with pure imaginary spectrum (propagators) and its representation remains efficient for propagation over 1–2 periods (wavelengths).

The PLR representation is defined recursively by splitting a matrix into four blocks. The two diagonal blocks are split further, whereas the two off-diagonal blocks are maintained using a low rank representation of the form $\sum_i \sigma_i e_i f_i^*$. The 3-level PLR representation is illustrated in Fig. 9, where we use D_l, U_l^k, and L_l^k to denote the diagonal, upper and lower blocks of the partitioned matrix at different levels. This notation is convenient when describing linear algebra operations in the PLR representation.

In all our computations for a given accuracy $\epsilon > 0$, we seek an approximation $\tilde{A}$ of an operator A such that $\|A - \tilde{A}\| < \epsilon$, where $\|\cdot\|$ is an operator norm. For many operators in the PLR representation, the coefficients of the low rank representation of the off-diagonal blocks decay rapidly and we truncate the sum. Let us estimate the threshold value at each level in the PLR representation of an operator A such that $\|A - \tilde{A}\|_2 < \epsilon$, where $\tilde{A}$ is the truncated operator.

Proposition 7. *Consider a matrix A given by the m-level PLR representation. Let $\tilde{A}$ denote an approximation of A, where each off-diagonal block B is approximated by $\tilde{B}$,* so that $\|B - \tilde{B}\| \le \epsilon/(2^k m)$ *for level k, where $k = 1, \ldots m$. Then we have $\|A - \tilde{A}\| \le \epsilon$.*

Proof. We have

$$\|A\|_2 \le \sum_{l=1}^{2^k} \|D_l^k\|_2 + \sum_{k=1}^{m} \sum_{l=1}^{2^{k-1}} (\|U_l^k\|_2 + \|L_l^k\|_2). \tag{28}$$

Using the bound for the off-diagonal blocks, we arrive at the estimate.

Let a matrix A be in an m-level PLR representation. In order to compute the matrix–vector product Au, it is clear from Fig. 9 that there are two types of matrix–vector multiplications we need to evaluate. We need to compute the dense matrix–vector products for the diagonal blocks and the matrix–vector products $L_l^k \tilde{u}$ and $U_l^k \tilde{u}$, where $\tilde{u} \in \mathbb{C}^{N_k}$, $N_k = N/2^k$. If $L_l^k = \sum_{i=1}^r s_i e_i f_i^*$, then the matrix–vector product is computed as

$$L_l^k u = \sum_{i=1}^{r} s_i \langle u, f_i \rangle e_i. \tag{29}$$

The cost of such matrix–vector multiplication is $O(rN_k)$. Assuming that the rank of all off-diagonal blocks is the same, the total cost of computing Au is then estimated as $2^m N_m^2 + \sum_{k=1}^m 2^k r N_k$. If the total size of A is $N = 2^m$, then we have the estimate of the total cost of matrix–vector multiplication in the PLR representation as $O(N + rN \log N)$.

We next describe an algorithm for computing the product of two matrices given in the PLR representation. Consider the matrix product AB, where A and B are matrices in the m-level PLR representation. We consider A and B as the block matrices,

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \quad \text{and} \quad B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix},$$

where off-diagonal blocks are of the form $\sum_{i=1}^r \sigma_i e_i f_i^*$ and the diagonal blocks are in the $m-1$-level PLR representation. The product of A and B involves three types of block multiplications. The first is the product between two low rank representations, and such multiplication preserves the low rank form. The second is the product between the $m-1$-level PLR and a low rank representation which amounts to matrix–vector multiplications described above. The resulting low rank representation has to be added at the appropriate levels of the PLR representation and the rank of the result reduced via an algorithm described below. The third type of block multiplication is the product between the two $m-1$-level PLR representations which we treat recursively.

5.3. Rank reduction

In order to reduce the rank of a separated representation of the form $\sum_{i=1}^r \sigma_i e_i f_i^*$, where $\|e_i\| = \|f_i\| = 1$, we need to orthogonalize either the vectors $\{e_i\}_{i=1}^r$ or $\{f_i\}_{i=1}^r$ to reveal the actual rank of the representation. We call such procedure an orthogonalization sweep and use the size of the dynamically adjusted σ_i as pivots in choosing the order in which orthogonalization is performed. As we orthogonalize the vectors $\{e_i\}_{i=1}^r$, we simultaneously modify the vectors $\{f_i\}_{i=1}^r$ as to maintain the representation.

Once the vectors $\{e_i\}_{i=1}^k$ have been orthogonalized, we project the vectors $\{e_i\}_{i=k+1}^r$ onto the orthogonal complement of e_k, normalize, and modify f_k to maintain the representation. The vectors $\{e_i\}_{i=1}^{k+1}$ are now an orthonormal set and we continue the procedure recursively. We also use the dynamically computed σ_i to truncate this process as σ_i become smaller than the threshold of accuracy.

After one orthogonalization sweep, only one set of vectors remains orthogonal and we may choose to repeat the procedure for the other set. Such iterations eventually converge to the SVD. Since we do not require orthogonality in separated representations, we typically perform only two orthogonalization sweeps. This algorithm is briefly described in [15], where a significantly more complicated algorithm for reducing the separation rank is presented for higher dimensions.

6. Solution of the acoustic equation in two dimensions

Let us consider the acoustic equation

$$u_{tt} = \frac{1}{\kappa}[(\sigma u_x)_x + (\sigma u_y)_y] + F \tag{30}$$

with the initial conditions $u(x, y, t)_{|t=0} = f(x, y)$, $u_t(x, y, t)_{|t=0} = g(x, y)$ and the boundary condition $u_{|\partial\mathcal{D}} = h$, where $(x, y) \in \mathcal{D}$ and $t \in [0, \infty)$. Here the function $\kappa = \kappa(x, y)$ is the compressibility, and the function $\sigma = \sigma(x, y)$ is the specific volume (the inverse of density) of the medium.

