

# Practical Implementation of Krylov Subspace Spectral Methods

James V. Lambers

Published online: 6 July 2007  
© Springer Science+Business Media, LLC 2007

**Abstract** Krylov subspace spectral methods have been shown to be high-order accurate in time and more stable than explicit time-stepping methods, but also more difficult to implement efficiently. This paper describes how these methods can be fashioned into practical solvers by exploiting the simple structure of differential operators. Numerical results concerning accuracy and efficiency are presented for parabolic problems in one and two space dimensions.

**Keywords** Lanczos method · Spectral methods · Gaussian quadrature

## 1 Introduction

Consider the initial-boundary value problem in one space dimension,

$$\frac{\partial u}{\partial t}(x, t) + L(x, D)u(x, t) = 0, \quad 0 < x < 2\pi, \quad t > 0, \quad (1.1)$$

$$u(x, 0) = f(x), \quad 0 < x < 2\pi, \quad (1.2)$$

with periodic boundary conditions

$$u(0, t) = u(2\pi, t), \quad t > 0, \quad (1.3)$$

where  $L(x, D)$  is an  $m$ -th order differential operator of the form

$$L(x, D)u(x) = \sum_{\mu=0}^m a_{\mu}(x)D^{\mu}u, \quad D = \frac{d}{dx}, \quad (1.4)$$

---

J.V. Lambers (✉)

Department of Energy Resources Engineering, Stanford University, Stanford, CA 94305-2220, USA  
e-mail: lambers@stanford.edu

with coefficients  $a_\mu, \mu = 0, 1, \dots, m$ . We assume that  $L(x, D)$  is self-adjoint and positive semi-definite.

In [15, 16], a class of methods for numerically solving problems of this form, called Krylov subspace spectral methods, was introduced. These methods achieve high-order accuracy in time, and greater stability than explicit time-stepping methods. However, in exchange for these desirable properties, the approach employed by these methods can be computationally expensive compared to existing methods.

In this paper, we discuss how these methods can be implemented to obtain a much more efficient algorithm. We will exploit the structure of differential operators in order to reduce the work required per time step by an order of magnitude in the number of points in the spatial grid. We show how this is done in one and two space dimensions.

### 2 Krylov Subspace Spectral Methods

Let  $S(x, D; t) = \exp[-L(x, D)t]$  represent the exact solution operator of the problem (1.1), (1.2), (1.3), and let  $\langle \cdot, \cdot \rangle$  denote the standard inner product of functions defined on  $[0, 2\pi]$ ,

$$\langle f(x), g(x) \rangle = \int_0^{2\pi} \overline{f(x)}g(x) dx. \tag{2.1}$$

Krylov subspace spectral methods, introduced in [15, 16], use Gaussian quadrature on the spectral domain to compute the Fourier components of the solution. These methods are time-stepping algorithms that compute the solution at time  $t_1, t_2, \dots$ , where  $t_n = n\Delta t$  for some choice of  $\Delta t$ . Given the computed solution  $\tilde{u}(x, t_n)$  at time  $t_n$ , the solution at time  $t_{n+1}$  is computed by approximating the Fourier components that would be obtained by applying the exact solution operator to  $\tilde{u}(x, t_n)$ ,

$$\hat{u}(\omega, t_{n+1}) = \left\langle \frac{1}{\sqrt{2\pi}} e^{i\omega x}, S(x, D; t)\tilde{u}(x, t_n) \right\rangle. \tag{2.2}$$

Krylov subspace spectral methods approximate these components with higher-order temporal accuracy than traditional spectral methods and time-stepping schemes. We briefly review how these methods work.

We discretize functions defined on  $[0, 2\pi]$  on an  $N$ -point uniform grid with spacing  $h = 2\pi/N$ . With this discretization, the operator  $L(x, D)$  and the solution operator  $S(x, D; \Delta t)$  can be approximated by  $N \times N$  matrices that represent linear operators on the space of grid functions, and the quantity (2.2) can be approximated by a bilinear form

$$\hat{u}(\omega, t_{n+1}) \approx \hat{\mathbf{e}}_\omega^H S_N(\Delta t)\mathbf{u}(t_n), \tag{2.3}$$

where

$$[\hat{\mathbf{e}}_\omega]_j = \frac{1}{\sqrt{2\pi}} e^{i\omega jh}, \quad [\mathbf{u}(t_n)]_j = u(jh, t_n), \tag{2.4}$$

and

$$S_N(t) = \exp[-L_N t], \quad [L_N]_{jk} = \sum_{\mu=0}^m a_\mu(jh)[D_N^\mu]_{jk}, \tag{2.5}$$

where  $D_N$  is a spectral discretization of the differentiation operator that is defined on the space of grid functions, which are the eigenfunctions of  $D_N$ . Our goal is to approximate (2.3) by computing an approximation to

$$[\hat{\mathbf{u}}^{n+1}]_\omega = \hat{\mathbf{e}}_\omega^H \mathbf{u}(t_{n+1}) = \hat{\mathbf{e}}_\omega^H S_N(\Delta t) \mathbf{u}(t_n). \tag{2.6}$$

In [11] Golub and Meurant describe a method for computing quantities of the form

$$\mathbf{u}^T f(A) \mathbf{v}, \tag{2.7}$$

where  $\mathbf{u}$  and  $\mathbf{v}$  are  $N$ -vectors,  $A$  is an  $N \times N$  symmetric positive semi-definite matrix, and  $f$  is a smooth function. Our goal is to apply this method with  $A = L_N$  where  $L_N$  was defined in (2.5),  $f(\lambda) = \exp(-\lambda t)$  for some  $t$ , and the vectors  $\mathbf{u}$  and  $\mathbf{v}$  are derived from  $\hat{\mathbf{e}}_\omega$  and  $\mathbf{u}(t_n)$ .

The basic idea is as follows: since the matrix  $A$  is symmetric positive semi-definite, it has real eigenvalues

$$b = \lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_N = a \geq 0, \tag{2.8}$$

and corresponding orthogonal eigenvectors  $\mathbf{q}_j$ ,  $j = 1, \dots, N$ . Therefore, the quantity (2.7) can be rewritten as

$$\mathbf{u}^T f(A) \mathbf{v} = \sum_{\ell=1}^N f(\lambda_\ell) \mathbf{u}^T \mathbf{q}_\ell \mathbf{q}_\ell^T \mathbf{v}. \tag{2.9}$$

We let  $a = \lambda_N$  be the smallest eigenvalue,  $b = \lambda_1$  be the largest eigenvalue, and define the measure  $\alpha(\lambda)$  by

$$\alpha(\lambda) = \begin{cases} 0, & \text{if } \lambda < a, \\ \sum_{j=i}^N \alpha_j \beta_j, & \text{if } \lambda_i \leq \lambda < \lambda_{i-1}, \\ \sum_{j=1}^N \alpha_j \beta_j, & \text{if } b \leq \lambda, \end{cases} \quad \alpha_j = \mathbf{u}^T \mathbf{q}_j, \beta_j = \mathbf{q}_j^T \mathbf{v}. \tag{2.10}$$

If this measure is positive and increasing, then the quantity (2.7) can be viewed as a Riemann-Stieltjes integral

$$\mathbf{u}^T f(A) \mathbf{v} = I[f] = \int_a^b f(\lambda) d\alpha(\lambda). \tag{2.11}$$

As discussed in [5, 8, 9, 11], the integral  $I[f]$  can be bounded using either Gauss, Gauss-Radau, or Gauss-Lobatto quadrature rules, all of which yield an approximation of the form

$$I[f] = \sum_{j=1}^K w_j f(t_j) + \sum_{j=1}^M v_j f(z_j) + R[f], \tag{2.12}$$

where the nodes  $t_j$ ,  $j = 1, \dots, K$ , and  $z_j$ ,  $j = 1, \dots, M$ , as well as the weights  $w_j$ ,  $j = 1, \dots, K$ , and  $v_j$ ,  $j = 1, \dots, M$ , can be obtained using the symmetric Lanczos algorithm if  $\mathbf{u} = \mathbf{v}$ , and the unsymmetric Lanczos algorithm if  $\mathbf{u} \neq \mathbf{v}$ .

