Electronic Transactions on Numerical Analysis.
Volume 20, pp. 212-234, 2005.

ISSN 1068-9613.

ETNA
Kent State University
etna@mcs.kent.edu

KRYLOV SUBSPACE SPECTRAL METHODS FOR VARIABLE-COEFFICIENT INITIAL-BOUNDARY VALUE PROBLEMS*

JAMES V. LAMBERS†

Abstract. This paper presents an alternative approach to the solution of diffusion problems in the variable-coefficient case that leads to a new numerical method, called a Krylov subspace spectral method. The basic idea behind the method is to use Gaussian quadrature in the spectral domain to compute components of the solution, rather than in the spatial domain as in traditional spectral methods. For each component, a different approximation of the solution operator by a restriction to a low-dimensional Krylov subspace is employed, and each approximation is optimal in some sense for computing the corresponding component. This strategy allows accurate resolution of all desired frequency components without having to resort to smoothing techniques to ensure stability.

Key words. spectral methods, Gaussian quadrature, variable-coefficient, Lanczos method

AMS subject classifications. 65M12, 65M70, 65D32

1. Introduction. Let L be a self-adjoint second-order differential operator of the form

$$Lu = -(pu_x)_x + qu, \tag{1.1}$$

where $p(x) > 0$ and $q(x) \geq 0$ are 2π-periodic functions. We consider the diffusion equation on a bounded domain,

$$u_t + Lu = 0, \quad 0 < x < 2\pi, \quad t > 0, \tag{1.2}$$

$$u(x, 0) = f(x), \quad 0 < x < 2\pi, \tag{1.3}$$

with periodic boundary conditions

$$u(0, t) = u(2\pi, t), \quad t > 0. \tag{1.4}$$

In Section 4 we will discuss applications of the methods presented in this paper to more general problems.

1.1. Difficulties of Variable-Coefficient Problems. Spectral methods are extremely effective for solving the problem (1.2), (1.3), (1.4) in the case where the coefficients p and q are constant; see for instance [1], [7]. Unfortunately, the variable-coefficient case presents some difficulties for these methods:

- Let $\{\phi_\omega(x)\}_{\omega=0}^{N-1}$ be the natural basis of N trial functions defined by

  $$\phi_\omega(x) = \frac{1}{\sqrt{2\pi}} e^{i\omega x}.$$

  In the constant-coefficient case, these trial functions are eigenfunctions of L, but this is not true in the variable-coefficient case; in fact, the matrix of L in this basis is a full matrix in the general case.
- The phenomenon of *aliasing* can lead to *weak instability* (see for instance [5]), which manifests itself in the sudden blow-up of the solution. Unlike strong instability, it cannot be overcome simply by using a smaller time step, but rather one must use more grid points or filtering techniques (see [1]).

*Received October 17, 2003. Accepted for publication July 25, 2005. Recommended by D. Calvetti.
†Department of Petroleum Engineering, Stanford University, Stanford, CA 94305-2220 (lambers@stanford.edu).

ETNA
Kent State University
etna@mcs.kent.edu

As a result, substantially more computational effort must be expended for less information than in the constant-coefficient case. As variable-coefficient problems can be viewed as perturbations of their constant-coefficient counterparts, it should be possible to develop numerical methods that exploit this useful perspective.

1.2. Proposed Approach. Traditional Galerkin methods seek a solution in the space of trial functions that satisfies the PDE (1.2) in an average sense. In this paper we will instead compute an approximate solution $\tilde{u}(x,t)$ of the form

$$\tilde{u}(x,t) = \sum_{\omega=1}^{N} \hat{u}_\omega(t)\phi_\omega(x)$$

where $\{\phi_\omega\}_{\omega=1}^N$ is an orthonormal set of trial functions. For each $t > 0$, the coefficients $\{\hat{u}_\omega(t)\}_{\omega=1}^N$ are approximations of the coefficients of the exact solution $u(x,t) = e^{-Lt}f(x)$ in the basis $\{\phi_\omega\}$. Specifically,

$$\hat{u}_\omega(t) = \langle \phi_\omega, \tilde{u}(\cdot,t)\rangle \approx \langle \phi_\omega, \exp[-Lt]f\rangle$$

where the inner product $\langle\cdot,\cdot\rangle$ is defined by

$$\langle f, g\rangle = \int_0^{2\pi} \overline{f(x)}g(x)\,dx$$

and the solution operator $\exp[-Lt]$ is approximated using Krylov subspaces of L. This approach, in and of itself, is not new; for example, Hochbruck and Lubich have developed a method for approximating $\exp[-At]\mathbf{v}$, for a given vector $\mathbf{v}$ and Hermitian positive definite matrix A, using a Krylov subspace

$$\mathcal{K}(A, \mathbf{v}, m) = \text{span}\{\mathbf{v}, A\mathbf{v}, A^2\mathbf{v}, \ldots, A^{m-1}\mathbf{v}\} \tag{1.5}$$

for some choice of m. However, this approach is most accurate when the eigenvalues of A are clustered, which is not the case for a matrix that represents a discretization of L. For such stiff systems, one must take care to choose m sufficiently large, or t sufficiently small, in order to obtain sufficient accuracy (see [8] for details).

Our approach is to use a different approximation for each component $\hat{u}_\omega(t)$. By writing

$$\langle \phi_\omega, e^{-Lt}f\rangle = \frac{1}{4\delta}\left[\langle \phi_\omega + \delta f, e^{-Lt}(\phi_\omega + \delta f)\rangle - \langle \phi_\omega - \delta f, e^{-Lt}(\phi_\omega - \delta f)\rangle\right],$$

for some nonzero constant δ, we can reduce the problem of approximating $\langle \phi_\omega, e^{-Lt}f\rangle$ to that of approximating quadratic forms

$$\langle v_\omega, e^{-Lt}v_\omega\rangle$$

where $v_\omega = \phi_\omega \pm \delta f$. Each such quadratic form is computed using the Krylov subspace $\mathcal{K}(L, v_\omega, n)$ for some n. In this way, each frequency component of $\tilde{u}(x,t)$ can be computed independently, using an approximation that is, in some sense, optimal for that component; we will elaborate on this statement in Section 3. Furthermore, as we will see in Section 2, high-order temporal accuracy can be obtained using only low-dimensional Krylov subspaces.

2. Krylov Subspace Spectral Methods. In this section we describe Krylov subspace spectral methods and prove results concerning their convergence.

ETNA
Kent State University
etna@mcs.kent.edu

2.1. Elements of Functions of Matrices. We first discuss the approximation of quadratic forms

$$\langle u, f(L)u\rangle$$

where, in our application, $f(\lambda) = \exp[-\lambda t]$ for some $t > 0$. We first discretize the operator L on a uniform grid

$$x_j = jh, \quad j = 0, 1, \ldots, N-1, \quad h = \frac{2\pi}{N}, \tag{2.1}$$

where N is the number of gridpoints. On this grid, L is represented by an $N \times N$ symmetric positive definite matrix L_N defined by

$$S_N(t) = \exp[-L_N t], \quad [L_N]_{jk} = -p(jh)[D_N^2]_{jk} - p'(jh)[D_N]_{jk} + q(jh), \tag{2.2}$$

where D_N is a discrete differentiation operator defined on the space of grid functions. The function u is represented by an N-vector

$$\mathbf{u}_N = \begin{bmatrix} u(x_0) & \cdots & u(x_{N-1}) \end{bmatrix}^T.$$

We denote the eigenvalues of L_N by

$$b = \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N = a > 0$$

and the corresponding eigenvectors by $\mathbf{q}_1, \ldots, \mathbf{q}_N$. We can compute the quantity

$$\mathbf{u}_N^T f(L_N)\mathbf{u}_N \tag{2.3}$$

using a Gaussian quadrature rule to evaluate the Riemann-Stieltjes integral

$$I[f] = \int_a^b f(\lambda)\, d\alpha(\lambda) \tag{2.4}$$

where the measure $\alpha(\lambda)$ is defined by

$$\alpha(\lambda) = \begin{cases} 0 & \lambda < a \\ \sum_{j=i+1}^N |\mathbf{u}_N^T \mathbf{q}_j|^2 & \lambda_{i+1} \leq \lambda < \lambda_i \\ \sum_{j=1}^N |\mathbf{u}_N^T \mathbf{q}_j|^2 & b \leq \lambda. \end{cases} \tag{2.5}$$

The nodes and weights of the Gaussian quadrature rule can be obtained by applying the symmetric Lanczos algorithm (see [3]) to L_N with initial vector $\mathbf{u}_N$. After K iterations, we obtain a $K \times K$ tridiagonal matrix

$$J_K = \begin{bmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & \beta_{K-1} \\ & & & \beta_{K-1} & \alpha_K \end{bmatrix} \tag{2.6}$$

whose eigenvalues are the Gaussian quadrature rules for the integral (2.4). The weights are the squares of the first components of the eigenvectors of J_K. With $f(\lambda) = e^{-\lambda t}$, Gaussian quadrature yields a lower bound for $I[f]$ (see [15] for details). We can extend J_K to obtain a tridiagonal matrix J_{K+1} that has an eigenvalue at a. The resulting rule is a Gauss-Radau rule, which yields an upper bound for $I[f]$ (see [3]).

ETNA
Kent State University
etna@mcs.kent.edu

2.2. Algorithm Description. We now describe an algorithm for solving (1.2), (1.3), (1.4) using the quadrature rules described above. First, we consider the computation of quadratic forms of the form (2.3), where $f(\lambda) = \exp[-\lambda t]$ for given t.

For convenience, We denote by V_N the space of real-valued 2π-periodic functions of the form

$$f(x) = \frac{1}{\sqrt{2\pi}} \sum_{\omega=-N/2+1}^{N/2} e^{i\omega x} \hat{f}(\omega), \quad 0 < x < 2\pi,$$

and assume that the initial data $f(x)$ and the coefficients p and q of the operator L belong to V_N. Furthermore, we associate a grid function $\mathbf{u}_N$ with the function $u_N(x) \in V_N$ defined by

$$u_N(x) = \sum_{\omega=-N/2+1}^{N/2-1} \tilde{u}_N(\omega) e^{i\omega x},$$

where

$$\tilde{u}_N(\omega) = \frac{h_N}{\sqrt{2\pi}} \sum_{j=0}^{N-1} e^{-i\omega j h_N} [\mathbf{u}_N]_j, \quad h_N = \frac{2\pi}{N}.$$

If we discretize the operator L by an $N \times N$ matrix L_N and compute $\mathbf{v}_N = L_N \mathbf{u}_N$, then high-frequency components of v_N are lost due to aliasing.