We consider the domain $\mathcal{D}$ to be a rectangle which is subdivided, if necessary, into rectangular subdomains. We are using bandlimited functions on intervals and, thus, can subdivide without inflicting an unreasonable increase in the number of terms in the representation of functions on subdomains. In the future extensions of our approach for more general domains $\mathcal{D}$, we plan to subdivide them into subdomains and map such subdomains into rectangles so that we can use the tools developed in this paper.

Let us first rewrite the acoustic equation (30) as a first order system in time. Since the coefficients κ and σ are time independent, the propagator for the homogeneous problem ($F = 0$) is given by the exponential of a matrix. We represent the spatial operator by the separated representation described in Section 5.1 which decomposes the operator into a short sum of matrices acting in one dimension. We then compress these matrices using the PLR representation in Section 5.2.

Following the derivation by Bazer and Burridge [14], we introduce the functions v and w, and write the acoustic equation (30) as

$$\begin{bmatrix} w \\ v \\ u \end{bmatrix}_t = \begin{bmatrix} 0 & 0 & \sigma\dfrac{\partial^b}{\partial x} \\ 0 & 0 & \sigma\dfrac{\partial^b}{\partial_y} \\ \dfrac{1}{\kappa}\dfrac{\partial}{\partial x} & \dfrac{1}{\kappa}\dfrac{\partial}{\partial_y} & 0 \end{bmatrix} \begin{bmatrix} w \\ v \\ u \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \int_0^t F(x, y, \tau)\,\mathrm{d}\tau \end{bmatrix}, \tag{31}$$

where each block is given in its separated representation. Here $\partial^b/\partial x$ and $\partial^b/\partial y$ denote differentiation operators with boundary conditions imposed in the x and y direction. We rewrite equation (31) as

$$\mathbf{u}_t = L\mathbf{u} + \mathbf{F}, \quad \text{with } \mathbf{u}(0) = \mathbf{u}_0, \tag{32}$$

where $\mathbf{u} = [\, w\ v\ u\,]^{\mathrm{T}}$ and L is the linear operator incorporating the block matrix on the right-hand side of (31). Using bases for bandlimited functions (or finite differences in comparison codes), we discretize (32) in space resulting in a finite-dimensional system of ODEs. If L is time independent, then $\mathbf{u}(t) = \mathrm{e}^{\mathrm{tL}}\,\mathbf{u_0}$.

Eq. (32) is then solved by

$$\mathbf{u}(t) = P(t)\mathbf{u}_0 + P(t)\int_0^t P(\tau)^{-1}\mathbf{F}(\tau)\,\mathrm{d}\tau,$$

where the propagator $P(t)$ is an operator solving the integral equation:

$$P(t) = I + \int_0^t L(\tau)P(\tau)\,\mathrm{d}\tau.$$

We construct L using derivative operators for bandlimited functions with boundary and interface conditions incorporated according to Section 3 and propagate the solution by applying the matrix exponential e^{tL}. Let us describe

a numerical scheme for solving the homogeneous acoustic Eq. (30) in two dimensions with time independent coefficients and boundary conditions. We proceed via the following steps:

(1) Construct the derivative matrices representing $\partial/\partial x$, $\partial^b/\partial x$, $\partial/\partial y$ and $\partial^b/\partial y$ using the results in Section 3.
(2) Construct separated representations of the multiplication operators $1/\kappa(x, y)$ and $\sigma(x, y)$
(3) Construct blocks of the 3×3 spatial operator in (31) and use the algorithm in Section 5.3 to reduce the separation rank of each block.
(4) Select the time step Δt (see below) and compute the matrix exponential $e^{\Delta t L}$ using the scaling and squaring algorithm (see e.g. [31]). The linear combinations and products of the matrix blocks given in the separated representation are computed using the methods described in Section 5.
(5) Compute the solution $\mathbf{u}(t_k) = e^{\Delta t L}\,\mathbf{u}(t_{k-1})$, starting from $\mathbf{u}(t_0) = \mathbf{u}(0)$, for $k = 1, \ldots, N_{\text{time}}$.

We refer to this algorithm as the method of bandlimited bases (MBB) with the exponential propagator (EP). We note that for time dependent boundary conditions the problem can be reduced to that with zero boundary conditions and a forcing term.

If the factors in the separated representation (which are ordinary matrices) are large, we use the PLR representation described in Section 5.2 to speed up the computations in Steps 4 and 5 above. As discussed in Section 4, the norm of the derivative projectors can be greatly reduced by using spectral projectors. For example, to construct projected versions of $\partial/\partial x$ and $\partial^b/\partial x$ (derivative operators in the x-direction), we form the block matrix,

$$L = \begin{bmatrix} 0 & D_0 \\ D & 0 \end{bmatrix},$$

where D and D_0 are constructed as in Section 3. The boundary conditions are enforced only for the function u in (31) which results in the structure of the matrix L above. We then construct the projected operator,

$$L_{\text{proj}} = \sum_{|\lambda_k| \le c} \lambda_k \mathbf{e}_k \mathbf{f}_k^{\mathrm{T}},$$

where $\mathbf{e}_k$ and $\mathbf{f}_k$ are the left and the right eigenvector of L, scaled so that $\mathbf{f}_k^{\mathrm{T}} \mathbf{e}_k = 1$. The projected version of $\partial/\partial x$ is now given by the lower left block of L_{proj}, and the projected version of $\partial^b/\partial x$ for zero boundary conditions is given by the upper right block of L_{proj}. Using these blocks, we assemble the 3×3 block-matrix representation of the operator in (31).