In the case  $\mathbf{u} \neq \mathbf{v}$ , there is the possibility that the weights may not be positive, which destabilizes the quadrature rule (see [2] for details). Therefore, it is best to handle this case by rewriting (2.7) using decompositions such as

$$\mathbf{u}^T f(A) \mathbf{v} = \frac{1}{\delta} [\mathbf{u}^T f(A) (\mathbf{u} + \delta \mathbf{v}) - \mathbf{u}^T f(A) \mathbf{u}], \tag{2.13}$$

or the polar decomposition

$$\mathbf{u}^T f(A)\mathbf{v} = \frac{1}{4\delta} [(\mathbf{u} + \delta\mathbf{v}) f(A)(\mathbf{u} + \delta\mathbf{v}) - (\mathbf{u} - \delta\mathbf{v})^T f(A)(\mathbf{u} - \delta\mathbf{v})], \tag{2.14}$$

where  $\delta$  is a small constant. Guidelines for choosing an appropriate value for  $\delta$  can be found in [16, Sect. 2.2].

Employing these quadrature rules yields the following basic process (for details see [15, 16]) for computing the Fourier coefficients of  $\mathbf{u}(t_{n+1})$  from  $\mathbf{u}(t_n)$ . It is assumed that when the Lanczos algorithm (symmetric or unsymmetric) is employed,  $M + K$  iterations are performed to obtain the  $M + K$  quadrature nodes and weights.

```

for  $\omega = -N/2 + 1, \dots, N/2 - 1$ 
    Choose a scaling constant  $\delta_\omega$ 
    Compute  $u_1 \approx \hat{\mathbf{e}}_\omega^H S_N(\Delta t) \hat{\mathbf{e}}_\omega$ 
        using the symmetric Lanczos algorithm
    Compute  $u_2 \approx \hat{\mathbf{e}}_\omega^H S_N(\Delta t) (\hat{\mathbf{e}}_\omega + \delta_\omega \mathbf{u}^n)$ 
        using the unsymmetric Lanczos algorithm
     $[\hat{\mathbf{u}}^{n+1}]_\omega = (u_2 - u_1) / \delta_\omega$ 
end
    
```

It should be noted that the constant  $\delta_\omega$  plays the role of  $\delta$  in the decomposition (2.13), and the subscript  $\omega$  is used to indicate that a different value may be used for each wave number  $\omega = -N/2 + 1, \dots, N/2 - 1$ . Also, in the presentation of this algorithm in [16], the polar decomposition (2.14) is used instead of (2.13), and is applied to sines and cosines instead of complex exponential functions.

This algorithm has high-order temporal accuracy, as indicated by the following theorem. Let  $V_N = \text{span}\{e^{-i\omega x}\}_{\omega=-N/2+1}^{N/2-1}$ .

**Theorem 2.1** *Let  $L(x, D)$  be a self-adjoint  $m$ -th order positive definite differential operator on  $C_p([0, 2\pi])$  with coefficients in  $V_N$ . Let  $f \in V_N$ , and let  $M = 0$ . Then the preceding algorithm, applied to the problem (1.1), (1.2), (1.3), is consistent; i.e.*

$$[\hat{\mathbf{u}}^1]_\omega - \hat{u}(\omega, \Delta t) = O(\Delta t^{2K}),$$

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

*Proof* See [16, Lemma 2.1, Theorem 2.4]. □

Using results in [11] regarding the error term  $R[f]$  in (2.12), it can be shown that if  $M$  prescribed nodes are used in addition to the  $K$  free nodes, then the local truncation error is  $O(\Delta t^{2K+M})$ . As shown in [16], significantly greater accuracy can be achieved for some problems by using a Gauss-Radau rule with one prescribed node that approximates the smallest eigenvalue of  $L(x, D)$ . Also, it should be noted that in [12], a variation of Krylov subspace spectral methods is applied to variable-coefficient second-order wave equations, achieving  $O(\Delta t^{4K+2M})$  accuracy. Furthermore, for such problems, Dirichlet boundary conditions were used instead of the periodic conditions that will be imposed throughout this paper, thus demonstrating the suitability of the method for “true” IBVP.

The preceding result can be compared to the accuracy achieved by an algorithm described by Hochbruck and Lubich in [13] for computing  $e^{A\Delta t}\mathbf{v}$  for a given matrix  $A$  and vector  $\mathbf{v}$  using the unsymmetric Lanczos algorithm. As discussed in [13], this algorithm can be used

to compute the solution of some ODEs without time-stepping, but this becomes less practical for ODEs arising from a semi-discretization of problems such as (1.1), (1.2), (1.3), due to their stiffness. In this situation, it is necessary to either use a high-dimensional Krylov subspace, in which case reorthogonalization is required, or one can resort to time-stepping, in which case the local temporal error is only  $O(\Delta t^K)$ , assuming a  $K$ -dimensional Krylov subspace. Regardless of which remedy is used, the computational effort needed to compute the solution at a fixed time  $T$  increases substantially.

The difference between Krylov subspace spectral methods and the approach described in [13] is that in the former, a different  $K$ -dimensional Krylov subspace is used for each Fourier component, instead of the same subspace for all components as in the latter. As can be seen from numerical results comparing the two approaches in [16], using the same subspace for all components causes a loss of accuracy as the number of grid points increases, whereas Krylov subspace spectral methods do not suffer from this phenomenon. Furthermore, while traditional Krylov subspace methods for time-dependent differential equations use the natural choice of the solution from the previous time step as the initial vector for a Krylov subspace, Krylov subspace spectral methods use *perturbations* of component-dependent Krylov subspaces, in the direction of the previous solution. It is this adjustment which allows a small number of quadrature nodes to be used and still obtain high-order accuracy.

Unfortunately, in spite of this high-order accuracy, a straightforward implementation of this algorithm is much too slow to be used as a time-stepping scheme, because at least  $O(N^2 \log N)$  operations are required to carry out the Lanczos iteration  $2N$  times, if differentiation is implemented using the FFT. Fortunately, we can take advantage of the fact that  $L_N$  represents a differential operator in order to optimize this process, as described in Sect. 3. In addition, the fact that the initial vectors used for these iterations can easily be described as a one-parameter family can also be exploited, yielding further savings. This is discussed in Sect. 4. Sections 5 and 6 show how Krylov subspace spectral methods can be efficiently generalized to problems in higher space dimensions or problems containing source terms. Section 7 features comparisons with other numerical methods, in terms of both accuracy and efficiency. Section 8 consists of concluding remarks and indications of future directions.

### 3 Symbolic Computation of Jacobi Matrices

Let  $J_\omega$  be the Jacobi matrix created by the symmetric Lanczos iteration with starting vector  $\hat{e}_\omega$ . The basic idea is to use symbolic calculus to create a representation of the nonzero elements of  $J_\omega$  as a function of  $\omega$ . Consider the symmetric Lanczos iteration applied to a general matrix  $A$  with starting vector  $\mathbf{r}_0$ :

```

Choose  $\mathbf{r}_0$ 
 $\beta_0 = 1$ 
 $\mathbf{x}_0 = 0$ 
for  $j = 1, \dots, k$ 
     $\mathbf{x}_j = \mathbf{r}_{j-1} / \beta_{j-1}$ 
     $\alpha_j = \mathbf{x}_j^H A \mathbf{x}_j$ 
     $\mathbf{r}_j = (A - \alpha_j I) \mathbf{x}_j - \beta_{j-1} \mathbf{x}_{j-1}$ 
     $\beta_j^2 = \mathbf{r}_j^H \mathbf{r}_j$ 
end
    
```

It would be desirable to re-use as much computational effort as possible in applying this algorithm for each frequency  $\omega$ . To that end, we will now carry out this iteration for a given operator  $L(x, D)$  and variable frequency  $\omega$  and compute elements of  $J_\omega$ , represented as functions of  $\omega$ , in order to determine how much re-use is possible.