We can avoid this loss of information using a finer grid. Given a grid function $\mathbf{f}$ defined on an N-point uniform grid of the form (2.1), the grid function $\mathbf{f}_M$, for $M > N$, is defined by interpolating the values of $\mathbf{f}$ on the finer M-point grid; i.e.,

$$[\mathbf{f}_M]_j = \frac{1}{\sqrt{2\pi}} \sum_{\omega=-N/2+1}^{N/2-1} e^{i\omega 2\pi j/M} \tilde{f}(\omega), \quad j = 0, \ldots, M-1,$$

where

$$\tilde{f}(\omega) = \left[\frac{h_N}{\sqrt{2\pi}} \sum_{k=0}^{N-1} e^{-i\omega k h_N} [\mathbf{f}]_k \right].$$

If $M \geq 2N$, and the coefficients of L belong to V_N, then $\mathbf{v}_M = L_M \mathbf{u}_M$ retains the high-frequency components of $L u_N$.

In the following algorithm, we compute bounds on $\langle u_N, e^{-Lt} u_N \rangle$ using a K-point Gaussian rule and a $(K+1)$-point Gauss-Radau rule. Both quadrature rules are obtained by applying the symmetric Lanczos algorithm to the matrix L_{M_K} with initial vector $\mathbf{u}_{M_K}$, where $M_K = 2^K N$. Because the coefficients of L belong to V_N, we do not need to work with L_{M_K} explicitly at each iteration; we can instead begin with a $2N$-point grid and refine after each iteration.

After K iterations of the Lanczos algorithm, the $K \times K$ tridiagonal matrix J_K defined in (2.6) is obtained. The Gaussian quadrature approximation of $\langle u_N, e^{-Lt} u_N \rangle$ is given by $\mathbf{u}_N^T \mathbf{u}_N [\exp(-J_K t)]_{11}$. Then, J_K is extended to a matrix J_{K+1} that has an eigenvalue that approximates the smallest eigenvalue of L. For details on the extension of J_K, see [3]. Finally, the Gauss-Radau approximation of $\langle u_N, e^{-Lt} u_N \rangle$ is given by $\mathbf{u}_N^T \mathbf{u}_N [\exp(-J_{K+1} t)]_{11}$.

We now describe the algorithm in full detail.

Algorithm 2.1 Given a real-valued grid function $\mathbf{u}_N$ defined on an N-point uniform grid (2.1), a self-adjoint differential operator L of the form (1.1) and a scalar t, the following algorithm computes bounds z_1 and z_2 on $\mathbf{u}_{M_K}^T \exp[-L_{M_K} t]\mathbf{u}_{M_K}$, where K is the number of Gaussian quadrature nodes to be computed and $M_K = 2^K N$.

$\beta_0 = \|\mathbf{u}_N\|_2$
$\mathbf{f}_N^1 = \mathbf{u}_N/\beta_0$
$\mathbf{f}_N^0 = \mathbf{0}$
$M = 2N$
for $j = 1, \ldots, K$
  $\mathbf{h}_M^j = L_M \mathbf{f}_M^j$
  $\alpha_j = [\mathbf{f}_M^j]^T \mathbf{h}_M^j$
  $\mathbf{r}_M^j = \mathbf{h}_M^j - \alpha_j \mathbf{f}_M^j - \gamma_{j-1}\mathbf{f}_M^{j-1}$
  $\beta_j = \|\mathbf{r}_M^j\|_2$
  $\mathbf{f}_M^{j+1} = \mathbf{r}_M^j/\beta_j$
  $M = 2M$
end
Let J_K be the $K \times K$ matrix defined by (2.6)
$z_1 = h\beta_0^2[\exp(-J_K t)]_{11}$
Let a be an approximation to the smallest eigenvalue of L
Solve $(J_K - aI)\delta = \beta_K^2 \mathbf{e}_K$
$\alpha_{K+1} = a + \delta_K$
Let J_{K+1} be the matrix obtained from J_K by adding
  α_{K+1} to the diagonal, β_K to the superdiagonal
  and β_K to the subdiagonal
$z_2 = h\beta_0^2[\exp(-J_{K+1} t)]_{11}$

The approximation to the smallest eigenvalue of L, required by the Gauss-Radau rule, can be obtained by applying the symmetric Lanczos algorithm to a discretization of L with initial vector $\begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}^T$. This choice of initial vector is motivated by the fact that, as $|\omega|$ increases, $Le^{i\omega x} \sim p(x)|\omega|^2 e^{i\omega x}$. Therefore, in order to obtain a function u for which $\|Lu\|/\|u\|$ is as small as possible, it is a good heuristic to avoid high frequencies.

Now, we can describe an algorithm for computing the approximate solution $\tilde{u}(x,t)$ of (1.2), (1.3), (1.4) at times $\Delta t, 2\Delta t, \ldots, n\Delta t = t_{final}$. At each time step, we compute approximate Fourier components of the solution by using the *polar decomposition*

$$\mathbf{u}^T f(A)\mathbf{v} = \frac{1}{4}\left[(\mathbf{u}+\mathbf{v})^T f(A)(\mathbf{u}+\mathbf{v}) - (\mathbf{u}-\mathbf{v})^T f(A)(\mathbf{u}-\mathbf{v})\right] \tag{2.7}$$

to express the Fourier components in terms of quadratic forms, which in turn are approximated using Algorithm 2.1.

To avoid complex arithmetic, we use the grid functions

$$[\hat{\mathbf{c}}_\omega]_j = \frac{1}{\sqrt{\pi}}\cos(\omega j h), \quad j = 0, \ldots, N-1, \quad \omega = 1, \ldots, N/2-1,$$

$$[\hat{\mathbf{s}}_\omega]_j = \frac{1}{\sqrt{\pi}}\sin(\omega j h), \quad j = 0, \ldots, N-1, \quad \omega = 1, \ldots, N/2-1,$$

and

$$[\hat{\mathbf{e}}_0]_j = \frac{1}{\sqrt{2\pi}}, \quad j = 0, \ldots, N-1.$$

Also, it will be assumed that all bounds on quantities of the form $\mathbf{u}^T \exp(-A\Delta t)\mathbf{u}$, for any matrix A and vector $\mathbf{u}$, are computed using Algorithm 2.1.

Algorithm 2.2 Given a grid function $\mathbf{f}_N$ representing the initial data $f(x)$ on a uniform N-point grid of the form (2.1), a final time t_{final} and a timestep Δt such that $t_{final} = n\Delta t$ for some integer n, the following algorithm computes an approximation $\tilde{\mathbf{u}}_j^{n+1}$ to the solution $u(x,t)$ of (1.2), (1.3), (1.4) evaluated at each gridpoint $x_j = jh$ for $j = 0, 1, \ldots, N-1$ with $h = 2\pi/N$ and times $t_n = n\Delta t$ for $n = 0, 1, \ldots, t_{final}/\Delta t$.

$\tilde{\mathbf{u}}^0 = \mathbf{f}_N$
for $n = 0, 1, \ldots, t_{final}/\Delta t$ **do**
  Choose a nonzero constant δ
  $\mathbf{v} = \hat{\mathbf{e}}_0 - \delta\tilde{\mathbf{u}}^n$
  $\mathbf{w} = \hat{\mathbf{e}}_0 + \delta\tilde{\mathbf{u}}^n$
  Compute bounds e_{11} and e_{12} for $\mathbf{v}_{M_K}^T \exp[-L_{M_K}\Delta t]\mathbf{v}_{M_K}$
  Compute bounds e_{21} and e_{22} for $\mathbf{w}_{M_K}^T \exp[-L_{M_K}\Delta t]\mathbf{w}_{M_K}$
  Let $\hat{\mathbf{u}}_0^{n+1} = (e_{2i} - e_{1j})/(4\delta)$ where i and j
    are chosen to minimize error in $\hat{\mathbf{u}}_0^{n+1}$
  for $\omega = 1, \ldots, N/2 - 1$
    $\mathbf{v} = \hat{\mathbf{c}}_\omega - \delta\tilde{\mathbf{u}}^n$
    $\mathbf{w} = \hat{\mathbf{c}}_\omega + \delta\tilde{\mathbf{u}}^n$
    Compute bounds c_{11} and c_{12} for $\mathbf{v}_{M_K}^T \exp[-L_{M_K}\Delta t]\mathbf{v}_{M_K}$
    Compute bounds c_{21} and c_{22} for $\mathbf{w}_{M_K}^T \exp[-L_{M_K}\Delta t]\mathbf{w}_{M_K}$
    $\mathbf{v} = \hat{\mathbf{s}}_\omega - \delta\tilde{\mathbf{u}}^n$
    $\mathbf{w} = \hat{\mathbf{s}}_\omega + \delta\tilde{\mathbf{u}}^n$
    Compute bounds s_{11} and s_{12} for $\mathbf{v}_{M_K}^T \exp[-L_{M_K}\Delta t]\mathbf{v}_{M_K}$
    Compute bounds s_{21} and s_{22} for $\mathbf{w}_{M_K}^T \exp[-L_{M_K}\Delta t]\mathbf{w}_{M_K}$
    Let $c_\omega = (c_{2i} - c_{1j})/(4\delta)$ where i and j
      are chosen to minimize error in c_ω
    Let $s_\omega = (s_{2i} - s_{1j})/(4\delta)$ where i and j
      are chosen to minimize error in s_ω
    $\hat{\mathbf{u}}_\omega^{n+1} = c_\omega + is_\omega$
    $\hat{\mathbf{u}}_{-\omega}^{n+1} = c_\omega - is_\omega$
  end
  $\tilde{\mathbf{u}}^{n+1} = T^{-1}\hat{\mathbf{u}}^{n+1}$ (inverse discrete Fourier transform)
end

In computing quantities of the form $\mathbf{u}^T f(A)\mathbf{v}$ using the polar decomposition (2.7), this algorithm actually computes $\frac{1}{\delta}[\mathbf{u}^T f(A)(\delta\mathbf{v})]$ where the scalar δ is chosen at the beginning of each time step. On the one hand, smaller values of δ are desirable because the quadrature error is reduced when the vector $\mathbf{w}$ in $\mathbf{w}^T f(A)\mathbf{w}$ is an approximate eigenvector of the matrix A. However, δ should not be chosen to be so small that $\mathbf{u}$ and $\mathbf{u} \pm \delta\mathbf{v}$ are virtually indistinguishable for the given precision. In practice, it is wise to choose δ to be proportional to $\|\tilde{\mathbf{u}}_n\|$ when computing $\tilde{\mathbf{u}}_{n+1}$. This explains why δ is chosen at the beginning of each time step in the preceding algorithm.

Various strategies can be used to determine whether the upper or lower bound on each integral should be used in computing the approximation to each component of the solution. For example, a Gauss-Radau rule with an appropriate choice of prescribed node can be compared with the approximation computed using a Gaussian rule in order to estimate its accuracy. Alternatively, Gauss-Kronrod rules can be used from the previously constructed Gaussian rules to estimate the accuracy of each bound; for details see [2].

ETNA
Kent State University
etna@mcs.kent.edu

2.3. Convergence Analysis. We now prove that Algorithm 2.2 is convergent. The approach is analogous to that used to prove convergence for finite-difference schemes. We will denote by $\tilde{S}(\Delta t, \Delta x; f)$ the result of applying Algorithm 2.2 to the function $f(x)$ using a discretization of space and time with uniform spacings Δx and Δt, respectively.