In most numerical methods for wave propagation, the time step is restricted by the Courant–Friedrich–Lewy (CFL) condition (see e.g. [32]). According to the CFL condition, the spatial step is controlled by the speed of the propagating waves to ensure stability. When using the matrix exponential, the time step Δt can be chosen as large as desired without causing instabilities as long as the operator L has no eigenvalues with positive real part. In our approach a very large time step will increase the separation rank and the ranks in the partitioned low rank representation. We have found that choosing the time step between 0.5 and 2 periods gives a good compromise between an efficient representation of the matrix exponential and the number of time steps.

In solving the equation, we maintain solutions at the quadrature nodes of the bandlimited representation. For illustrations, we interpolate the result to an equally spaced grid (see Section 14).

6.1. Comparison of the results

We compare the MBB with the exponential propagator (MBB with EP) to two other methods. In the first comparison we use the MBB but replace Steps 4 and 5 with the explicit RK4 solver in time (MBB with RK4). In

the second comparison we write the acoustic equation (30) as a first order system:

$$\begin{bmatrix} v \\ u \end{bmatrix}_t = \begin{bmatrix} 0 & \frac{1}{\kappa}\left[\frac{\partial}{\partial x}\left(\sigma\frac{\partial^b}{\partial x}\right) + \frac{\partial}{\partial y}\left(\sigma\frac{\partial^b}{\partial y}\right)\right] \\ I & 0 \end{bmatrix} \begin{bmatrix} v \\ u \end{bmatrix} + \begin{bmatrix} F \\ 0 \end{bmatrix} \tag{33}$$

and discretize it in space using the fourth-order finite difference stencil on an equally spaced grid including the endpoints (see [25]) and use the explicit RK4 solver in time (FD with RK4).

To compare methods for solving the acoustic equation, we introduce the characteristic time and length scales. Let us consider on $[-1, 1]$ the wave equation $u_{tt} = u_{xx}$ with the zero boundary conditions. This equation describes a medium with the unit compressibility and specific volume and, thus, the unit velocity. For the initial conditions $u(x, 0) = \sin((k\pi/2)(x + 1))$ and $u_t(x, 0) = 0$, and each integer k, we have the solution

$$u(x, t) = \sin(\tfrac{1}{2}(k\pi)(x + 1)) \cos(\tfrac{1}{2}(k\pi)t).$$

Since the velocity is equal to 1, we define the characteristic period, $\tau = \pi/b$, and the characteristic length scale, $\lambda = \pi/b$, where $b = k\pi/2$ is the bandlimit. For periodic solutions in dimension $d = 1$, the Nyquist sampling rate (in time) requires two samples per characteristic period τ. We note that in dimension $d = 2$, for sampling purposes the corresponding time period has an extra factor $\sqrt{2}$ since the solutions of $u_{tt} = u_{xx} + u_{yy}$ in $[-1, 1] \times [-1, 1]$, with the zero boundary condition and the initial conditions $u(x, y, 0) = \sin((k\pi/2)(x + 1)) \sin((k\pi/2)(y + 1))$ and $u_t(x, y, 0) = 0$, contain the factor $\cos(\sqrt{2}\pi t/\tau)$.

6.1.1. Comparison of accuracy and speed for constant coefficients

For our first set of experiments, we solve

$$\begin{aligned} &u_{tt} = u_{xx} + u_{yy}, \quad (x, y) \in (-1, 1) \times (-1, 1), \\ &u(x, y, 0) = \sin(\tfrac{1}{2}(\pi(x + 1))) \sin(\tfrac{1}{2}(\pi(y + 1))) + \sin(b(x + 1)) \sin(b(y + 1)), \\ &u(\pm 1, y) = u(x, \pm 1) = u_t(x, y, 0) = 0, \end{aligned} \tag{34}$$

where $b = k\pi/2$ for some integer $k > 1$, and the solution is given by

$$u(x, y, t) = \sin\left(\frac{\pi(x + 1)}{2}\right) \sin\left(\frac{\pi(y + 1)}{2}\right) \cos\left(\frac{\pi}{\sqrt{2}}t\right) + \sin(b(x + 1)) \sin(b(y + 1)) \cos(\sqrt{2}bt).$$

We note that this solution contains both low frequency (the first term) and high frequency (the second term) modes. For the experiments in this section, we measure the error of the vector $\mathbf{u} = [\, w \; v \; u \,]^{\mathrm{T}}$ using the relative norm, namely, if $\tilde{\mathbf{u}}$ approximates the exact solution $\mathbf{u}$, then

$$\text{error} = \frac{\|\mathbf{u} - \tilde{\mathbf{u}}\|_\infty}{\|\mathbf{u}\|_\infty}.$$

In the first experiment, we solve (34) using $b = 22.5\pi$. We propagate the solution and evaluate the error over a range of approximately 1–10^4 characteristic periods, and also record the CPU time it took to produce the solution.

For the MBB with EP, we construct 64 quadrature nodes and weights for the bandlimit $c = 23\pi$, which for periodic functions corresponds to an oversampling factor of approximately 1.4. We set the accuracy in the construction to $\epsilon = 10^{-7}$ resulting in 64 nodes, and select the time step $\Delta t = \sqrt{2}/23$ corresponding to approximately 1.4 characteristic periods. We represent the operator using the separated and PLR representations. This results in the separation rank $r = 5$ for the blocks in the exponential operator. For comparison, we use the same spatial