### 3.1 A Simple Example

Consider a second-order self-adjoint differential operator of the form

$$L(x, D) = a_2 D^2 + a_0(x), \tag{3.1}$$

where  $a_2 > 0$  and  $a_0(x) \leq 0$ , with symbol

$$L(x, \xi) = -a_2 \xi^2 + a_0(x). \tag{3.2}$$

We assume that  $a_0(x)$  can be written as a truncated Fourier series

$$a_0(x) = \sum_{\omega=-N/2+1}^{N/2-1} e^{i\omega x} \hat{a}_0(\omega). \tag{3.3}$$

We will now apply the symmetric Lanczos iteration to a discretization of this operator with initial vector  $\hat{\mathbf{e}}_\omega$ , which is a discretization of the function

$$\hat{e}_\omega(x) = \frac{1}{\sqrt{2\pi}} e^{i\omega x} \tag{3.4}$$

on a uniform grid  $x_j = jh, j = 0, 1, \dots, N - 1$ , with  $h = 2\pi/N$ . Specifically,

$$[\hat{\mathbf{e}}_\omega]_j = \frac{1}{\sqrt{2\pi}} e^{i\omega x_j}.$$

For given  $2\pi$ -periodic functions  $f$  and  $g$ , we will approximate the inner product  $\langle f, g \rangle$  defined in (2.1) by the discrete inner product  $\langle f, g \rangle_h$  of the corresponding grid functions:

$$\langle f, g \rangle_h = h \sum_{j=0}^{N-1} \overline{f(x_j)} g(x_j). \tag{3.5}$$

In applying the Lanczos algorithm, we will examine how  $\alpha_1, \beta_1$  and  $\alpha_2$  can be computed as efficiently as possible. We have

$$\begin{aligned} \alpha_1 &= \langle \hat{e}_\omega, L(x, D)\hat{e}_\omega \rangle_h \\ &= -a_2 \omega^2 + \text{Avg } a_0, \end{aligned}$$

where

$$\text{Avg } f = \frac{1}{2\pi} \int_0^{2\pi} f(x) dx. \tag{3.6}$$

Proceeding to  $\beta_1$ , we have

$$\begin{aligned} \beta_1^2 &= \|(L(x, D) - \alpha_1 I)\hat{e}_\omega\|^2 \\ &= \|a_0(x)\|^2 - |\text{Avg } a_0|^2. \end{aligned}$$

Finally, for  $\alpha_2$  we let  $b(x) = (a_0(x) - \text{Avg } a_0)/\beta_1$  and obtain

$$\begin{aligned} \alpha_2 &= \langle b\hat{e}_\omega, L(x, D)b\hat{e}_\omega \rangle_h \\ &= \left\langle b\hat{e}_\omega, a_2 \frac{d^2(b\hat{e}_\omega)}{dx^2} \right\rangle_h + \langle b\hat{e}_\omega, a_0 b\hat{e}_\omega \rangle_h \\ &= a_2 \left[ \left\langle b\hat{e}_\omega, \frac{d^2(b\hat{e}_\omega)}{dx^2} \right\rangle_h \right] + \langle b, a_0 b \rangle_h \\ &= a_2 \left[ \left\langle b\hat{e}_\omega, \frac{d^2 b}{dx^2} \hat{e}_\omega \right\rangle_h + 2 \left\langle b\hat{e}_\omega, (i\omega) \frac{db}{dx} \hat{e}_\omega \right\rangle_h + \langle b\hat{e}_\omega, (i\omega)^2 b\hat{e}_\omega \rangle_h \right] + \langle b, a_0 b \rangle_h \\ &= a_2 \left[ \left\langle b, \frac{d^2 b}{dx^2} \right\rangle_h + 2 \left\langle b, (i\omega) \frac{db}{dx} \right\rangle_h + \langle b, (i\omega)^2 b \rangle_h \right] + \langle b, a_0 b \rangle_h \\ &= a_2 \left[ - \left\langle \frac{db}{dx}, \frac{db}{dx} \right\rangle_h + 2i\omega \left\langle b, \frac{db}{dx} \right\rangle_h - \omega^2 \right] + \langle b, a_0 b \rangle_h \\ &= -a_2 \omega^2 - a_2 \|b'\|_h^2 + \langle b, a_0 b \rangle_h. \end{aligned}$$

These coefficients can be computed in  $13N + \frac{3}{2}N \log N - 1$  floating-point operations.

We see that, so far, the entries of the tridiagonal matrix constructed by the Lanczos iteration are polynomials in  $\omega$ . While this does not hold in general, the entries can still be represented as functions of  $\omega$ . Therefore, we can construct the  $K$ -point Gaussian quadrature rules for all frequencies  $\omega = -N/2 + 1, \dots, N/2 - 1$  by first constructing representations for the elements  $\alpha_j, j = 1, \dots, K$ , and  $\beta_j, j = 1, \dots, K - 1$ , as functions of  $\omega$ , and then evaluating these representations for each  $\omega$ . By computing a representation of  $\beta_K$  as well, we can obtain  $K$ -point Gauss-Radau rules.

### 3.2 Variable Leading Coefficients

Now, suppose the operator  $L(x, D)$  has the form

$$L(x, D) = Da_2(x)D. \tag{3.7}$$

We assume that  $a_2(x)$  is twice continuously differentiable. In this case, it is also straightforward to compute the recursion coefficients  $\alpha_1, \beta_1$  and  $\alpha_2$  for a 2-node Gaussian rule. For convenience, we define, for a continuous function  $f(x)$ ,

$$\bar{f} = \text{Avg } f = \frac{1}{2\pi} \int_0^{2\pi} f(x) dx. \tag{3.8}$$

The formulas for the recursion coefficients are

$$\alpha_1 = -\bar{a}_2 \omega^2, \tag{3.9}$$

$$\beta_1^2 = \bar{a}_2^2 \omega^4 + \overline{[a_2']^2} \omega^2 - \alpha_1^2, \tag{3.10}$$

$$\alpha_2 = -\beta_1^{-2} [\omega^2 (\bar{a}_2^3 \omega^4 - 2\bar{a}_2^2 a_2'' \omega^2 + 4\overline{a_2 (a_2')^2} \omega^2 + \overline{a_2 (a_2'')^2}) + 2\alpha_1 \beta_1^2 + \alpha_1^3]. \tag{3.11}$$

The recursion coefficients for more general operators can be obtained by computing the symbols of powers of  $L(x, D)$  and averaging their coefficients. Details of the algorithm can be found in [15].

### 3.3 Efficient Computation of Quadrature Rules

Once the recursion coefficients are obtained, it is necessary to compute the nodes and weights of the Gaussian quadrature rule that will be used to approximate (2.3). As discussed in [11], the nodes are the eigenvalues of the matrix  $J_\omega$  whose elements, in the 2-node case, are computed using formulas such as those in the previous two subsections. The weights are the squares of the first components of the normalized eigenvectors.