2.3.1. Consistency. First, we will prove that the approximate Fourier components of the solution at time Δt computed by Algorithm 2.2, using a K-point Gaussian quadrature rule, converge to the corresponding Fourier components of the exact solution as $\Delta t \to 0$ at a rate of $O(\Delta t^{2K})$. In order to analyze the quadrature error for the integrand $f(\lambda) = \exp[-\lambda t]$, we first need to consider the case $f(\lambda) = \lambda^j$.

LEMMA 2.1. *Let A be an $n \times n$ symmetric positive definite matrix. Let $\mathbf{u}$ and $\mathbf{v}$ be fixed vectors, and define $\mathbf{u}_\delta = \mathbf{u} + \delta\mathbf{v}$. For j a positive integer, let $\tilde{g}_j(\delta)$ be defined by*

$$\tilde{g}_j(\delta) = \frac{1}{2}\mathbf{e}_1^T T_\delta^j \mathbf{e}_1 \|\mathbf{u}_\delta\|_2^2,$$

where T_δ is the Jacobi matrix produced by the symmetric Lanczos iteration applied to A with starting vector $\mathbf{u}_\delta$. Then, for some η satisfying $0 < \eta < \delta$,

$$\begin{aligned} \frac{\tilde{g}_j(\delta) - \tilde{g}_j(-\delta)}{2\delta} &= \mathbf{u}^T A^j \mathbf{v} + \\ &\quad \sum_{k=K}^{j-K} \mathbf{e}_1^T \left[T^k X^T - X^T A^k\right]' \mathbf{r}\mathbf{e}_K^T T^{j-k-1}\mathbf{e}_1 \mathbf{u}^T\mathbf{u} + \\ &\quad \frac{\delta^2}{6}\left[\sum_{k=K}^{j-K} \mathbf{e}_1^T \left[T_\delta^k X_\delta^T - X_\delta^T A^k\right]' \mathbf{r}_\delta \mathbf{e}_K^T T_\delta^{j-k-1}\mathbf{e}_1 \mathbf{u}_\delta^T\mathbf{u}_\delta\right]''\Bigg|_{\delta=\eta} \end{aligned} \tag{2.8}$$

Proof. See Appendix A.1. ∎

The following corollary summarizes the integrands for which Gaussian quadrature is exact.

COROLLARY 2.2. *Under the assumptions of the lemma,*

$$\frac{\tilde{g}_j(\delta) - \tilde{g}_j(-\delta)}{2\delta} = \mathbf{u}^T A^j \mathbf{v},$$

for $0 \le j < 2K$.

Lemma 2.1 can be used to show consistency of the computed solution with $e^{-L_N t}\mathbf{f}_N$, but we need to show consistency with the exact solution of the underlying PDE, $u(x,t) = e^{-Lt}f(x)$. Therefore, we need the following result to relate the discrete inner products employed by Algorithm 2.1 to the continuous inner products that describe the frequency components of $u(x,t)$. Recall the definition of V_N from the beginning of Section 2.2.

LEMMA 2.3. *Let $f \in V_N$ and L be an m-th order differential operator of the form (1.1) such that the coefficients p and q belong to V_N. Then $Lf \in V_{2N}$ and*

$$\langle \hat{e}_\omega, Lf\rangle = \hat{\mathbf{e}}_\omega^T L_M \mathbf{f}_M, \quad \omega = -M/2+1, \ldots, M/2-1, \tag{2.9}$$

for $M = 2^j N$, where j is a positive integer.

Proof. See Appendix A.2. ∎

We can now bound the local truncation error in each Fourier component of the computed solution.

THEOREM 2.4. *Let L be a self-adjoint m-th order positive definite differential operator with coefficients in V_N, and let $f(x) \in V_N$. Then Algorithm 2.2 is consistent; i.e.*

$$\langle \hat{e}_\omega, \tilde{S}(\Delta t, \Delta x; f) - \exp[-L\Delta t]f\rangle = O(\Delta t^{2K}),$$

ETNA
Kent State University
etna@mcs.kent.edu

for $\omega = -N/2+1, \dots, N/2-1$.

Proof. Let $\tilde{g}(\delta)$ be the function from Lemma 2.1 with $A = L_{M_K}$, $\mathbf{u} = \mathbf{c}_\omega$ and $\mathbf{v} = \mathbf{f}$. Furthermore, denote the entries of T_δ by

$$T_\delta = \begin{bmatrix} \alpha_1(\delta) & \beta_1(\delta) & & & \\ \beta_1(\delta) & \alpha_2(\delta) & \beta_2(\delta) & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{K-2}(\delta) & \alpha_{K-1}(\delta) & \beta_{K-1}(\delta) \\ & & & \beta_{K-1}(\delta) & \alpha_K(\delta) \end{bmatrix}.$$

Finally, let $\beta_0(\delta) = \|\mathbf{u}_\delta\|_2$ and $\beta_K(\delta) = \|\mathbf{r}_\delta\|_2$. Then, by Lemmas 2.1 and 2.3,

$$\begin{aligned}
\langle \hat{c}_\omega, u(\cdot,\Delta t)\rangle - c_\omega &= \sum_{j=0}^{\infty} \frac{\Delta t^j}{j!} \left\{ \langle \hat{c}_\omega, L^j f\rangle - \frac{\tilde{g}_j(\delta) - \tilde{g}_j(-\delta)}{2\delta} \right\} \\
&= \sum_{j=0}^{\infty} \frac{\Delta t^j}{j!} \left\{ \langle \hat{c}_\omega, L^j f\rangle - \mathbf{c}_\omega^T L_{M_K}^j \mathbf{f} + \right. \\
&\quad \left. \sum_{k=K}^{j-K} \mathbf{e}_1^T \frac{d}{d\delta} \left[T_\delta^k X_\delta^T - X_\delta^T L_{M_K}^k \right] \Big|_{\delta=0} \mathbf{r} \mathbf{e}_K^T T^{j-k-1} \mathbf{e}_1 \right\} + \\
&\quad O(\delta \Delta t^{2K}) \\
&= \frac{\Delta t^{2K}}{(2K)!} \mathbf{e}_1^T \frac{d}{d\delta} \left[T_\delta^K X_\delta^T - X_\delta^T L_{M_K}^K \right] \Big|_{\delta=0} \mathbf{r} \mathbf{e}_K^T T^{K-1} \mathbf{e}_1 + \\
&\quad O(\delta \Delta t^{2K}) \\
&= \frac{\Delta t^{2K}}{(2K)!} \mathbf{e}_1^T \frac{d}{d\delta} \left[\sum_{j=0}^{K-1} T_\delta^j \mathbf{e}_K \mathbf{r}_\delta^T L_{M_K}^{K-j-1} \right] \Bigg|_{\delta=0} \mathbf{r} \mathbf{e}_K^T T^{K-1} \mathbf{e}_1 + \\
&\quad O(\delta \Delta t^{2K}) \\
&= \frac{\Delta t^{2K}}{(2K)!} \mathbf{e}_1^T \frac{d}{d\delta} \left[T_\delta^{K-1} \mathbf{e}_K \mathbf{r}_\delta^T \right] \Big|_{\delta=0} \mathbf{r} \mathbf{e}_K^T T^{K-1} \mathbf{e}_1 + O(\delta \Delta t^{2K}) \\
&= \frac{1}{2} \frac{\Delta t^{2K}}{(2K)!} \frac{d}{d\delta} \left[\|\mathbf{r}_\delta\| \mathbf{e}_1^T T_\delta^{K-1} \mathbf{e}_K \right]^2 \Big|_{\delta=0} + O(\delta \Delta t^{2K}) \\
&= \frac{1}{2} \frac{\Delta t^{2K}}{(2K)!} \frac{d}{d\delta} \left(\beta_0(\delta) \cdots \beta_K(\delta) \right)^2 \Big|_{\delta=0} + O(\delta \Delta t^{2K}) \\
&= O(\Delta t^{2K}). \qquad (2.10)
\end{aligned}$$

A similar argument applies to $\langle \hat{s}_\omega, u(\cdot,\Delta t)\rangle - s_\omega$. □

The preceding result indicates that even if low-dimensional Krylov subspaces are used to approximate each component of the solution, high-order temporal accuracy can be obtained.

On the other hand, it is important to note that the spatial error depends heavily on the smoothness of the coefficients of the operator L, as well as the initial data $f(x)$. The following result quantifies the effect of the smoothness of the coefficients.

THEOREM 2.5. *Let* $p(x) > 0$ *and* $q(x) \geq 0$ *belong to* V_N*, and let*

$$\bar{p} = \operatorname{Avg} p = \frac{1}{2\pi} \int_0^{2\pi} p(x)\, dx, \quad \bar{q} = \operatorname{Avg} q.$$

ETNA
Kent State University
etna@mcs.kent.edu

Let L be defined as in (1.1), and assume that, $L = C + V$, where

$$C = -\bar{p}\partial_{xx} + \bar{q}.$$

Let $\phi_\omega(x) = \frac{1}{\sqrt{2\pi}} e^{i\omega x}$, and let $\tilde{I}_\omega(\Delta t)$ be the approximation of

$$I_\omega(\Delta t) = \langle \phi_\omega, e^{-L\Delta t}\phi_\omega \rangle$$

computed using Algorithm 2.1 with an N-point grid of the form (2.1) and K Gaussian quadrature nodes. Then, for $|\omega| < N/2$,

$$I_\omega(\Delta t) - \tilde{I}_\omega(\Delta t) = O(\Delta t^{2K} \|V_{M_K}\|).$$

Proof. See Appendix A.3. □

Note that this result implies that Krylov subspace spectral methods reduce to the Fourier method in the case where L has constant coefficients.

2.3.2. Stability. We now examine the stability of this time-stepping algorithm. For simplicity, we only consider the case where the $K = 1$; that is, we are using a one-node Gaussian rule for each Fourier component.

THEOREM 2.6. *Let the differential operators L, C and V be defined as in Theorem 2.5, and let $f \in V_N$. Let $K = 1$ in Algorithm 2.2, and assume that the algorithm uses only the bounds obtained from Algorithm 2.1 by Gaussian quadrature. Then, in the limit as $\delta \to 0$, the approximate solution $\tilde{S}(\Delta t, \Delta x; f)$ to (1.2), (1.3), (1.4) computed by one time step in Algorithm 2.2 is given by*

$$\tilde{S}(\Delta t, \Delta x; f) = e^{-C\Delta t} P_N (I - \Delta t V) f,$$

where P_N is the orthogonal projection onto V_N.