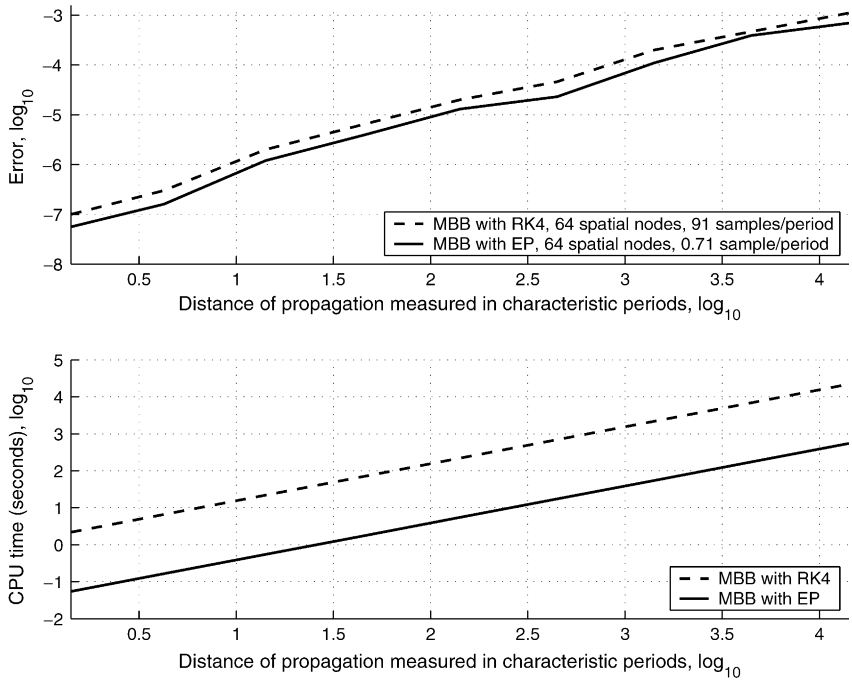

Fig. 10. Relative error in the max-norm for approximating the solution to (34) (top) and the computational time (bottom).

discretization as for the MBB with EP, but use the RK4 solver in time with the timestep $\Delta t/128$. The results are shown in Fig. 10.

In order for the finite difference fourth order scheme to reach similar accuracy, we need more than 1024 samples in space corresponding to an oversampling factor of approximately 22 (for periodic functions) and a timestep $t = \Delta t/128$. With this sampling rate, the computational time per characteristic period is almost 3 min, or more than 5000 times slower than using the MBB with EP. However, such oversampling factor is significantly larger than is typically used. For this reason, in the next experiment we solve the same equation, but use 400 samples in space for the fourth order scheme, corresponding to an oversampling factor of approximately 8.7 (compared for the Nyquist frequency for periodic functions), and a timestep $\Delta t/32$. The results are shown in Fig. 11. In this experiment, the computational times for the two methods are comparable, but the MBB with EP is significantly more accurate.

In the next experiment, we demonstrate that the cost of improving accuracy is small for the MBB with EP. Let us fix $b = 19.5$, and solve the model problem (34) using the MBB with EP for the bandlimit $c = 20\pi$ with 52, 56, 60, 64, and 68 nodes. For all solutions, we use the time step $\Delta t = \sqrt{2}/20$ (approximately 1.4 characteristic periods). The result is shown in Fig. 12.

We observe that using 60 nodes takes approximately two times longer than using 52 nodes but gives approximately 4 more digits of accuracy. We also note that the error increases linearly over time.

6.1.2. Numerical dispersion

Due to inaccuracies of differentiation, the different Fourier modes of a pulse propagate with different speeds. After some time the shape of the pulse deteriorates.

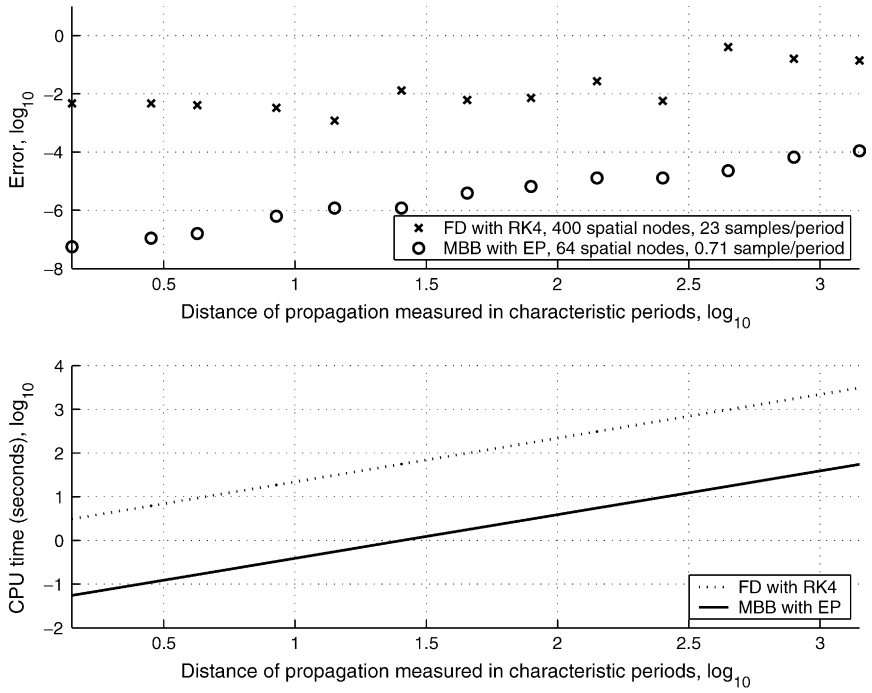

Fig. 11. Relative error ($\log_{10}$) in the max-norm for approximating the solution to (34) (top), and the computational time (bottom).