In the case of a 2-node Gaussian rule, the nodes  $t_1, t_2$  and weights  $w_1, w_2$  are computed as follows:

$$\begin{aligned} r &= \alpha_1 + \alpha_2, \\ d &= \sqrt{(\alpha_1 - \alpha_2)^2 + 4\beta_1^2}, \\ t_1 &= \frac{r + d}{2}, \\ t_2 &= \frac{r - d}{2}, \\ w_1 &= \frac{\alpha_1 - t_2}{d}, \\ w_2 &= 1 - w_1. \end{aligned}$$

These computations can easily be vectorized, leading to much greater efficiency in, for example, a MATLAB implementation. For an operator of the form (3.7), it should be noted that for  $\omega = 0, \beta_1 = 0$ , so there is only one recursion coefficient,  $\alpha_1$ , but by setting  $\alpha_2 = \alpha_1$ , the above formulas can still be used, thus facilitating vectorization.

### 3.4 Choice of Spatial Discretization

In the computation of the recursion coefficients in this section, we used a spectral discretization of the differentiation operator,

$$D_N = T_N^{-1} \Lambda_N T_N, \quad (3.12)$$

where  $T_N$  is the matrix of the FFT on an  $N$ -point grid, and

$$\Lambda_N = \text{diag}(-N/2 + 1, -N/2 + 2, \dots, -1, 0, 1, \dots, N/2 - 1, N/2). \quad (3.13)$$

Certainly, this is a very accurate and efficient choice for the given starting vectors  $\hat{\mathbf{e}}_\omega$ . However, it is worth noting that any other spatial discretization of  $L(x, D)$  can be used to compute the recursion coefficients, without having to change the set of trial functions. In other words, the form of the solution, the temporal discretization of each component, and the spatial discretization used to compute the recursion coefficients are all independent of one another. This independence can be exploited for cases in which spectral differentiation is not as effective, such as with discontinuous functions.

## 4 Perturbations of Initial Vectors

The operations described in the previous section can be used to carry out the symmetric Lanczos iteration with starting vector  $\hat{\mathbf{e}}_\omega$  simultaneously for all  $\omega, \omega = -N/2 +$

$1, \dots, N/2 - 1$ . While they can also be used for the unsymmetric Lanczos iteration with starting vectors  $\hat{\mathbf{e}}_\omega$  and  $\hat{\mathbf{e}}_\omega + \delta_\omega \mathbf{f}$ , where  $\mathbf{f}$  is a given vector and  $\delta_\omega$  a constant depending on  $\omega$ , there is a more efficient alternative, which we describe in this section.

Let  $A$  be a symmetric positive definite  $n \times n$  matrix and let  $\mathbf{r}_0$  be an  $n$ -vector. Suppose that we have already carried out the symmetric Lanczos iteration given at the beginning of Sect. 3 to obtain orthogonal vectors  $\mathbf{r}_0, \dots, \mathbf{r}_k$  and the Jacobi matrix

$$J_k = \begin{bmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{k-2} & \alpha_{k-1} & \beta_{k-1} \\ & & & \beta_{k-1} & \alpha_k \end{bmatrix}, \tag{4.1}$$

along with the value  $\beta_k$ .

Now, we wish to compute the entries of the modified Jacobi matrix

$$\hat{J}_k = \begin{bmatrix} \hat{\alpha}_1 & \hat{\beta}_1 & & & \\ \hat{\beta}_1 & \hat{\alpha}_2 & \hat{\beta}_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \hat{\beta}_{k-2} & \hat{\alpha}_{k-1} & \hat{\beta}_{k-1} \\ & & & \hat{\beta}_{k-1} & \hat{\alpha}_k \end{bmatrix}, \tag{4.2}$$

along with the value  $\hat{\beta}_k$ , that result from applying the unsymmetric Lanczos iteration with the same matrix  $A$  and the initial vectors  $\mathbf{r}_0$  and  $\mathbf{r}_0 + \mathbf{f}$ , where  $\mathbf{f}$  is a given perturbation. The following iteration produces these values.

**Algorithm 1** Given the Jacobi matrix (4.1), the first  $k + 1$  unnormalized Lanczos vectors  $\mathbf{r}_0, \dots, \mathbf{r}_k$ , the value  $\beta_k = \|\mathbf{r}_k\|_2$ , and a vector  $\mathbf{f}$ , the following algorithm generates the modified Jacobi matrix (4.2) that is produced by the unsymmetric Lanczos iteration with left initial vector  $\mathbf{r}_0$  and right initial vector  $\mathbf{r}_0 + \mathbf{f}$ , along with the value  $\hat{\beta}_k$ .

$$\alpha_0 = \hat{\alpha}_0 = 0$$

$$\beta_{-1} = 0$$

$$\mathbf{q}_{-1} = 0$$

$$\mathbf{q}_0 = \mathbf{f}$$

$$\hat{\beta}_0^2 = \beta_0^2 + \mathbf{r}_0^H \mathbf{q}_0$$

$$s_0 = \frac{\beta_0}{\hat{\beta}_0^2}$$

$$t_0 = \frac{\beta_0^2}{\hat{\beta}_0^2}$$

$$d_0 = 0$$

**for**  $j = 1, \dots, k$

$$\hat{\alpha}_j = \alpha_j + s_{j-1} \mathbf{r}_j^H \mathbf{q}_{j-1} + d_{j-1} \beta_{j-2} t_{j-1}^{-1/2}$$

$$d_j = (d_{j-1} \beta_{j-2} + (\alpha_j - \hat{\alpha}_j) t_{j-1}^{1/2}) / \hat{\beta}_{j-1}$$

$$\mathbf{q}_j = (A - \hat{\alpha}_j I) \mathbf{q}_{j-1} - \hat{\beta}_{j-1}^2 \mathbf{q}_{j-2}$$

$$\hat{\beta}_j^2 = t_{j-1} \beta_j^2 + s_{j-1} \mathbf{r}_j^H \mathbf{q}_j$$

$$s_j = \frac{\beta_j}{\hat{\beta}_j^2} s_{j-1}$$

$$t_j = \frac{\beta_j^2}{\hat{\beta}_j^2} t_{j-1}$$

end

We now prove the correctness of this algorithm.

**Theorem 4.1** *Let  $A$  be an  $n \times n$  symmetric positive definite matrix and  $\mathbf{r}_0$  be an  $n$ -vector. Let  $J_K$  be the Jacobi matrix obtained by applying the symmetric Lanczos iteration to  $A$  with initial vector  $\mathbf{r}_0$ , i.e.*

$$AR_K = R_K J_K + \beta_K \mathbf{r}_K, \tag{4.3}$$

where  $R_K = [\mathbf{r}_0 \ \cdots \ \mathbf{r}_{K-1}]$ . Then Algorithm 1 computes the entries of the modified Jacobi matrix  $\hat{J}_K$  obtained by applying the unsymmetric Lanczos iteration to  $A$  with left initial vector  $\mathbf{r}_0$  and right initial vector  $\mathbf{r}_0 + \mathbf{f}$ , along with the value  $\hat{\beta}_K = [\hat{J}_{K+1}]_{K+1,K}$ .