Proof. We use the notation of Algorithm 2.2. First, we note that

$$\lim_{\delta\to 0} c_\omega = \frac{1}{2}\frac{d}{d\delta}\left[(\hat{\mathbf{c}}_\omega + \delta\mathbf{u}_n)^T(\hat{\mathbf{c}}_\omega + \delta\mathbf{u}_n)\exp[-\Delta t\alpha_\omega(\delta)]\right]\Bigg|_{\delta=0}, \tag{2.11}$$

where

$$\alpha_\omega(\delta) = \frac{(\hat{\mathbf{c}}_\omega + \delta\mathbf{u}_n)^T L_{M_1}(\hat{\mathbf{c}}_\omega + \delta\mathbf{u}_n)}{(\hat{\mathbf{c}}_\omega + \delta\mathbf{u}_n)^T(\hat{\mathbf{c}}_\omega + \delta\mathbf{u}_n)}. \tag{2.12}$$

A similar statement applies to s_ω and $\hat{\mathbf{u}}_0^{n+1}$. The result then follows from a von Neumann stability analysis of the approximate solution obtained from the limits of c_ω, s_ω and $\hat{\mathbf{u}}_0^{n+1}$ as $\delta \to 0$, which can be computed by differentiating expressions such as (2.11) and (2.12) analytically. □

Because $V \equiv 0$ when the operator L has constant coefficients, the preceding theorem indicates that stability is dependent on the variation in the coefficients, not their magnitude, as is the case with explicit finite-difference methods. In fact, it can be shown that, if the coefficients of L are sufficiently smooth, then Algorithm 2.2, under the assumptions of the theorem, is stable regardless of the time step Δt. Stability will be discussed further in an analysis that will be presented in [13].

ETNA
Kent State University
etna@mcs.kent.edu

2.3.3. Convergence. We are now ready to state and prove the principal result of this paper. As with the Lax-Richtmyer Equivalence Theorem for finite-difference methods, the consistency and stability of Algorithm 2.2 can be used to prove that it is also convergent.

THEOREM 2.7. *Let $u(x,t)$ be the solution of (1.2), (1.3), (1.4), where L is a self-adjoint positive definite differential operator with coefficients in V_N and the initial data $f(x)$ belongs to V_N. Let the differential operators C and V be defined as in Theorem 2.5. Furthermore, assume that the Fourier coefficients $\{\hat{u}(\omega,\Delta t)\}$ of $u(x,\Delta t) = \exp[-L\Delta t]f(x)$ satisfy an estimate*

$$|\hat{u}(\omega,\Delta t)| \leq \frac{C}{|\omega|^M}, \quad \omega \neq 0, \quad 0 \leq \Delta t \leq t_{final}, \quad M > 1.$$

Let $\tilde{u}(x,t_{final})$ be the approximate solution computed by Algorithm 2.2. If

$$\lim_{\Delta x,\Delta t\to 0} \frac{\Delta x^{M-1}}{\Delta t} = 0, \quad \Delta x = \frac{2\pi}{N},$$

and Δt satisfies

$$\left\| e^{-C\Delta t} P_N (I - \Delta t V) P_N \right\|_{L^2} < 1, \tag{2.13}$$

then Algorithm 2.2, in conjunction with Algorithm 2.1 using $K = 1$ Gaussian quadrature nodes, is convergent; i.e.

$$\lim_{\Delta x,\Delta t\to 0} \|\tilde{u}(\cdot,t_{final}) - u(\cdot,t_{final})\| = 0.$$

Proof. Let $e_n(x) = \tilde{u}(x,t_n) - u(x,t_n)$. If we choose the parameter δ sufficiently small in Algorithm 2.2, then it follows from Theorems 2.4 and 2.6 that

$$\begin{aligned}
\|e_{n+1}\| &= \|\tilde{u}(\cdot,t_{n+1}) - u(\cdot,t_{n+1})\| \\
&= \|\tilde{S}(\Delta t,\Delta x;\tilde{u}(\cdot,t_n)) - \exp[-L\Delta t]u(\cdot,t_n)\| \\
&\leq \|\tilde{S}(\Delta t,\Delta x;\tilde{u}(\cdot,t_n)) - \tilde{S}(\Delta t,\Delta x;u(\cdot,t_n))\| + \\
&\quad\ \|\tilde{S}(\Delta t,\Delta x;u(\cdot,t_n)) - \exp[-L\Delta t]u(\cdot,t_n)\| \\
&< \|e_n\| + C_1\Delta t^2 + C_2\Delta x^{M-1}
\end{aligned}$$

where the constants C_1 and C_2 are independent of Δx and Δt. We conclude that

$$\|e_n\| < n\left(C_1\Delta t^2 + C_2\Delta x^{M-1}\right) < \frac{t_{final}}{\Delta t}\left(C_1\Delta t^2 + C_2\Delta x^{M-1}\right)$$

which tends to zero as Δt, $\Delta x \to 0$ under the given assumptions. ∎

It is important to note that because L is positive definite, it is always possible to find $\Delta t > 0$ so that the stability condition (2.13) holds.

2.4. Practical Implementation. A companion paper [12] discusses practical implementation of Algorithms 2.1 and 2.2 in detail, but we highlight the main implementation issues here.

2.4.1. Parameter Selection. We now discuss how one can select two key parameters in the algorithm: the number of quadrature nodes K and the time step Δt. While it is obviously desirable to use a larger number of quadrature nodes, various difficulties can arise in addition to the expected computational expense of additional Lanczos iterations. As is well known, the Lanczos method suffers from loss of orthogonality of the Lanczos vectors, and this vulnerability increases with the number of iterations since it tends to occur as Ritz pairs converge to eigenpairs (for details see [4]).

In order to choose an appropriate time step Δt, one can compute components of the solution using a Gaussian quadrature rule, and then extend the rule to a Gauss-Radau rule and compare the approximations, selecting a smaller Δt if the error is too large relative to the norm of the data. Alternatively, one can use the Gaussian rule to construct a Gauss-Kronrod rule and obtain a second approximation; for details see [2]. However, it is useful to note that the time step only plays a role in the last stage of the computation of each component of the solution. It follows that one can easily construct a representation of the solution that can be evaluated at *any* time, thus allowing a residual $\partial u/\partial t + Lu$ to be computed. This aspect of our algorithm is fully exploited in [12].

By estimating the error in each component, one can avoid unnecessary construction of quadrature rules. For example, suppose that a timestep Δt has been selected, and the approximate solution $\tilde{u}(x, \Delta t)$ has been computed using Algorithm 2.2. Before using this approximate solution to construct the quadrature rules for the next time step, we can determine whether the rules constructed using the initial data $f(x)$ can be used to compute any of the components of $\tilde{u}(x, 2\Delta t)$ by evaluating the integrand at time $2\Delta t$ instead of Δt. If so, then there is no need to construct new quadrature rules for these components. This idea is explored further in [12].

2.4.2. Improving Performance. Theorem 2.6 implies that Algorithm 2.2 yields greater accuracy if the coefficients of L are smoother. Therefore, it is advisable to use similarity transformations to “precondition” L so that it more closely resembles a constant-coefficient operator. Some unitary similarity transformations that can be used for this purpose will be discussed in [11].

It is easy to see that a straightforward implementation of Algorithm 2.2 is prohibitively expensive, as it employs the Lanczos algorithm $O(N)$ times per time step, with each application requiring at least $O(N)$ operations. This complexity can be reduced to $O(N)$ by exploiting the fact that the matrix L_N represents a differential operator, and that the initial vectors can be parametrized using the wave number. A practical implementation of Algorithm 2.2 can be found in [12].

The fact that the time step plays a limited role in the computation, and, in particular, is not used to construct the quadrature rules, implies that the computed components can easily be represented as simple continuous functions of t without using interpolation. The availability of such a representation is exploited in [12] to obtain an even more efficient algorithm. This representation can also be differentiated analytically with respect to t, which is also exploited in [12] to construct a straightforward procedure for deferred correction.

3. Numerical Results. In this section we will display numerical results comparing the performance of Krylov subspace spectral methods with that of other numerical methods for problems of the form (1.2) as well as for more general problems.

3.1. Construction of Test Cases. In many of the following experiments, it is necessary to construct functions of a given smoothness. To that end, we rely on the following result (see [7]):

THEOREM 3.1. (GUSTAFSSON, KREISS, OLIGER) *Let $f(x)$ be a 2π-periodic function and assume that its pth derivative is a piecewise C^1 function. Then,*

$$|\hat{f}(\omega)| \leq constant/(|\omega|^{p+1} + 1). \tag{3.1}$$

Based on this result, the construction of a C^{p+1} function $f(x)$ proceeds as follows:

1. For each $\omega = 1, \dots, N/2-1$, choose the discrete Fourier coefficient $\hat{f}(\omega)$ by setting $\hat{f}(\omega) = (u+iv)/|\omega^{p+1}+1|$, where u and v are random numbers uniformly distributed on the interval $(0,1)$.
2. For each $\omega = 1, \dots, N/2-1$, set $\hat{f}(-\omega) = \overline{\hat{f}(\omega)}$.
3. Set $\hat{f}(0)$ equal to any real number.
4. Set $f(x) = \frac{1}{\sqrt{2\pi}} \sum_{|\omega|<N/2} \hat{f}(\omega) e^{i\omega x}$.

In the following test cases, coefficients and initial data are constructed so that their third derivatives are piecewise C^1, unless otherwise noted.

We will now introduce some differential operators and functions that will be used in a number of the experiments described in this section. As most of these functions and operators are randomly generated, we will denote by $R_1, R_2, \dots$ the sequence of random numbers obtained using MATLAB's random number generator `rand` after setting the generator to its initial state. These numbers are uniformly distributed on the interval $(0,1)$.

- We will make use of a two-parameter family of functions defined on the interval $[0, 2\pi]$. First, we define

$$f^0_{j,k}(x) = \mathrm{Re}\left\{ \sum_{|\omega|<N/2,\omega\neq 0} \hat{f}_j(\omega)(1+|\omega|)^{k+1} e^{i\omega x} \right\}, \quad j,k = 0,1,\dots,$$

where

$$\hat{f}_j(\omega) = R_{jN+2(\omega+N/2)-1} + iR_{jN+2(\omega+N/2)}.$$

The parameter j indicates how many functions have been generated in this fashion since setting MATLAB's random number generator to its initial state, and the parameter k indicates how smooth the function is. Figure 3.1 shows selected functions from this collection.

In many cases, it is necessary to ensure that a function is positive or negative, so we define the translation operators E^+ and E^- by

$$E^+ f(x) = f(x) - \min_{x\in[0,2\pi]} f(x) + 1, \tag{3.2}$$

$$E^- f(x) = f(x) - \max_{x\in[0,2\pi]} f(x) - 1. \tag{3.3}$$

- Some experiments will involve the one-parameter family of randomly generated self-adjoint differential operators

$$L_k = \partial_x \left(E^- f_{0,k} \partial_x \right) + E^+ f_{1,k}, \quad k = 0,1,\dots.$$

where the operators E^+ and E^- were defined in (3.2), (3.3).

Many of the experiments described in this section are intended to illustrate the convergence behavior of Algorithm 2.2, with certain variations, on various problems.