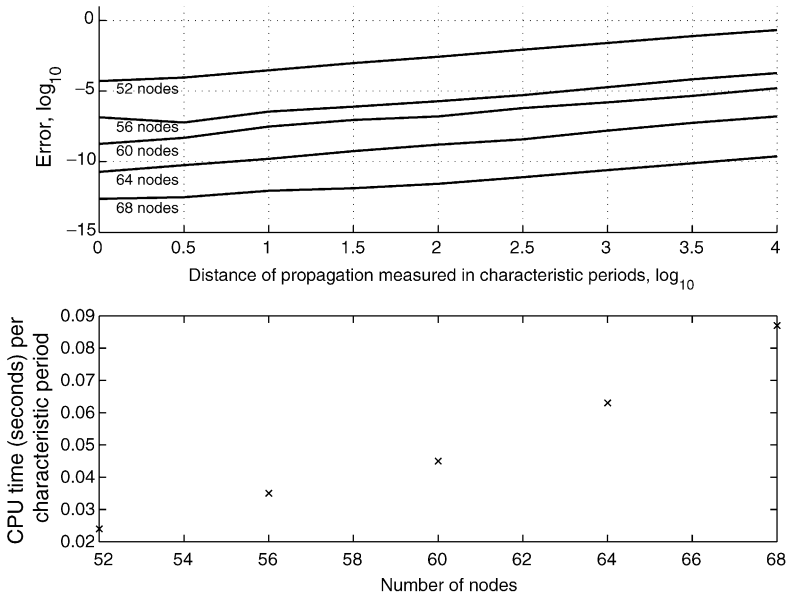

Fig. 12. Relative error in the max-norm for approximating the solution to (34) for $b = 19.5\pi$ using two different sampling rates (top). The CPU time for propagating the wave one characteristic period for a varying number of nodes (bottom).

To examine this phenomenon, let us consider the wave equation in one dimension, $u_t + cu_x = 0$, the solutions of which correspond to right-traveling waves. Solutions of this equation take the form:

$$u(x,t) = \mathrm{e}^{\mathrm{i}\omega(\mathrm{x}-\mathrm{ct})},$$

which we refer to as a Fourier mode of frequency ω traveling to the right with velocity c. Exact differentiation of this solution yields

$$\frac{\partial}{\partial x}u = \mathrm{i}\omega\,\mathrm{e}^{\mathrm{i}\omega(\mathrm{x}-\mathrm{ct})}.$$

If the error in the representation of the differentiation operator is of the form

$$\frac{\partial}{\partial x}u \simeq \mathrm{i}\,f(\omega)\,\mathrm{e}^{\mathrm{i}\omega(\mathrm{x}-\mathrm{ct})},$$

then the Fourier mode propagates with the velocity $cf(\omega)/\omega$. Unless $f(\omega) = \omega$, which corresponds to the exact differentiation, the Fourier modes of different frequencies travel with different velocities. For example, in the case of the second order centered finite difference approximation of the derivative, $f(\omega) = \sin(\omega)$.

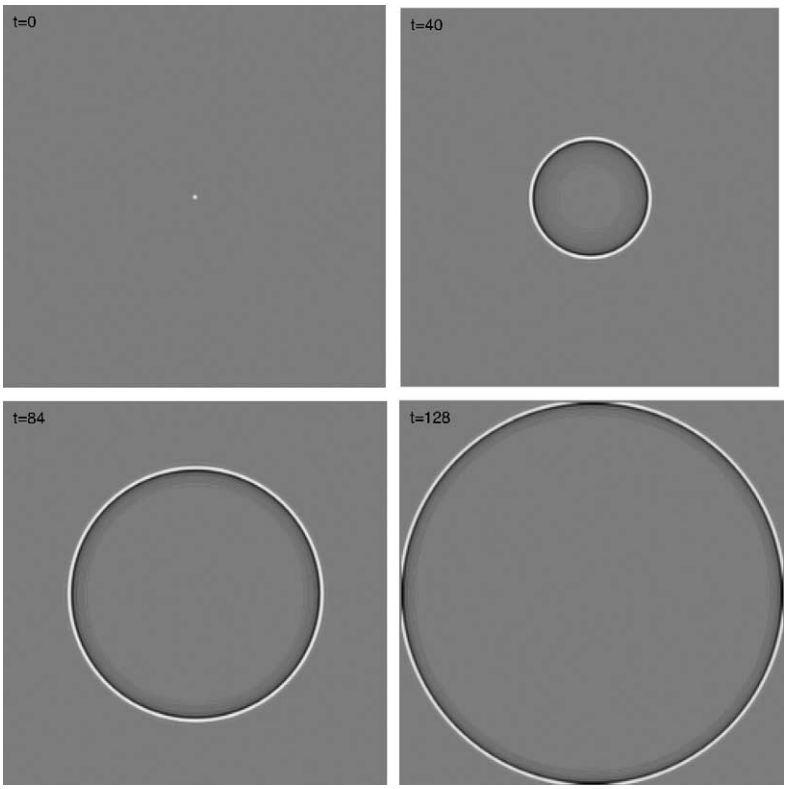

Fig. 13. Solution of (35) using the MBB with EP. The shape of the pulse is maintained throughout the propagation.

In this section we compare numerical dispersion using the MBB with EP and the FD with RK4 described in Section 6.1. Let us solve

$$u_{tt} = u_{xx} + u_{yy}, \quad (x, y) \in (-2, 2) \times (-2, 2), \qquad u(x, y, 0) = \mathrm{sinc}^2(27\pi x)\mathrm{sinc}^2(27\pi y),$$
$$u(\pm 2, y) = u(x, \pm 2) = u_t(x, y, 0) = 0. \tag{35}$$

The solution is a sharp pulse originating at the center of the domain, and expanding outward. In the absence of numerical dispersion, the shape of the pulse should be maintained.

For the MBB with EP, we construct 128 quadrature nodes and weights for the bandlimit $c = 54\pi$. We set the accuracy in the construction to $\epsilon = 10^{-7}$. We divide the domain into four subdomains and approximate the solution on each subdomain using 128-by-128 nodes. We use the time step $\Delta t = 2\pi/c$ corresponding to propagating two characteristic wavelengths, and represent the operator using the separated and PLR representations. This results in separation rank either $r = 5$ or 6 for the blocks of the exponential operator. For the fourth order scheme, we use 432 samples in space and the timestep $\Delta t = \pi/10c$ corresponding to propagating a tenth of the characteristic wavelength. This sampling rate yields approximately the same computational time for the two schemes. The results are shown as sequences of images in Figs. 13 and 14.