*Proof* It is sufficient to verify the correctness of the recurrence relations

$$\hat{\beta}_j^2 = t_{j-1} \beta_j^2 + s_{j-1} \mathbf{r}_j^H \mathbf{q}_j, \quad j \geq 0, \quad s_{-1} = 0, \tag{4.4}$$

and

$$\hat{\alpha}_j = \alpha_j + s_{j-1} \mathbf{r}_j^H \mathbf{q}_{j-1} + d_{j-1} \beta_{j-2} t_{j-1}^{-1/2}, \quad j \geq 1, \quad \alpha_0 = \hat{\alpha}_0 = 0, \tag{4.5}$$

where the coefficients  $d_j$ ,  $s_j$  and  $t_j$  are as defined in Algorithm 1. Consider the unsymmetric Lanczos iteration

$$\hat{\mathbf{x}}_0 = 0$$

$$\hat{\mathbf{y}}_0 = 0$$

$$\hat{\beta}_0^2 = \hat{\mathbf{p}}_0^H \hat{\mathbf{r}}_0$$

for  $j = 1, \dots, k$

$$\hat{\mathbf{x}}_j = \hat{\mathbf{r}}_{j-1} / \hat{\beta}_{j-1}$$

$$\hat{\mathbf{y}}_j = \hat{\mathbf{p}}_{j-1} / \hat{\beta}_{j-1}$$

$$\hat{\alpha}_j = \hat{\mathbf{y}}_j^H A \hat{\mathbf{x}}_j$$

$$\hat{\mathbf{r}}_j = (A - \hat{\alpha}_j I) \hat{\mathbf{x}}_j - \hat{\beta}_{j-1} \hat{\mathbf{x}}_{j-1}$$

$$\hat{\mathbf{p}}_j = (A - \hat{\alpha}_j I) \hat{\mathbf{y}}_j - \hat{\beta}_{j-1} \hat{\mathbf{y}}_{j-1}$$

$$\hat{\beta}_j^2 = \hat{\mathbf{p}}_j^H \hat{\mathbf{r}}_j$$

end

where  $\hat{\mathbf{r}}_0 = \mathbf{r}_0 + \mathbf{f}$  and  $\hat{\mathbf{p}}_0 = \mathbf{r}_0$ . Clearly,

$$\hat{\beta}_0^2 = \hat{\mathbf{p}}_0^H \hat{\mathbf{r}}_0 = \mathbf{r}_0^H \mathbf{r}_0 + \mathbf{r}_0^H \mathbf{f} = \beta_0^2 + \mathbf{r}_0^H \mathbf{f} = \beta_0^2 + \mathbf{r}_0^H \mathbf{q}_0. \tag{4.6}$$

Let  $j \geq 1$ . To verify the recurrence relations for  $\hat{\alpha}_j$  and  $\hat{\beta}_j^2$ , we must use the relations

$$\hat{\mathbf{p}}_j = c_j \mathbf{r}_j + d_j \mathbf{r}_{j-1} + \cdots, \tag{4.7}$$

and

$$\hat{\mathbf{r}}_j = c_j \mathbf{r}_j + d_j \mathbf{r}_{j-1} + f_j \mathbf{q}_j + \cdots, \tag{4.8}$$

where  $d_j$  is as defined in Algorithm 1, and the coefficients  $c_j$  and  $f_j$  are defined by

$$c_0 = 1, \quad c_j = \frac{\beta_{j-1}}{\hat{\beta}_{j-1}} c_{j-1}, \quad j \geq 1, \tag{4.9}$$

and

$$f_0 = 1, \quad f_j = \frac{1}{\hat{\beta}_{j-1}} c_{j-1}, \quad j \geq 1. \tag{4.10}$$

In the recurrence relations (4.7) and (4.8), the  $+\dots$  signifies terms that will vanish when inner products of the form  $\hat{\mathbf{p}}_j^H \hat{\mathbf{r}}_j$  or  $\hat{\mathbf{p}}_j^H A \hat{\mathbf{r}}_j$  are computed. These relations can be verified by induction; the reader is referred to [15] for details.

It follows from these recurrences that

$$\begin{aligned} \hat{\beta}_j^2 &= \hat{\mathbf{p}}_j^H \hat{\mathbf{r}}_j \\ &= c_j \mathbf{r}_j^H [c_j \mathbf{r}_j + f_j \mathbf{q}_j] \\ &= c_j^2 \beta_j^2 + c_j f_j \mathbf{r}_j^H \mathbf{q}_j \\ &= t_{j-1} \beta_j^2 + s_{j-1} \mathbf{r}_j^H \mathbf{q}_j, \end{aligned}$$

and

$$\begin{aligned} \hat{\alpha}_j &= \hat{\mathbf{y}}_j^H A \hat{\mathbf{x}}_j \\ &= \frac{1}{\hat{\beta}_{j-1}^2} \hat{\mathbf{p}}_{j-1}^H A \hat{\mathbf{r}}_{j-1} \\ &= \frac{1}{\hat{\beta}_{j-1}^2} [c_{j-1} \mathbf{r}_{j-1} + d_{j-1} \mathbf{r}_{j-2}]^H A \hat{\mathbf{r}}_{j-1} \\ &= \frac{1}{\hat{\beta}_{j-1}^2} \{c_{j-1} [\beta_{j-1} \mathbf{r}_j + \alpha_j \mathbf{r}_{j-1}]^H \hat{\mathbf{r}}_{j-1} + d_{j-1} \beta_{j-2} \mathbf{r}_{j-1}^H \hat{\mathbf{r}}_{j-1}\} \\ &= \frac{1}{\hat{\beta}_{j-1}^2} c_{j-1} [\beta_{j-1} f_{j-1} \mathbf{r}_j^H \mathbf{q}_{j-1} + \alpha_j (c_{j-1} \beta_{j-1}^2 + f_{j-1} \mathbf{r}_{j-1}^H \mathbf{q}_{j-1})] \\ &\quad + \frac{\beta_{j-2}}{\hat{\beta}_{j-1}^2} d_{j-1} \mathbf{r}_{j-1}^H \hat{\mathbf{r}}_{j-1} \\ &= \left( t_{j-1} + \frac{1}{\hat{\beta}_{j-1}^2} s_{j-2} \mathbf{r}_{j-1}^H \mathbf{q}_{j-1} \right) \alpha_j + s_{j-1} \mathbf{r}_j^H \mathbf{q}_{j-1} + \frac{\beta_{j-2}}{\hat{\beta}_{j-1}^2} d_{j-1} \mathbf{r}_{j-1}^H \hat{\mathbf{r}}_{j-1} \\ &= \frac{1}{\hat{\beta}_{j-1}^2} (t_{j-2} \beta_j^2 + s_{j-2} \mathbf{r}_{j-1}^H \mathbf{q}_{j-1}) \alpha_j + s_{j-1} \mathbf{r}_j^H \mathbf{q}_{j-1} + \frac{\beta_{j-2}}{\hat{\beta}_{j-1}^2} d_{j-1} \mathbf{r}_{j-1}^H \hat{\mathbf{r}}_{j-1} \\ &= \alpha_j + s_{j-1} \mathbf{r}_j^H \mathbf{q}_{j-1} + \frac{\beta_{j-2}}{\hat{\beta}_{j-1}^2} d_{j-1} \mathbf{r}_{j-1}^H \hat{\mathbf{r}}_{j-1} \\ &= \alpha_j + s_{j-1} \mathbf{r}_j^H \mathbf{q}_{j-1} + \frac{\beta_{j-2} d_{j-1}}{c_{j-1}}. \end{aligned} \tag{Q.E.D.}$$

It should be noted that this iteration produces the updated Jacobi matrix with less information than is required by the modified Chebyshev algorithm described in [6, 19], which employs *modified moments*. The modified Chebyshev algorithm is designed for an arbitrary modification of the measure  $\alpha(\lambda)$  of the underlying integral (2.11). By exploiting our knowledge of the specific modification of  $\alpha(\lambda)$ , we are able to develop a more efficient algorithm. The basic idea is similar to that of an algorithm described by Golub and Gutknecht in [10] that also overcomes the need for extra information required by the modified Chebyshev algorithm. The main difference between Algorithm 1 and the algorithm from [10] is that Algorithm 1 computes the elements of  $\hat{J}_K$  directly from those of  $J_K$ , instead of computing the necessary modified moments as an intermediate step.