3.2. Parabolic Problems. For our first example, we will solve the problem

$$\frac{\partial u}{\partial t}(x,t) + L_3 u(x,t) = 0, \quad 0 < x < 2\pi, \quad t > 0, \tag{3.4}$$

$$u(x,0) = E^+ f_{0,3}(x), \quad 0 < x < 2\pi, \tag{3.5}$$

ETNA
Kent State University
etna@mcs.kent.edu

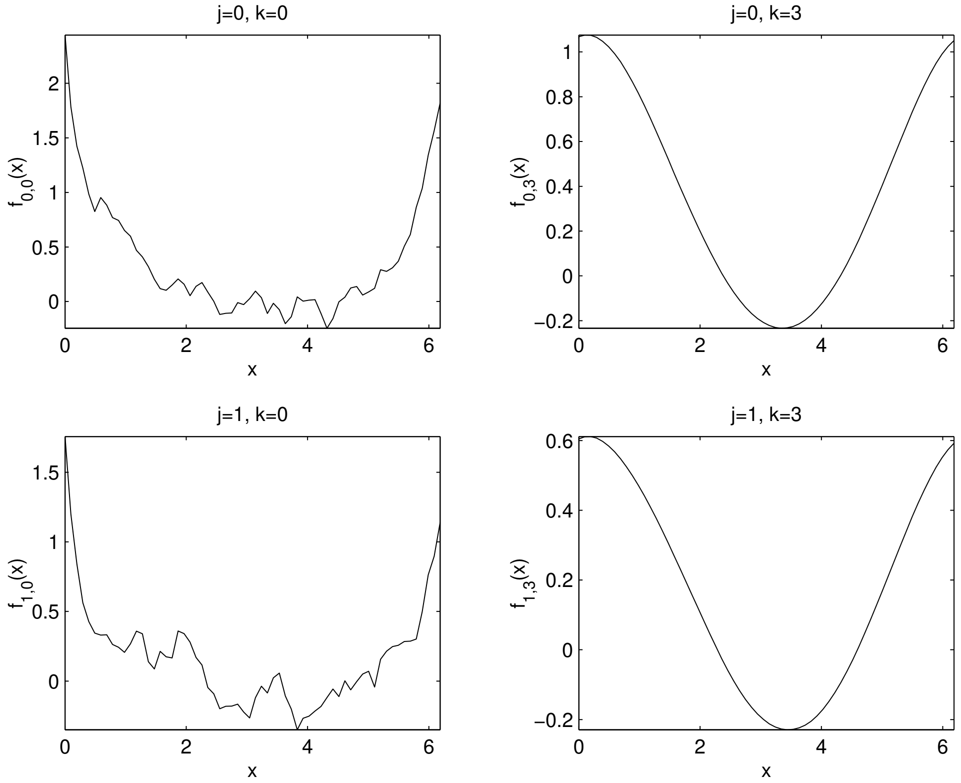

FIG. 3.1. *Functions from the collection $f_{j,k}(x)$, for selected values of j and k.*

$$u(x,t) = u(x + 2\pi, t), \quad t > 0 \tag{3.6}$$

using the following methods:

- The Crank-Nicolson method with central differencing
- Algorithm 2.2, with 2 nodes determined by Gaussian quadrature
- Algorithm 2.2, with 2 nodes determined by Gaussian quadrature and one additional prescribed node. The prescribed node is obtained by estimating the smallest eigenvalue of L using the symmetric Lanczos algorithm.

In all cases, $N = 64$ gridpoints are used. For $j = 0, \ldots, 6$, we compute an approximate solution $u^{(j)}(x,t)$ at $t = 1$, using $\Delta t = 2^{-j}$. Figure 3.2 shows estimates of the relative error in $u^{(j)}(x,1)$ for $j = 0, \ldots, 5$. Note the significant benefit of the prescribed node in the Gauss-Radau rule.

3.2.1. Varying Spatial Resolution. In Figure 3.3 we illustrate the benefit of using component-specific Krylov subspace approximations. We solve the problem (3.4), (3.5), (3.6) using the following methods:

- A two-stage, third-order scheme described by Hochbruck and Lubich in [8] for solving systems of the form $\mathbf{y}' = A\mathbf{y} + \mathbf{b}$, where, in this case, $\mathbf{b} = 0$ and A is an $N \times N$ matrix that discretizes the operator L_3. The scheme involves multiplication of vectors by $\varphi(\gamma h A)$, where γ is a parameter (chosen to be $\frac{1}{2}$), h is the step size, and $\varphi(z) = (e^z - 1)/z$. The computation of $\varphi(\gamma h A)\mathbf{v}$, for a given vector $\mathbf{v}$, is accomplished by applying the Lanczos iteration to A with initial vector $\mathbf{v}$ to obtain

ETNA
Kent State University
etna@mcs.kent.edu

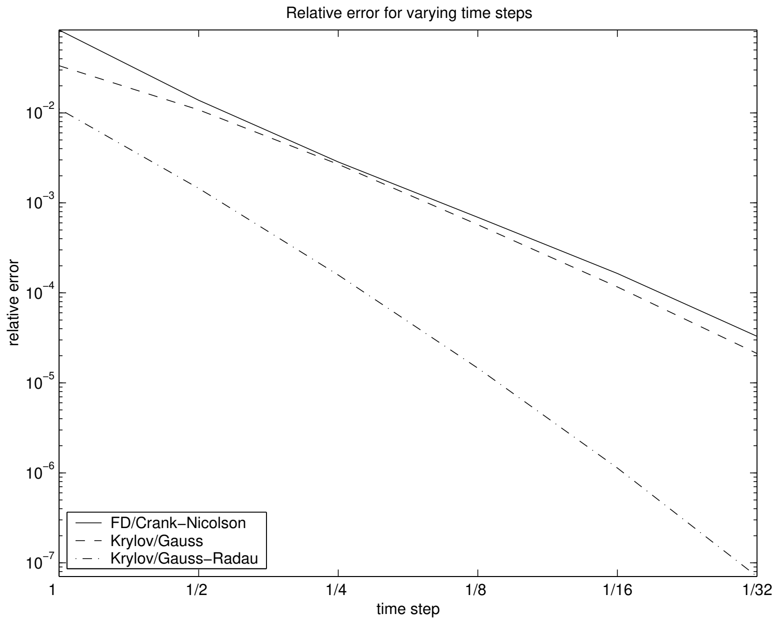

FIG. 3.2. *Estimates of relative error in the computed solution $\tilde{u}(x,t)$ of (3.4), (3.5), (3.6) at $t = 1$. Solutions are computed using finite differences with Crank-Nicolson (solid curve), Algorithm 2.2 with Gaussian quadrature (dashed curve), and Algorithm 2.2 with Gauss-Radau quadrature (dotted-dashed curve) with various time steps and $N = 64$ grid points.*

an approximation to $\varphi(\gamma h A)\mathbf{v}$ that belongs to the m-dimensional Krylov subspace $\mathcal{K}(A, \mathbf{v}, m) = \text{span}\{\mathbf{v}, A\mathbf{v}, A^2\mathbf{v}, \ldots, A^{m-1}\mathbf{v}\}$.

- Algorithm 2.2, with m nodes determined by Gaussian quadrature and one additional prescribed node. The prescribed node is obtained by estimating the smallest eigenvalue of L using the symmetric Lanczos algorithm.

We choose $m = 2$ in both cases, so that both algorithms perform the same number of matrix-vector multiplications during each time step. Note that, as N increases from 64 to 128, there is no impact on the accuracy of Algorithm 2.2; the curves corresponding to this method are virtually indistinguishable. On the other hand, this increase, which results in a stiffer system, reduces the time step at which the method from [8] begins to show reasonable accuracy.

This loss of accuracy can be explained by observing that for each component of the solution, a Krylov subspace spectral method implicitly constructs a polynomial $p(\lambda)$ that interpolates $e^{-\lambda \Delta t}$ for some Δt. The interpolation points are chosen in order to maximize the degree of a quadrature rule that is used to integrate $p(\lambda)$ with respect to the *component-dependent measure* $\alpha(\lambda)$ defined in (2.5). It is in this sense that the approximation of each component is optimal for that component.

The method from [8] effectively uses the same polynomial approximation of $e^{-\lambda \Delta t}$ for all components, resulting in a lower degree of accuracy. If the initial data is smooth, then this uniform approximation is still very accurate for computing low-frequency components of the solution, but as N increases, the computed solution includes more (erroneous) high-frequency components.

3.2.2. Convergence of Derivatives. Figure 3.4 shows the accuracy in each frequency component of the computed solution using various methods. This accuracy is measured by computing the relative difference in the first and second derivatives of approximate solutions

ETNA
Kent State University
etna@mcs.kent.edu

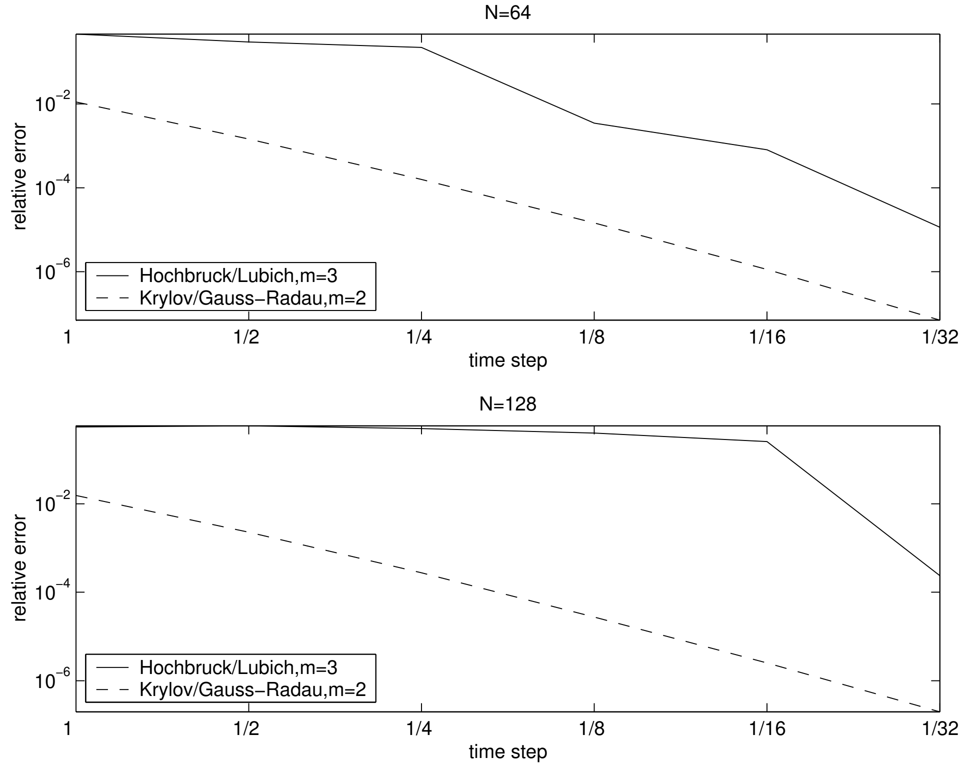

FIG. 3.3. *Estimates of relative error in the computed solution $\tilde{u}(x,t)$ of (3.4), (3.5), (3.6) at $t = 1$. Solutions are computed using the third-order method of Hochbruck and Lubich described in [8] using a Krylov subspace of dimension $m = 3$ (solid curve), and Algorithm 2.2 with a 2-point Gauss-Radau rule (dashed curve) with various time steps and $N = 64$ grid points (top plot) or $N = 128$ grid points (bottom plot).*

$\tilde{\mathbf{u}}_j$ and $\tilde{\mathbf{u}}_{j-1}$ to the problem (3.4), (3.5), (3.6). Each approximate solution $\tilde{\mathbf{u}}_j$ is computed using $\Delta t = 2^{-j}$, for $j = 0, \ldots, 6$, and $N = 64$ gridpoints. In other words, we are measuring the error in $\mathbf{u}_j$ using the H^1 and H^2 seminorms (see [9]), where

$$\|u\|_{H^r}^2 = \int_0^{2\pi} |u^{(r)}(x)|\, dx. \tag{3.7}$$

The methods used for the comparison are Crank-Nicholson with finite differencing, backward Euler with the Fourier method, and Gauss-Radau quadrature with two Gaussian quadrature nodes. As can easily be seen, Gauss-Radau quadrature provides more rapid convergence for both higher- and lower-frequency components than the other two methods. Gaussian quadrature with no prescribed nodes does not perform as well, since the lower bounds that it yields for each integral are not as sharp as the upper bounds obtained via Gauss-Radau quadrature.