We note that in the MBB with EP the shape of the pulse is maintained. For the FD with RK4, the pulse begins to noticeably deteriorate, as the error accumulates due to the numerical dispersion. The numerical dispersion affects our method as well but at a much slower rate.

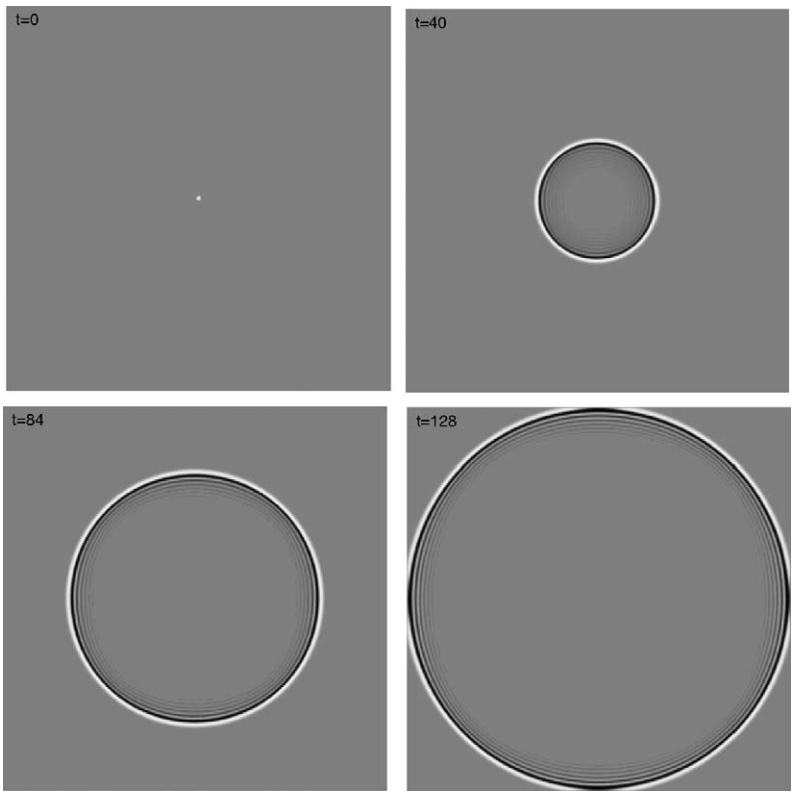

Fig. 14. Solution of (35) using the FD with RK4. Note the ripples near the wave front which are caused by numerical dispersion.

6.1.3. Numerical results for variable coefficients

Let us consider the acoustic equation with variable coefficients. Since we do not have an analytical solution, we simply display a sequence of images and study the shape of the pulse as it propagates throughout the domain. Let us solve

$$u_{tt} = \frac{1}{\kappa(y)}(u_{xx} + u_{yy}), \quad (x, y) \in (-1, 1) \times (-1, 1), \qquad u(x, y, 0) = \mathrm{e}^{-1000(\mathrm{x}^2+\mathrm{y}^2)},$$
$$u(\pm 1, y) = u(x, \pm 1) = u_t(x, y, 0) = 0, \tag{36}$$

where

$$\kappa(y) = \frac{1}{1 - \sin(\pi(y + 1))/2}.$$

The solution is a sharp pulse originating at the origin of the domain, and expanding outwards in the medium with varying velocity. For the MBB with EP, we construct 128 quadrature nodes and weights for the bandlimit $c = 54\pi$. We set the accuracy in the construction to $\epsilon = 10^{-7}$. We use the time step $\Delta t = 2\pi/c$ corresponding to propagating over two characteristic wavelengths. This choice of parameters yields the separation rank either $r = 7$ or 8 for the blocks of the exponential operator. Using the PLR representation for computing $\mathrm{e}^{\Delta \mathrm{tL}}$ u is in this case approximately 25% faster than using the dense representation of matrices in one dimension. The gain due to PLR increases for larger problems. For the FD with RK4, we use 216 samples in space and the timestep $\Delta t = \pi/10c$, corresponding

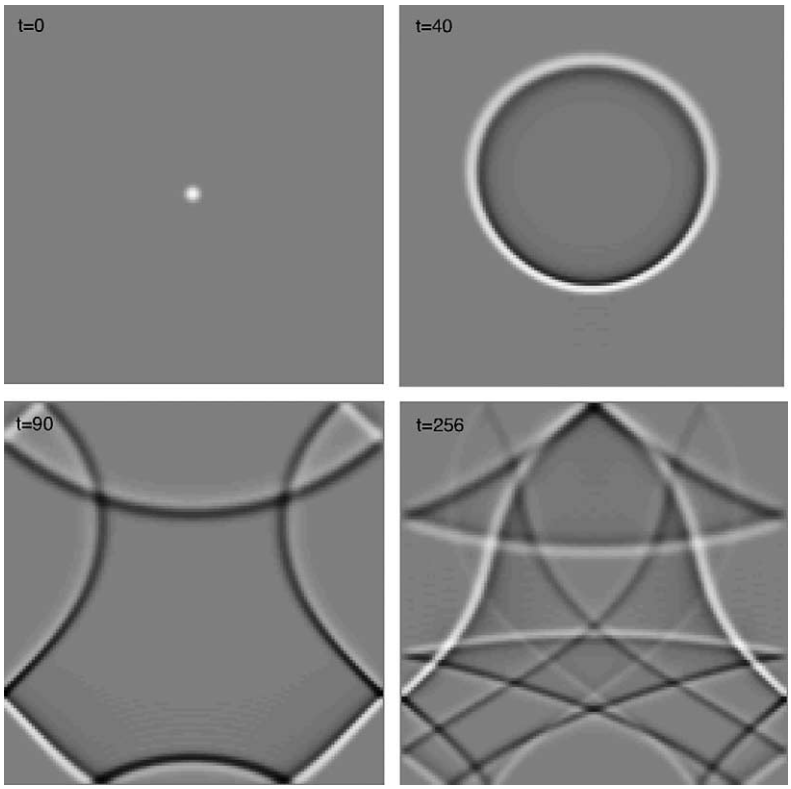

Fig. 15. Solution of (35) using the MBB with EP.