### 5 Higher Space Dimensions

The construction of the parametrized family of Jacobi matrices  $\{J_\omega\}_{\omega=-N/2+1}^{N/2-1}$  discussed in Sect. 3 generalizes to higher space dimensions in a straightforward manner. To simplify the presentation, we restrict ourselves to the two-dimensional case and present an example analogous to that presented in Sect. 3.1, computing a  $2 \times 2$  matrix  $J_{\omega_1, \omega_2}$  with entries that depend on a “wave number” in two dimensions,  $(\omega_1, \omega_2)$ .

Consider a second-order self-adjoint differential operator of the form

$$L(x, y, D_x, D_y) = a_2 \Delta + a_0(x, y), \tag{5.1}$$

with symbol

$$L(x, y, \xi, \eta) = -a_2 \xi^2 - a_2 \eta^2 + a_0(x, y). \tag{5.2}$$

We assume that  $a_0(x, y)$  can be written as a truncated Fourier series

$$a_0(x, y) = \sum_{\omega_1=-N/2+1}^{N/2-1} \sum_{\omega_2=-N/2+1}^{N/2-1} e^{i(\omega_1 x + \omega_2 y)} \hat{a}_0(\omega_1, \omega_2). \tag{5.3}$$

We will now apply the symmetric Lanczos iteration to a discretization of this operator with initial vector  $\hat{\mathbf{e}}_{\omega_1, \omega_2}$ , which is a discretization of the function

$$\hat{e}_{\omega_1, \omega_2}(x) = \frac{1}{2\pi} e^{i(\omega_1 x + \omega_2 y)} \tag{5.4}$$

on a uniform grid  $(x_j, y_k) = (jh, kh)$ ,  $j = 0, 1, \dots, N - 1$ ,  $k = 0, 1, \dots, N - 1$  with  $h = 2\pi/N$ . Specifically,

$$[\hat{\mathbf{e}}_{\omega_1, \omega_2}]_{(k-1)(N-1)+j} = \frac{1}{2\pi} e^{i(\omega_1 x_j + \omega_2 y_k)}.$$

For given functions  $f$  and  $g$  that are  $2\pi$ -periodic in both  $x$  and  $y$ , we will approximate the inner product

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

by the discrete inner product  $\langle f, g \rangle_h$  of the corresponding grid functions:

$$\langle f, g \rangle_h = h^2 \sum_{j=0}^{N-1} \sum_{k=0}^{N-1} \overline{f(x_j, y_k)} g(x_j, y_k). \tag{5.5}$$

In applying the Lanczos algorithm, we will examine how  $\alpha_1$ ,  $\beta_1$  and  $\alpha_2$  can be computed as efficiently as possible. We have

$$\begin{aligned} \alpha_1 &= \langle \hat{e}_{\omega_1, \omega_2}, L(x, y, D_x, D_y) \hat{e}_{\omega_1, \omega_2} \rangle_h \\ &= -a_2(\omega_1^2 + \omega_2^2) + \text{Avg } a_0, \end{aligned}$$

where

$$\text{Avg } f = \frac{1}{(2\pi)^2} \int_0^{2\pi} \int_0^{2\pi} f(x, y) dx dy. \tag{5.6}$$

Proceeding to  $\beta_1$ , we have

$$\begin{aligned} \beta_1^2 &= \|(L(x, y, D_x, D_y) - \alpha_1 I) \hat{e}_{\omega_1, \omega_2}\|^2 \\ &= \|a_0(x, y)\|^2 - |\text{Avg } a_0|^2. \end{aligned}$$

Finally, for  $\alpha_2$  we let  $\overline{b(x, y)} = (a_0(x, y) - \text{Avg } a_0) / \beta_1$  and obtain

$$\begin{aligned} \alpha_2 &= \langle b \hat{e}_{\omega_1, \omega_2}, L(x, y, D_x, D_y) b \hat{e}_{\omega_1, \omega_2} \rangle_h \\ &= \left\langle b \hat{e}_{\omega_1, \omega_2}, a_2 \frac{\partial^2 (b \hat{e}_{\omega_1, \omega_2})}{\partial x^2} \right\rangle_h + \left\langle b \hat{e}_{\omega_1, \omega_2}, a_2 \frac{\partial^2 (b \hat{e}_{\omega_1, \omega_2})}{\partial y^2} \right\rangle_h + \langle b \hat{e}_{\omega_1, \omega_2}, a_0 b \hat{e}_{\omega_1, \omega_2} \rangle_h \\ &= a_2 \left[ \left\langle b \hat{e}_{\omega_1, \omega_2}, \frac{\partial^2 (b \hat{e}_{\omega_1, \omega_2})}{\partial x^2} \right\rangle_h + \left\langle b \hat{e}_{\omega_1, \omega_2}, \frac{\partial^2 (b \hat{e}_{\omega_1, \omega_2})}{\partial y^2} \right\rangle_h \right] + \langle b, a_0 b \rangle_h \\ &= a_2 \left[ \left\langle b \hat{e}_{\omega_1, \omega_2}, \frac{\partial^2 b}{\partial x^2} \hat{e}_{\omega_1, \omega_2} \right\rangle_h + 2 \left\langle b \hat{e}_{\omega_1, \omega_2}, (i\omega_1) \frac{\partial b}{\partial x} \hat{e}_{\omega_1, \omega_2} \right\rangle_h + \langle b \hat{e}_{\omega_1, \omega_2}, (i\omega_1)^2 b \hat{e}_{\omega_1, \omega_2} \rangle_h \right. \\ &\quad \left. + \left\langle b \hat{e}_{\omega_1, \omega_2}, \frac{\partial^2 b}{\partial y^2} \hat{e}_{\omega_1, \omega_2} \right\rangle_h + 2 \left\langle b \hat{e}_{\omega_1, \omega_2}, (i\omega_2) \frac{\partial b}{\partial y} \hat{e}_{\omega_1, \omega_2} \right\rangle_h + \langle b \hat{e}_{\omega_1, \omega_2}, (i\omega_2)^2 b \hat{e}_{\omega_1, \omega_2} \rangle_h \right] \\ &\quad + \langle b, a_0 b \rangle_h \\ &= a_2 \left[ \left\langle b, \frac{\partial^2 b}{\partial x^2} \right\rangle_h + 2 \left\langle b, (i\omega_1) \frac{\partial b}{\partial x} \right\rangle_h + \langle b, (i\omega_1)^2 b \rangle_h \right. \\ &\quad \left. + \left\langle b, \frac{\partial^2 b}{\partial y^2} \right\rangle_h + 2 \left\langle b, (i\omega_2) \frac{\partial b}{\partial y} \right\rangle_h + \langle b, (i\omega_2)^2 b \rangle_h \right] + \langle b, a_0 b \rangle_h \\ &= a_2 \left[ - \left\langle \frac{\partial b}{\partial x}, \frac{\partial b}{\partial x} \right\rangle_h + 2i\omega_1 \left\langle b, \frac{\partial b}{\partial x} \right\rangle_h - \omega_1^2 - \left\langle \frac{\partial b}{\partial y}, \frac{\partial b}{\partial y} \right\rangle_h + 2i\omega_2 \left\langle b, \frac{\partial b}{\partial y} \right\rangle_h - \omega_2^2 \right] \\ &\quad + \langle b, a_0 b \rangle_h \\ &= -a_2(\omega_1^2 + \omega_2^2 + \|b_x\|_h^2 + \|b_y\|_h^2) + \langle b, a_0 b \rangle_h. \end{aligned}$$

We see that, so far, the entries of the tridiagonal matrix constructed by the Lanczos iteration are polynomials in  $\omega_1$  and  $\omega_2$ . While this does not hold in general, the entries can still be represented as algebraic functions of  $\omega_1$  and  $\omega_2$ . Therefore, we can construct the  $K$ -point Gaussian quadrature rules for all frequencies  $(\omega_1, \omega_2)$ , for  $\omega_1, \omega_2 = -N/2 + 1, \dots, N/2 - 1$ , by first constructing representations for the elements  $\alpha_j, j = 1, \dots, K$ , and  $\beta_j, j = 1, \dots, K - 1$ , as functions of  $\omega_1$  and  $\omega_2$ , and then evaluating these representations

for each pair  $(\omega_1, \omega_2)$ . By computing a representation of  $\beta_K$  as well, we can obtain  $K$ -point Gauss-Radau rules.