3.3. Non-Self-Adjoint Problems. While the development of our algorithm relied on the assumption that L was self-adjoint, it can be shown that it works quite well in cases where L is not self-adjoint. In [5], Goodman, Hou and Tadmor study the stability of the unsmoothed Fourier method when applied to the problem

$$\frac{\partial u}{\partial t}(x,t) - \frac{\partial}{\partial x}\left(\sin(x)u(x,t)\right) = 0, \quad 0 < x < 2\pi, \quad t > 0, \tag{3.8}$$

ETNA
Kent State University
etna@mcs.kent.edu

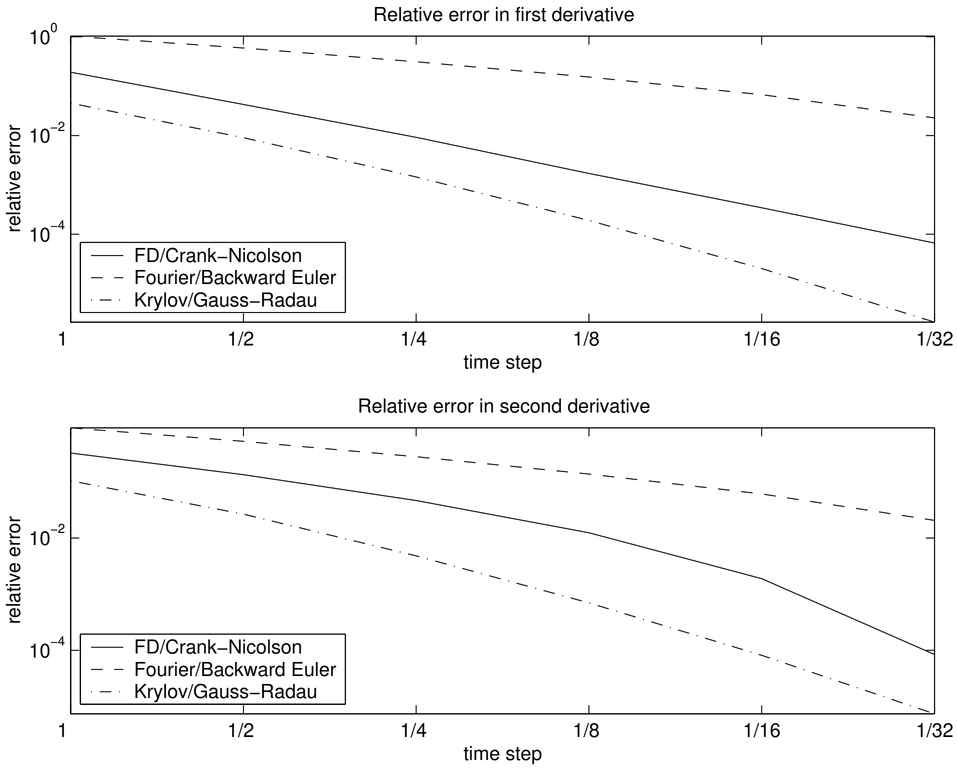

FIG. 3.4. *Relative error estimates in first and second derivatives of approximate solutions to (3.4), (3.5), (3.6), measured using the H^1 and H^2 seminorms, respectively. In all cases $N = 64$ gridpoints are used.*

$$u(x,0) = \frac{1}{\sqrt{2\pi}} \sum_{\omega=-N/2+1}^{N/2-1} e^{i\omega x} i\omega^{-3}, \quad 0 < x < 2\pi, \tag{3.9}$$

$$u(x,t) = u(x+2\pi,t), \quad t > 0. \tag{3.10}$$

Figure 3.5 compares the Fourier coefficients obtained using the Fourier method with those obtained using Gauss-Radau quadrature as in Algorithm 2.2. It is easy to see that using Algorithm 2.2 avoids the weak instability exhibited by the unsmoothed Fourier method. As noted in [5], this weak instability can be overcome by using a sufficiently large number of gridpoints, or by applying filtering techniques (see [1]) to remove high-frequency components that are contaminated by aliasing. Algorithm 2.2, by computing each Fourier component using an approximation to the solution operator that is tailored to that component, provides the benefit of smoothing, without the loss of resolution associated with filtering.

While the theory presented and cited in Section 2 is not applicable to the non-self-adjoint case, a plausible explanation can be given as to why Gaussian quadrature methods can still be employed for such problems. Each component of the solution is computed by approximating quantities of the form

$$f(\mathbf{u}) = \mathbf{u}^T \exp[-A\Delta t]\mathbf{u},$$

where $\mathbf{u}$ is an N-vector A is an $N \times N$ matrix that may or may not be symmetric. The approximation $\tilde{f}(\mathbf{u})$ of $f(\mathbf{u})$ takes the form

$$\tilde{f}(\mathbf{u}) = \mathbf{u}^T \left[\sum_{j=0}^{J} w_j e^{-\lambda_j \Delta t} A^j \mathbf{u} \right] = \mathbf{u}^T P_J(A) \mathbf{u},$$

and satisfies

$$f(\mathbf{u}) - \tilde{f}(\mathbf{u}) = \mathbf{u}^T \left(\sum_{k=2J}^{\infty} (-1)^k \frac{\Delta t^k}{k!} A^k \right) \mathbf{u},$$

due to the construction of the two sets of Lanczos vectors generated by the unsymmetric Lanczos iteration. In this sense, the high accuracy of Gaussian quadrature generalizes to the non-self-adjoint case. Each quantity $f(\mathbf{u})$ can be viewed as an Riemann-Stieltjes integral over a contour in the complex plane; the use of Gaussian quadrature to evaluate such integrals is discussed in [14].

It should be noted, however, that instability can still occur if the integrals are not computed with sufficient accuracy. Unlike the weak instability that occurs in the Fourier method, the remedy is not to use more gridpoints, but to ensure that the same components are computed with greater accuracy. This can be accomplished by choosing a smaller timestep or increasing the number of quadrature nodes, and both tactics have been successful with (3.8), (3.9) in practice.

4. Generalizations. This paper has focused primarily on the applicability of Krylov subspace spectral methods to the diffusion equation in one space dimension with periodic boundary conditions. However, as illustrated in the previous section, they are well suited to many other categories of problems, which we enumerate here.

1. Problems in higher space dimension: In [10] numerical results are presented for first-order wave equations and diffusion equations in two space dimensions, as well as discussion on how to use Krylov subspace spectral methods in any number of space dimensions.
2. Non-periodic boundary conditions: In [6] Krylov subspace spectral methods are applied to a problem with Dirichlet boundary conditions. More general discussion of other boundary conditions is contained in [10].
3. Second-order wave equations: Problems that contain higher-order derivatives in time, can be solved using Krylov subspace spectral methods very easily, because the computed solutions can be differentiated analytically with respect to time. This is exploited in [6] to solve the variable-speed wave equation in one space dimension. Results for two and three dimensions have been obtained and will be presented in an upcoming paper.

5. Conclusions. By reconsidering the role of numerical quadrature in Galerkin methods, we have succeeded in developing a class of numerical methods for solving the problem (1.2), (1.3), (1.4) that overcome some of the difficulties that variable-coefficient problems pose for traditional spectral methods. By using a low-order Krylov subspace approximation of the solution operator for each component instead of a single higher-order Krylov subspace approximation for all components, high-order accuracy and near-unconditional stability is attained.

Future work will be devoted to realizing further benefit by exploiting two key properties of these methods: first, that they are more accurate for problems with smoother coefficients, and second, that the components of the computed solution in the basis of trial functions can

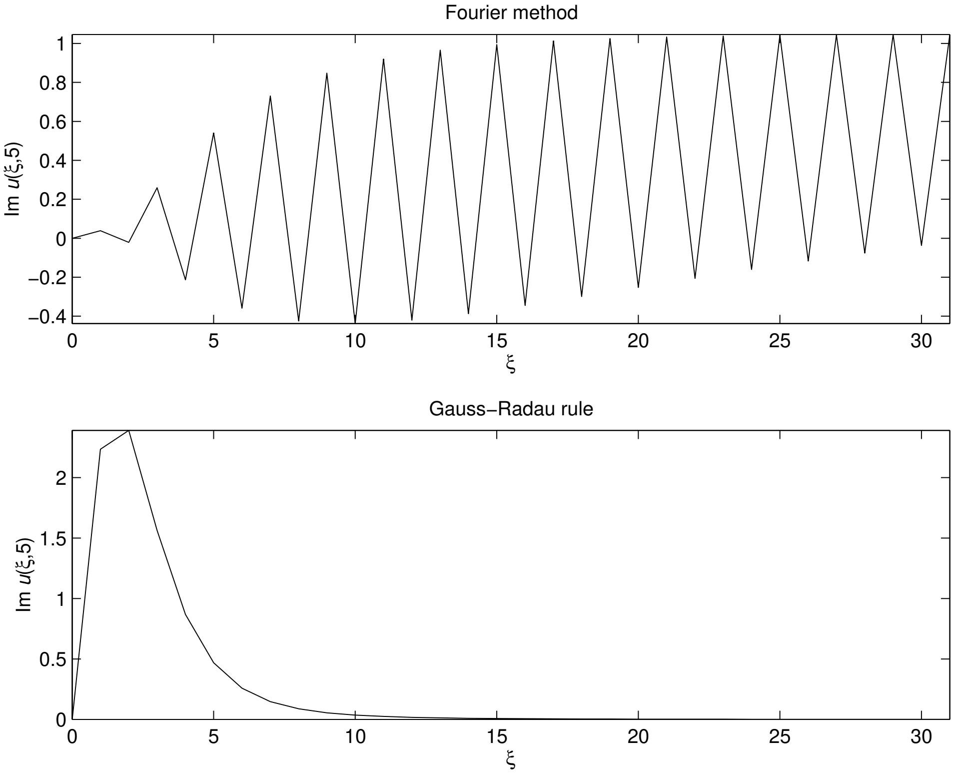

FIG. 3.5. *Fourier coefficients of the approximate solution $\tilde{u}(x, 5)$ of (3.8), (3.9), (3.10) computed using the Fourier method (top graph) and Algorithm 2.2 with Gauss-Radau quadrature (bottom graph) with $N = 64$ nodes and time step $\Delta t = 1/32$.*

be represented as continuous functions of t that have a reasonably simple structure. One goal is to combine methods for efficiently computing approximate eigenfunctions of L with Krylov subspace spectral methods to construct a continuous function that represents a highly accurate approximation of the exact solution $u(x, t)$ over as large a domain in (x, t)-space as possible, with less computational effort than that which traditional time-marching methods and subsequent interpolation would require. Such an approximation should yield useful insight into the nature of the exact solution as well as that of the eigensystem of L.