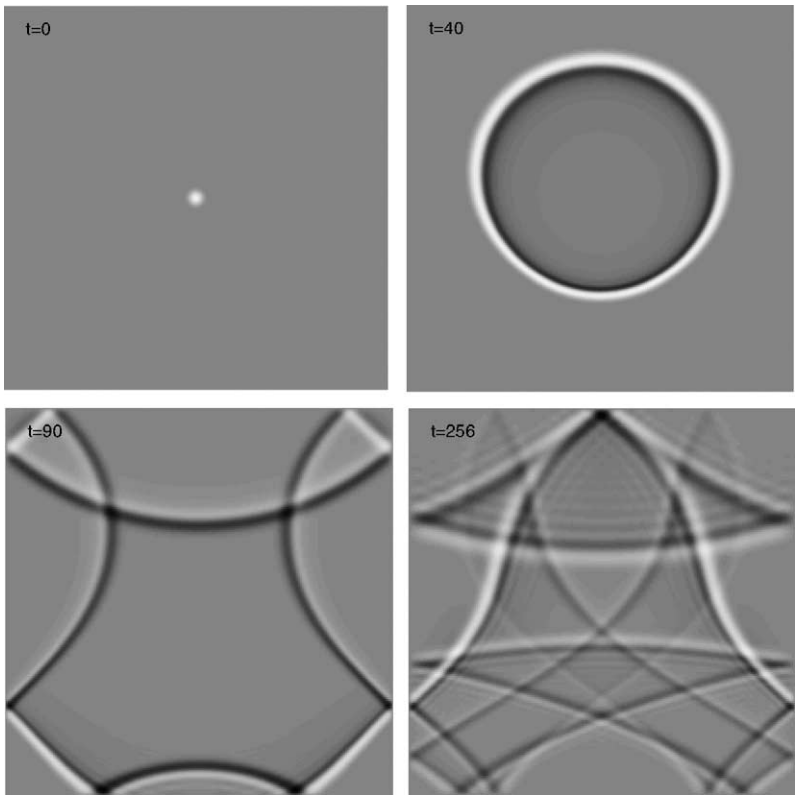

Fig. 16. Solution of (35) using the FD with RK4. Note the ripples which are caused by numerical dispersion.

to propagating over one-tenth of the characteristic wavelength. This sampling rate gives the two schemes approximately the same computational time. The results are shown as sequences of images in Figs. 15 and 16.

Both solutions behave qualitatively in the same way by propagating faster in the upper part of the domain where the wave velocity is higher. We note that for the MBB with EP, the shape of the pulse is maintained. For the FD with RK4, the pulse begins to noticeably deteriorate, as the error accumulates due to the numerical dispersion.

Appendix A. Proof of Theorem 3

First, let us consider a function $v \in \mathcal{B}_c \cap L^1(\mathbb{R})$. Using the Fourier transform, we write it (almost everywhere) as

$$\tilde{v}(x) = \int_{-c}^{c} \sigma(\omega)\, \mathrm{e}^{\mathrm{i}\omega x}\, \mathrm{d}\omega,$$

where σ is continuous and bounded since $v \in L^1(\mathbb{R})$. Let us define $b_k = -c + (2kc/N)$ for $k = 1, \ldots, N$. Then $|b_k| \leq c$ and we can approximate $\tilde{v}$ with the Riemann sum,

$$\tilde{v}(x) = \frac{2c}{N} \sum_{k=1}^{N} \sigma(b_k)\, \mathrm{e}^{\mathrm{i}b_k x} + \mathrm{E_N(x)},$$

where $\lim_{N\to\infty} E_N(x) = 0$ for all $x \in [-1, 1]$. We choose N sufficiently large, such that $\|E_N\|_{L^\infty[-1,1]} < \epsilon/2\sqrt{2}$ and define for $x \in [-1, 1]$

$$\tilde{u}(x) = \sum_{k=1}^{N} a_k \, \mathrm{e}^{\mathrm{i}b_k x},$$

where $a_k = 2c\sigma(b_k)/N$. Then $\tilde{u}$ is bounded on $[-1, 1]$ and $|v(x) - \tilde{u}(x)| < \epsilon/2\sqrt{2}$ almost everywhere.

Next we consider a function $u \in \mathcal{B}_c$. Then, since $\mathcal{B}_c \cap L^1(\mathbb{R})$ is dense in $\mathcal{B}_c$, there exists a function $v \in \mathcal{B}_c \cap L^1(\mathbb{R})$ such that

$$\|u - v\|_{L^2[-1,1]} < \tfrac{\epsilon}{2}. \tag{A.1}$$

As we showed above, there exists $\tilde{u} \in \mathcal{E}_c$ such that $|v(x) - \tilde{u}(x)| < \epsilon/2\sqrt{2}$ almost everywhere on $[-1, 1]$ and, hence,

$$\|v - \tilde{u}\|^2_{L^2[-1,1]} = \int_{-1}^{1} |v(x) - \tilde{u}(x)|^2 \, \mathrm{d}x \le \frac{\epsilon^2}{4},$$

which combined with (A.1) gives us

$$\|u - \tilde{u}\|_{L^2[-1,1]} \le \|u - v\|_{L^2[-1,1]} + \|v - \tilde{u}\|_{L^2[-1,1]} < \epsilon.$$

Acknowledegement

This research was supported in part by NSF/ITR grant DMS-0219326 (GB and KS), DOE grant DE-FG02-03ER25583 and NIMA grant NMA401-02-1-2002 (GB).