Once  $J_{\omega_1, \omega_2}$  is obtained, Algorithm 1 from Sect. 4 can be applied as in the one-dimensional case to compute  $J_{\omega_1, \omega_2, n}$ , where again each element of  $J_{\omega_1, \omega_2, n}$  is a 2-parameter family and the matrix-vector multiplication used to compute  $\mathbf{q}_j$  is replaced by application of a discretization of the operator  $L(x, y, D_x, D_y)$  to  $\mathbf{q}_{j-1}$ .

### 6 Source Terms

We now consider initial-boundary value problems of the form

$$\frac{\partial u}{\partial t}(x, t) + L(x, D)u(x, t) = F(x, t), \quad 0 < x < 2\pi, \quad t > 0, \tag{6.1}$$

with initial and boundary conditions specified by (1.2) and (1.3) as before. The exact solution can be expressed using Duhamel’s Principle,

$$u(x, t) = e^{-L(x, D)t} f(x) + \int_0^t e^{-L(x, D)(t-\tau)} F(x, \tau) d\tau. \tag{6.2}$$

We can apply Krylov subspace spectral methods to solve a problem of this form by computing the first term on the right side of (6.2) as before, since it is merely the solution of the homogeneous analogue of (6.1), and then approximating the integral using a quadrature rule on the interval  $[0, \Delta t]$  to obtain the solution at time  $t + \Delta t$ .

More precisely, if  $t_1, \dots, t_M$  are nodes of an  $M$ -point rule, with corresponding weights  $w_1, \dots, w_M$ , then our approximate solution  $\tilde{u}(x, t + \Delta t)$  has the form

$$\tilde{u}(x, t + \Delta t) = \tilde{S}(x, D, \Delta t, \tilde{u}(x, t)) + \sum_{k=1}^M w_k \tilde{S}(x, D, \Delta t - t_k, F(x, t_k)), \tag{6.3}$$

where for a given function  $f(x)$ ,  $S(x, D, t, f(x))$  denotes the approximate solution at time  $t$  of the homogeneous problem (1.1), (1.2), (1.3) obtained using a Krylov subspace spectral method. It should be noted that the Jacobi matrices  $\{J_\omega\}$  can be computed once and re-used for each term in (6.3).

An equally efficient but more accurate approach is to combine the computation of the solution of the homogeneous problem with the evolution of the source term. For convenience, we let  $t_{k+1} = \Delta t$ .

$$u^{(0)}(x, t) = \tilde{S}(x, D, t_1, \tilde{u}(x, t))$$

**for**  $k = 1, \dots, M$  **do**

$$u^{(k-1)}(x, t) = u^{(k-1)}(x, t) + w_k F(x, t + t_k)$$

$$u^{(k)}(x, t) = \tilde{S}(x, D, t_{k+1} - t_k, \tilde{u}^{(k-1)}(x, t))$$

**end**

The algorithm entails the same number of applications of  $\tilde{S}(x, D, \cdot, \cdot)$  as the previous procedure (6.3), but it effectively computes the solution of the homogeneous problem using a time step of average width  $\Delta t/M$  instead of  $\Delta t$ .

## 7 Numerical Results

In this section, we will present numerical results for comparisons, in terms of both accuracy and efficiency, of 2-node Krylov subspace spectral methods with other well-established methods for solving parabolic problems. The comparisons will focus on the accuracy of the temporal approximations employed by each method, for these reasons:

- Parabolic problems tend to yield smooth solutions, so a Fourier interpolant defined on a grid of reasonable size is sufficient to accurately represent the solution, and
- More importantly, the effect of spatial discretization error on the accuracy of Krylov subspace spectral methods is not straightforward to quantify, as it is two-fold. Such error affects the recursion coefficients discussed in Sect. 3, which in turn affect the quadrature nodes and weights, and therefore the accuracy of the temporal approximation of each component. The truncation of the Fourier series introduces additional spatial error, depending on the smoothness of the solution. As this paper concerns the efficient implementation of these methods, a thorough analysis of the spatial discretization error will be deferred for future work.

### 7.1 Construction of Test Cases

We introduce some differential operators and functions that will be used in 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` and after setting the generator to its initial state. These numbers are uniformly distributed on the interval  $(0, 1)$ .

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

$$f_{j,k}^0(x) = \text{Re} \left\{ \sum_{|\omega| < N/2, \omega \neq 0} \hat{f}_j(\omega) (1 + |\omega|)^{-(k+1)} e^{i\omega x} \right\}, \quad j, k = 0, 1, \dots, \tag{7.1}$$

where

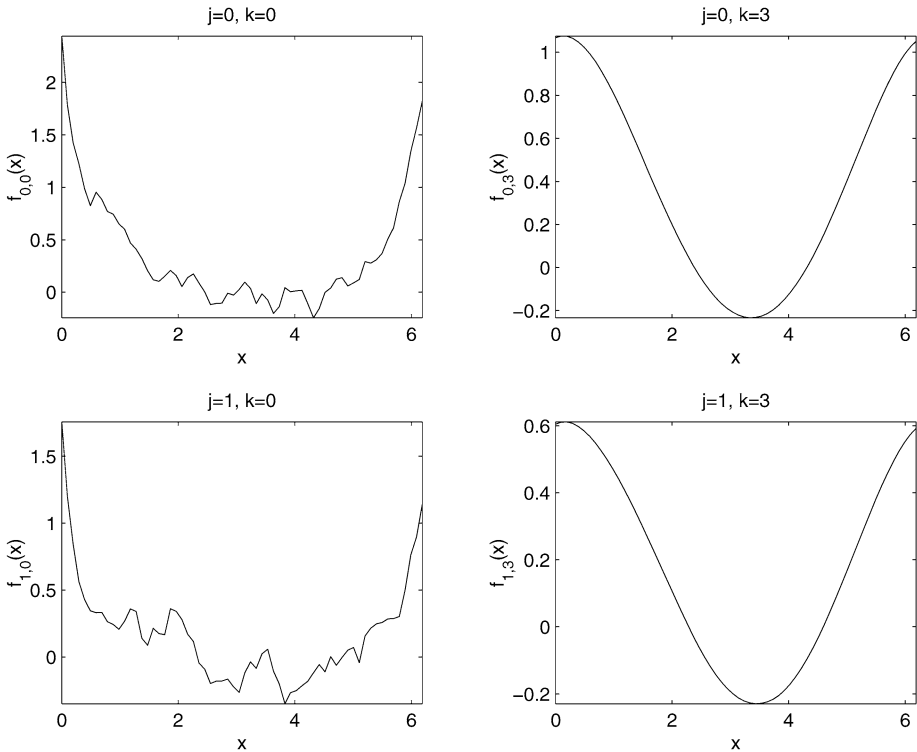
$$\hat{f}_j(\omega) = R_{jN+2(\omega+N/2)-1} + i R_{jN+2(\omega+N/2)}. \tag{7.2}$$