Appendix A. Proofs.

A.1. Proof of Lemma 2.1. From $X_\delta^T X_\delta = I$ we obtain

$$\begin{aligned}
\frac{dT_\delta^j}{d\delta} &= \sum_{k=0}^{j-1} T_\delta^k \frac{d(X_\delta^T A X_\delta)}{d\delta} T_\delta^{j-k-1} \\
&= \sum_{k=0}^{j-1} T_\delta^k [(X_\delta')^T A X_\delta + X_\delta^T A X_\delta'] T_\delta^{j-k-1} \\
&= \sum_{k=0}^{j-1} T_\delta^k [(X_\delta')^T (X_\delta T_\delta + \mathbf{r}_\delta \mathbf{e}_K^T) + (\mathbf{e}_K \mathbf{r}_\delta^T + T_\delta X_\delta^T) X_\delta'] T_\delta^{j-k-1} \\
&= (X_\delta')^T X_\delta T_\delta^j + T_\delta^j X_\delta^T X_\delta' +
\end{aligned}$$

ETNA
Kent State University
etna@mcs.kent.edu

$$\sum_{k=0}^{j-1} T_\delta^k (X_\delta')^T \mathbf{r}_\delta \mathbf{e}_K^T T_\delta^{j-k-1} + T_\delta^k \mathbf{e}_K \mathbf{r}_\delta^T X_\delta' T_\delta^{j-k-1}.$$

From symmetry, it follows that

$$\frac{1}{2}\frac{d}{d\delta}\left(\mathbf{e}_1^T T_\delta^j \mathbf{e}_1\right) = \mathbf{e}_1^T (X_\delta')^T X_\delta T_\delta^j \mathbf{e}_1 + \sum_{k=0}^{j-1} \mathbf{e}_1^T T_\delta^k (X_\delta')^T \mathbf{r}_\delta \mathbf{e}_K^T T_\delta^{j-k-1}\mathbf{e}_1.$$

From repeated application of the relation $AX_\delta = X_\delta T_\delta + \mathbf{r}_\delta \mathbf{e}_K^T$, we obtain

$$A^j X_\delta = X_\delta T_\delta^j + \sum_{k=0}^{j-1} A^k \mathbf{r}_\delta \mathbf{e}_K^T T_\delta^{j-k-1},$$

which yields

$$\begin{aligned}
\frac{1}{2}\frac{d}{d\delta}\left(\mathbf{e}_1^T T_\delta^j \mathbf{e}_1\right) &= \mathbf{e}_1^T (X_\delta')^T X_\delta T_\delta^j \mathbf{e}_1 + \sum_{k=0}^{j-1} \mathbf{e}_1^T T_\delta^k (X_\delta')^T \mathbf{r}_\delta \mathbf{e}_K^T T_\delta^{j-k-1}\mathbf{e}_1 \\
&= \mathbf{e}_1^T (X_\delta')^T A^j X_\delta \mathbf{e}_1 + \\
&\quad \sum_{k=0}^{j-1} \mathbf{e}_1^T \left[T_\delta^k (X_\delta')^T - (X_\delta')^T A^k\right] \mathbf{r}_\delta \mathbf{e}_K^T T_\delta^{j-k-1}\mathbf{e}_1 \\
&= \mathbf{e}_1^T (X_\delta')^T A^j X_\delta \mathbf{e}_1 + \\
&\quad \sum_{k=0}^{j-1} \mathbf{e}_1^T \left[(T_\delta^k)' X_\delta^T + T_\delta^k (X_\delta')^T - (X_\delta')^T A^k\right] \mathbf{r}_\delta \mathbf{e}_K^T T_\delta^{j-k-1}\mathbf{e}_1 \\
&= \mathbf{e}_1^T (X_\delta')^T A^j X_\delta \mathbf{e}_1 + \\
&\quad \sum_{k=0}^{j-1} \mathbf{e}_1^T \left[T_\delta^k X_\delta^T - X_\delta^T A^k\right]' \mathbf{r}_\delta \mathbf{e}_K^T T_\delta^{j-k-1}\mathbf{e}_1 \\
&= \mathbf{e}_1^T (X_\delta')^T A^j X_\delta \mathbf{e}_1 + \\
&\quad \sum_{k=K}^{j-K} \mathbf{e}_1^T \left[T_\delta^k X_\delta^T - X_\delta^T A^k\right]' \mathbf{r}_\delta \mathbf{e}_K^T T_\delta^{j-k-1}\mathbf{e}_1.
\end{aligned}$$

From the relations

$$X_\delta \mathbf{e}_1 = \frac{\mathbf{u}_\delta}{\|\mathbf{u}_\delta\|_2}, \quad X_\delta' \mathbf{e}_1 = \frac{1}{\|\mathbf{u}_\delta\|_2}\left(\mathbf{v} - \frac{\mathbf{u}^T\mathbf{v} + \delta \mathbf{v}^T\mathbf{v}}{\|\mathbf{u}_\delta\|_2^2}\mathbf{u}_\delta\right),$$

we obtain

$$\begin{aligned}
\tilde{g}_j'(\delta) &= \frac{1}{2}\left[\mathbf{e}_1^T \frac{dT_\delta^j}{d\delta}\mathbf{e}_1 \|\mathbf{u}_\delta\|_2^2 + 2\mathbf{e}_1^T T_\delta^j \mathbf{e}_1 (\mathbf{u}^T\mathbf{v} + \delta \mathbf{v}^T \mathbf{v})\right] \\
&= \mathbf{e}_1^T (X_\delta')^T A^j X_\delta \mathbf{e}_1 \mathbf{u}_\delta^T \mathbf{u}_\delta + \\
&\quad \sum_{k=K}^{j-K} \mathbf{e}_1^T \left[T_\delta^k X_\delta^T - X_\delta^T A^k\right]' \mathbf{r}_\delta \mathbf{e}_K^T T_\delta^{j-k-1}\mathbf{e}_1 \mathbf{u}_\delta^T \mathbf{u}_\delta + \\
&\quad \mathbf{e}_1^T T_\delta^j \mathbf{e}_1 (\mathbf{u}^T\mathbf{v} + \delta \mathbf{v}^T\mathbf{v})
\end{aligned}$$

$$
\begin{aligned}
&= \left(\mathbf{v} - \frac{\mathbf{u}^T\mathbf{v} + \delta\mathbf{v}^T\mathbf{v}}{\mathbf{u}_\delta^T\mathbf{u}_\delta}\mathbf{u}_\delta\right)^T A^j\mathbf{u}_\delta + \\
&\quad \sum_{k=K}^{j-K} \mathbf{e}_1^T \left[T_\delta^k X_\delta^T - X_\delta^T A^k\right]' \mathbf{r}_\delta \mathbf{e}_K^T T_\delta^{j-k-1}\mathbf{e}_1\mathbf{u}_\delta^T\mathbf{u}_\delta + \\
&\quad \mathbf{u}_\delta^T A^j\mathbf{u}_\delta \frac{\mathbf{u}^T\mathbf{v} + \delta\mathbf{v}^T\mathbf{v}}{\mathbf{u}_\delta^T\mathbf{u}_\delta} \\
&= \mathbf{u}_\delta^T A^j\mathbf{v} + \sum_{k=K}^{j-K} \mathbf{e}_1^T \left[T_\delta^k X_\delta^T - X_\delta^T A^k\right]' \mathbf{r}_\delta \mathbf{e}_K^T T_\delta^{j-k-1}\mathbf{e}_1\mathbf{u}_\delta^T\mathbf{u}_\delta.
\end{aligned}
$$

The lemma follows immediately from the Taylor expansion of $\tilde{g}_j(\delta)$.

A.2. Proof of Lemma 2.3. For convenience, we write

$$
L = \sum_{\alpha=0}^{m} a_\alpha(x)\left(\frac{\partial}{\partial x}\right)^\alpha,
$$

where $a_2(x) = -p(x)$, $a_1(x) = -p'(x)$, and $a_0(x) = q(x)$. For $j = 1$, we have

$$
\begin{aligned}
Lf(x) &= \sum_{\alpha=0}^{m} a_\alpha(x)\left(\frac{\partial}{\partial x}\right)^\alpha f(x) \\
&= \sum_{\alpha=0}^{m} \left(\frac{1}{\sqrt{2\pi}}\sum_{\omega=-N/2+1}^{N/2-1} \hat{a}_\alpha(\omega)e^{i\omega x}\right)\left(\frac{1}{\sqrt{2\pi}}\sum_{\xi=-N/2+1}^{N/2-1} \hat{f}(\xi)(i\xi)^\alpha e^{i\xi x}\right) \\
&= \sum_{\alpha=0}^{m} \left\{\frac{1}{\sqrt{2\pi}}\sum_{\omega=-N/2+1}^{N/2-1} \left[\frac{1}{\sqrt{2\pi}}\sum_{\xi=-N/2+1}^{N/2-1} \hat{a}_\alpha(\omega)\hat{f}(\xi)(i\xi)^\alpha e^{i(\omega+\xi)x}\right]\right\} \\
&= \sum_{\alpha=0}^{m} \left\{\frac{1}{\sqrt{2\pi}}\sum_{\omega=-N/2+1}^{N/2-1} \left[\frac{1}{\sqrt{2\pi}}\sum_{\eta=-N+1}^{N-1} \hat{a}_\alpha(\omega)\hat{f}(\eta-\omega)(i\xi)^\alpha e^{i\eta x}\right]\right\} \\
&= \sum_{\alpha=0}^{m} \left\{\frac{1}{\sqrt{2\pi}}\sum_{\eta=-N+1}^{N-1} \frac{1}{\sqrt{2\pi}}\left[\sum_{\omega=-N/2+1}^{N/2-1} \hat{a}_\alpha(\omega)\hat{f}(\eta-\omega)(i\xi)^\alpha\right] e^{i\eta x}\right\},
\end{aligned}
$$

thus $Lf \in V_{2N}$. Because Fourier interpolation of Lf, for any degree $\geq 2N$, is exact, (2.9) follows.