References

[1] D. Slepian, H.O. Pollak, Prolate spheroidal wave functions, Fourier analysis and uncertainty I, Bell Syst. Technol. J. 40 (1961) 43–63.
[2] H.J. Landau, H.O. Pollak, Prolate spheroidal wave functions, Fourier analysis and uncertainty II, Bell Syst. Technol. J. 40 (1961) 65–84.
[3] H.J. Landau, H.O. Pollak, Prolate spheroidal wave functions, Fourier analysis and uncertainty III, Bell Syst. Technol. J. 41 (1962) 1295–1336.
[4] D. Slepian, Prolate spheroidal wave functions, Fourier analysis and uncertainty IV. Extensions to many dimensions, generalized prolate spheroidal functions, Bell Syst. Technol. J. 43 (1964) 3009–3057.
[5] D. Slepian, Prolate spheroidal wave functions, Fourier analysis and uncertainty V. The discrete case, Bell Syst. Technol. J. 57 (1978) 1371–1430.
[6] H. Xiao, V. Rokhlin, N. Yarvin, Prolate spheroidal wavefunctions, quadrature and interpolation, Inverse Probl. 17 (4) (2001) 805–838.
[7] G. Beylkin, L. Monzón, On generalized Gaussian quadratures for exponentials and their applications, Appl. Comput. Harmon. Anal. 12 (3) (2002) 332–373.
[8] D. Gottlieb, S.A. Orszag, Numerical analysis of spectral methods: theory and applications, in: CBMS-NSF Regional Conference Series in Applied Mathematics, No. 26, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1977.
[9] J.P. Boyd, Prolate spheroidal wave functions as an alternative to Chebyshev and Legendre polynomials for spectral element and psedospectral algorithms, Preprint.
[10] Q. Chen, D. Gottlieb, J. Hesthaven, Spectral methods based on prolate spheroidal wave functions for hyperbolic PDEs, Preprint.
[11] B. Alpert, G. Beylkin, D. Gines, L. Vozovoi, Adaptive solution of partial differential equations in multiwavelet bases, J. Comput. Phys. 182 (1) (2002) 149–190.
[12] B. Alpert, L. Greengard, T. Hagstrom, An integral evolution formula for the wave equation, J. Comput. Phys. 162 (2000) 536–543.
[13] H. Tal-Ezer, Spectral methods in time for hyperbolic equations, SIAM J. Numer. Anal. 23 (1) (1986) 11–26.

[14] J. Bazer, R. Burridge, Energy partition in the reflection and refraction of plane waves, SIAM J. Appl. Math. 34 (1978) 78–92.
[15] G. Beylkin, M.J. Mohlenkamp, Numerical operator calculus in higher dimensions, Proc. Natl. Acad. Sci. USA 99 (16) (2002) 10246–10251.
[16] P. Jones, J. Ma, V. Rokhlin, A fast direct algorithm for the solution of the Laplace equation on regions with fractal boundaries, J. Comput. Phys. 113 (1) (1994).
[17] G. Beylkin, N. Coult, M.J. Mohlenkamp, Fast spectral projection algorithms for density-matrix computations, J. Comput. Phys. 152 (1) (1999) 32–54.
[18] G. Beylkin, M.J. Mohlenkamp, Algorithms for numerical analysis in high dimensions, University of Colorado, Applied Mathematics Preprint # 519, 2004, SIAM J. Sci. Comp. (2004), in press.
[19] H. Tal-Ezer, R. Kosloff, An accurate and efficient scheme for propagating the time dependent Schrödinger equation, The J. Chem. Phys. 81 (9) (1984) 3967–3971.
[20] D. Slepian, Some comments on Fourier analysis, uncertainty and modeling, SIAM Rev. 25 (3) (1983) 379–393.
[21] Y. Meyer, Wavelets and operators, Cambridge Studies in Advanced Mathematics, vol. 37, Cambridge University Press, Cambridge, 1992 (transl.; from the 1990 French original by D.H. Salinger).
[22] T. Rivlin, Chebyshev Polynomials, 2nd ed., Wiley, 1990.
[23] D. Kosloff, H. Tal-Ezer, A modified Chebyshev pseudospectral method with an $O(N^{-1})$ time step restriction, J. Comput. Phys. 104 (2) (1993) 457–469.
[24] J.S. Hesthaven, P.G. Dinesen, J.P. Lynov, Spectral collocation time-domain modeling of diffractive optical elements, J. Comput. Phys. 155 (2) (1999) 287–306.
[25] B. Fornberg, A Practical Guide to Pseudospectral Methods, Cambridge University Press, 1995.
[26] J.P. Boyd, Chebyshev and Fourier Spectral Methods, 2nd ed., Dover, Mineola, NY, 2001.
[27] L. Greengard, Spectral integration and two-point boundary value problems, SINUM 28 (4) (1991) 1071–1080.
[28] J.A.C. Weideman, L.N. Trefethen, The eigenvalues of second-order spectral differentiation matrices, SIAM J. Numer. Anal. 25 (6) (1988) 1279–1298.
[29] T. Hrycak, V. Rokhlin, An improved fast multipole algorithm for potential fields, SIAM J. Sci. Comp. 19 (6) (1998) 1804–1826.
[30] N. Yarvin, V. Rokhlin, A generalized one-dimensional fast multipole method with application to filtering of spherical harmonics, J. Comput. Phys. 147 (2) (1998) 594–609.
[31] G. Golub, C.V. Loan, Matrix Computations, 3rd ed., Johns Hopkins University Press, 1996.
[32] A. Iserles, A First Course in the Numerical Analysis of Differential Equations, Cambridge University Press, 1996.