A.3. Proof of Theorem 2.5. Let the vector $\hat{\mathbf{e}}_\omega$ be a discretization of $\phi_\omega(x) = \frac{1}{\sqrt{2\pi}}e^{i\omega x}$ on a uniform grid of the form (2.1); that is,

$$
[\hat{\mathbf{e}}_\omega]_j = \frac{1}{\sqrt{2\pi}}e^{i\omega jh}, \quad j = 0, 1, \ldots, N-1.
$$

The approximation $\tilde{I}_\omega(\Delta t)$ of $I_\omega(\Delta t)$ computed by Algorithm 2.1 has the form

$$
\tilde{I}_\omega(\Delta t) = \mathbf{e}_1^H e^{-T\Delta t}\mathbf{e}_1,
$$

where T is the $K \times K$ Jacobi matrix produced by the symmetric Lanczos algorithm applied to the matrix L_{M_K} defined in (2.2) with initial vector $\hat{\mathbf{e}}_\omega$. Thus we have

$$
L_{M_K}X = XT + \mathbf{r}\mathbf{e}_K^H, \quad X^H X = I_K, \quad X\mathbf{e}_1 = \hat{\mathbf{e}}_\omega.
$$

We can express the error $E_\omega(\Delta t) = \langle \phi_\omega, \exp[-L\Delta t]\phi_\omega\rangle - \tilde{I}_\omega(\Delta t)$ as

$$\begin{aligned}
E_\omega(\Delta t) &= \langle \phi_\omega, \exp[-L\Delta t]\phi_\omega\rangle - \tilde{I}_\omega(\Delta t) \\
&= \sum_{j=0}^{\infty} \frac{\Delta t^j}{j!}(\langle \phi_\omega, L^j\phi_\omega\rangle - \mathbf{e}_1^H T^j \mathbf{e}_1) \\
&= \sum_{j=0}^{\infty} \frac{\Delta t^j}{j!}(\langle \phi_\omega, L^j\phi_\omega\rangle - \hat{\mathbf{e}}_\omega^H L_{M_K}^j \hat{\mathbf{e}}_\omega + \\
&\qquad \hat{\mathbf{e}}_\omega^H L_{M_K}^j \hat{\mathbf{e}}_\omega - \mathbf{e}_1^H X^H X T^j \mathbf{e}_1) \\
&= \sum_{j=0}^{\infty} \frac{\Delta t^j}{j!}(\langle \phi_\omega, L^j\phi_\omega\rangle - \hat{\mathbf{e}}_\omega^H L_{M_K}^j \hat{\mathbf{e}}_\omega + \\
&\qquad \hat{\mathbf{e}}_\omega^H (L_{M_K}^j X - X T^j)\mathbf{e}_1) \\
&= \sum_{j=0}^{\infty} \frac{\Delta t^j}{j!}\Big[\langle \phi_\omega, L^j\phi_\omega\rangle - \hat{\mathbf{e}}_\omega^H L_{M_K}^j \hat{\mathbf{e}}_\omega + \\
&\qquad \hat{\mathbf{e}}_\omega^H \left(\sum_{k=0}^{j-1} L_{M_K}^k \mathbf{r}\mathbf{e}_K^H T^{j-k-1} \right) \mathbf{e}_1 \Big]. \qquad \text{(A.1)}
\end{aligned}$$

We first consider the expression $\mathbf{e}_K^H T^j \mathbf{e}_1$, where j is a positive integer and $K > 1$. Then

$$\beta_1 = T_{21} = \|V_{M_K}\hat{\mathbf{e}}_\omega\|_2.$$

It follows from the fact that T is tridiagonal, that

$$|[T^j]_{K1}| \leq \|V_{M_K}\|\|3L_{M_K}\|^{j-1}$$

and therefore, for $0 \leq k < j-1$,

$$|\hat{\mathbf{e}}_\omega^H L_{M_K}^k \mathbf{r}[T^{j-k-1}]_{K1}| \leq 3^{j-k-2}\|V_{M_K}\|\|L_{M_K}\|^{j+K-2}.$$

When $j = k-1$ and $K > 1$, the expression on the left side vanishes. If $K = 1$, then $T = \alpha_1$ and $\mathbf{r} = V_{M_1}\hat{\mathbf{e}}_\omega$, which yields a similar bound for $0 \leq k \leq j-1$.

Next, we consider the expression $E_{\omega,j} = \langle \phi_\omega, L^j\phi_\omega\rangle - \hat{\mathbf{e}}_\omega^H L_{M_K}^j \hat{\mathbf{e}}_\omega$, where j is a non-negative integer. By Lemma 2.1, $E_{\omega,j} = 0$ for $j \leq 2K$. For $j > 2K$, we define $f_{\omega,\ell}(x) = L^\ell \phi_\omega(x)$ for any nonnegative integer ℓ. Furthermore, for even positive integers M we define the following operators on the space of continuous functions defined on $[0, 2\pi]$:

- P_M is the orthogonal projection onto V_M:

$$P_M f(x) = \frac{1}{\sqrt{2\pi}} \sum_{\omega=-M/2+1}^{M/2-1} e^{i\omega x}\hat{f}(\omega).$$

- Π_M is the composition of P_M and the M-point interpolation operator, using an M-point uniform grid of the form (2.1):

$$\Pi_M f(x) = \frac{1}{\sqrt{2\pi}} \sum_{\omega=-M/2+1}^{M/2-1} e^{i\omega x} \left(\frac{h}{\sqrt{2\pi}} \sum_{j=0}^{M-1} e^{-i\omega 2\pi jh} f(jh) \right),$$

with $h = \frac{2\pi}{M}$. Certainly, if $f \in V_M$, then $\Pi_M f = f$.

ETNA
Kent State University
etna@mcs.kent.edu

Using these definitions, we obtain

$$\begin{aligned}E_{\omega,j} &= \langle \phi_\omega, L^j \phi_\omega\rangle - \hat{\mathbf{e}}_\omega^H L_{M_K}^j \hat{\mathbf{e}}_\omega \\ &= \langle f_{\omega,K}, [L^{j-2K} - (\Pi_{M_K} L \Pi_{M_K})^{j-2K}] f_{\omega,K}\rangle \\ &= \langle f_{\omega,K}, [(C+V)L^{j-2K-1} - \Pi_{M_K}(C+V)\Pi_{M_K}(\Pi_{M_K} L\Pi_{M_K})^{j-2K-1}] f_{\omega,K}\rangle.\end{aligned}$$

Let $j = 2K+1$. By Lemma 2.1, $f_{\omega,K} \in V_{M_K}$, from which it follows that $C^* f_{\omega,K} \in V_{M_K}$, and therefore

$$E_{\omega,2K+1} = \langle f_{\omega,K}, [V - \Pi_{M_K} V \Pi_{M_K}] f_{\omega,K}\rangle$$

from which it follows that

$$|E_{\omega,2K+1}| \leq 2\|V_{M_K}\|\|L_{M_K}\|^{2K}.$$

In general, we have

$$\begin{aligned}E_{\omega,j} &= \langle f_{\omega,K}, [L^{j-2K} - (\Pi_{M_K} L\Pi_{M_K})^{j-2K}] f_{\omega,K}\rangle \\ &= \langle f_{\omega,K}, [C^{j-2K} - (\Pi_{M_K} C\Pi_{M_K})^{j-2K}] f_{\omega,K}\rangle + \\ &\quad \langle f_{\omega,K}, [E_{j-2K} - E_{M_K,j-2K}] f_{\omega,K}\rangle \\ &= \langle f_{\omega,K}, [E_{j-2K} - E_{M_K,j-2K}] f_{\omega,K}\rangle\end{aligned}$$

where

$$E_{j-2K} = L^{j-2K} - C^{j-2K},$$

and

$$E_{M_K,j-2K} = (\Pi_{M_K} L \Pi_{M_K})^{j-2K} - (\Pi_{M_K} C\Pi_{M_K})^{j-2K}.$$

It follows that, for fixed Δt, $E_\omega(\Delta t) \to 0$ linearly with $\|V_{M_K}\|$. By Lemma 2.1, the terms in (A.1) that are of order $< 2K$ in Δt vanish, which completes the proof.

REFERENCES

[1] J. P. BOYD, *Chebyshev and Fourier Spectral Methods*, 2nd edition, Dover Publications, Inc., Mineola, NY, 2001.
[2] D. CALVETTI, G. H. GOLUB, W. B. GRAGG, AND L. REICHEL, *Computation of Gauss-Kronrod quadrature rules*, Math. Comp., 69 (2000), pp. 1035-1052.
[3] G. H. GOLUB AND C. MEURANT, *Matrices, Moments and Quadrature*, in *Proceedings of the 15th Dundee Conference*, June-July 1993, D. F. Griffiths and G. A. Watson (eds.), Longman Scientific & Technical, 1994.
[4] G. H. GOLUB AND C. F. VAN LOAN, *Matrix Computations*, 3rd edition, Johns Hopkins University Press, 1996.
[5] J. GOODMAN, T. HOU, AND E. TADMOR, *On the stability of the unsmoothed Fourier method for hyperbolic equations*, Numer. Math., 67 (1994), pp. 93-129.
[6] P. GUIDOTTI, J. V. LAMBERS, AND K. SØLNA, *Analysis of Wave Propagation in 1D Inhomogeneous Media*, to appear in Numer. Funct. Anal. Optim., 2005.
[7] B. GUSTAFSSON, H.-O. KREISS, AND J. OLIGER, *Time-Dependent Problems and Difference Methods*, Wiley, New York, 1995.
[8] M. HOCHBRUCK AND C. LUBICH, *On Krylov Subspace Approximations to the Matrix Exponential Operator*, SIAM J. Numer. Anal., 34 (1996), pp. 1911-1925.
[9] C. JOHNSON, *Numerical solutions of partial differential equations by the finite element method*, Cambridge University Press, 1987.

[10] J. V. LAMBERS, *Krylov Subspace Methods for Variable-Coefficient Initial-Boundary Value Problems*, Ph.D. Thesis, Stanford University, SCCM Program, 2003, available at `http://sccm.stanford.edu/pub/sccm/theses/James_Lambers.pdf`.
[11] ———, *Approximating Eigenfunctions of Variable-Coefficient Differential Operators*, in preparation.
[12] ———, *Practical Implementation of Krylov Subspace Spectral Methods*, submitted.
[13] ———, *A Stability Analysis of Krylov Subspace Spectral Methods*, in preparation.
[14] P. E. SAYLOR AND D. C. SMOLARSKI, *Why Gaussian quadrature in the complex plane?*, Numer. Algorithms, 26 (2001), pp. 251-280.
[15] J. STOER AND R. BURLISCH, *Introduction to Numerical Analysis*, 2nd edition, Springer-Verlag, 1983.