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 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

$$E^+ f(x) = f(x) - \min_{x \in [0, 2\pi]} f(x) + 1, \tag{7.3}$$

$$E^- f(x) = f(x) - \max_{x \in [0, 2\pi]} f(x) - 1. \tag{7.4}$$



**Fig. 1** Functions from the collection  $f_{j,k}(x)$ , for selected values of  $j$  and  $k$

- We define a similar two-parameter family of functions defined on the rectangle  $[0, 2\pi] \times [0, 2\pi]$ :

$$g_{j,k}(x, y) = \operatorname{Re} \left\{ \sum_{|\omega|, |\xi| < N/2, \omega\xi \neq 0} \hat{g}_j(\omega, \xi) (1 + |\omega|)^{-(k+1)} (1 + |\xi|)^{-(k+1)} e^{i(\omega x + \xi y)} \right\}, \quad (7.5)$$

where  $j$  and  $k$  are nonnegative integers, and

$$\begin{aligned} \hat{g}_j(\omega, \xi) = & R_{jN^2+2[N(\omega+N/2-1)+(\xi+N/2)]-1} \\ & + i R_{jN^2+2[N(\omega+N/2-1)+(\xi+N/2)]}. \end{aligned} \quad (7.6)$$

Figure 2 shows selected functions from this collection.

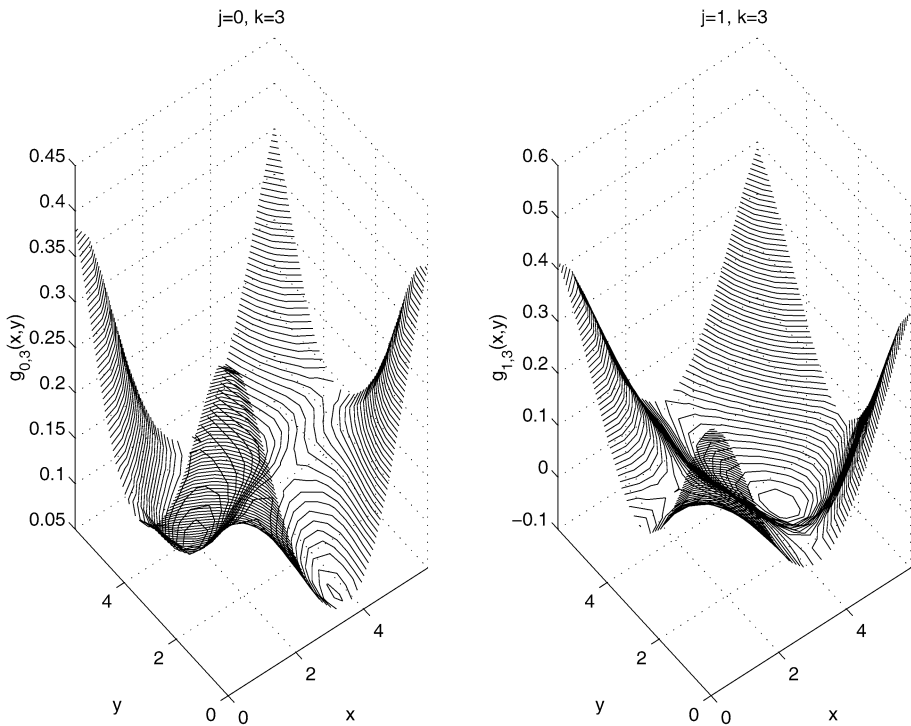
- Experiments will involve the one-parameter family of randomly generated self-adjoint differential operators

$$L_k(x, D) = D(E^- f_{0,k} D) + E^+ f_{1,k}, \quad k = 0, 1, \dots, \quad (7.7)$$

where the operators  $E^+$  and  $E^-$  were defined in (7.3), (7.4).

- Another one-parameter family of differential operators that we will use is defined by

$$M_j(x, D) = D(E^- [D^j f_{0,3}] D) + E^+ [D^j f_{1,3}], \quad j = 0, 1, \dots \quad (7.8)$$



**Fig. 2** Functions from the collection  $g_{j,k}(x, y)$ , for selected values of  $j$  and  $k$

In all experiments,  $N = 64$  grid points are used, and solutions  $u^{(j)}(x, t)$  are computed using time steps  $\Delta t = 2^{-j}$ , for  $j = 0, \dots, 6$ . The error estimates are obtained by computing  $\|u^{(j)}(\cdot, 1) - u^{(6)}(\cdot, 1)\| / \|u^{(6)}(\cdot, 1)\|$ . This method of estimating error assumes that  $u^{(6)}(x, t)$  is a sufficiently accurate approximation to the exact solution, but this has proven in practice to be a valid assumption by comparing  $u^{(6)}$  against approximate solutions computed using established methods, and by comparing  $u^{(6)}$  against solutions obtained using various methods with smaller time steps. It should be noted that we are not seeking a sharp estimate of the error, but rather an indication of the rate of convergence, and for this goal, using  $u^{(6)}$  as an approximation to the exact solution is sufficient.

### 7.2 Problems in Higher Space Dimensions

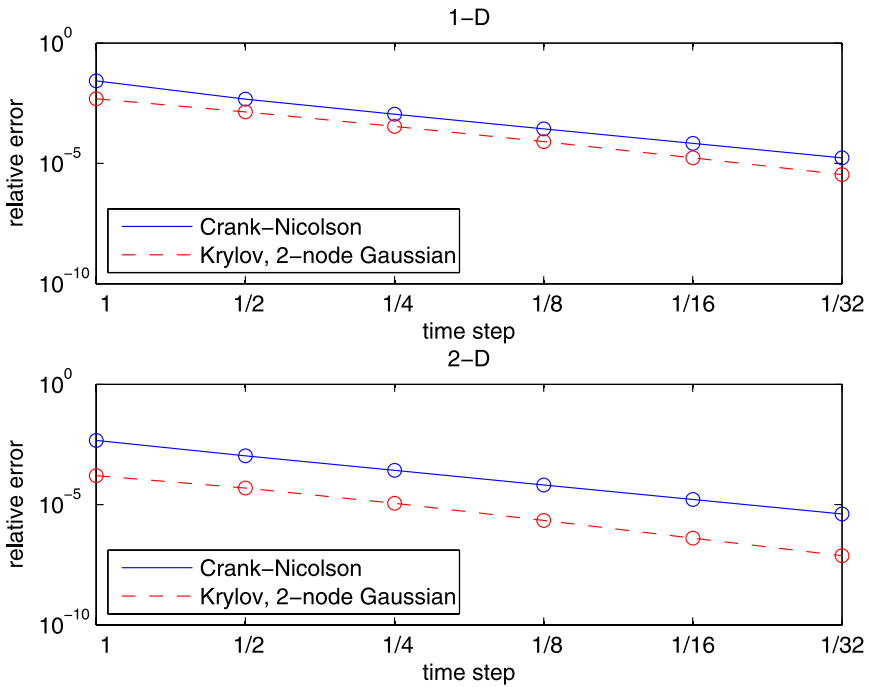
We now show the accuracy of our approach for higher space dimensions. We solve a variable-coefficient heat equation with randomly generated coefficients as discussed in Sect. 7.1.

First, we solve the following parabolic problems:

- $\frac{\partial u}{\partial t}(x, t) - \frac{\partial^2 u}{\partial x^2}(x, t) - E^- f_{1,2}(x)u(x, t) = 0, \quad 0 < x < 2\pi, \quad 0 < y < 2\pi, \quad t > 0, \quad (7.9)$

$$u(x, 0) = E^+ f_{0,3}(x), \quad 0 < x < 2\pi, \quad (7.10)$$

$$u(x, t) = u(x + 2\pi, t), \quad t > 0, \quad (7.11)$$



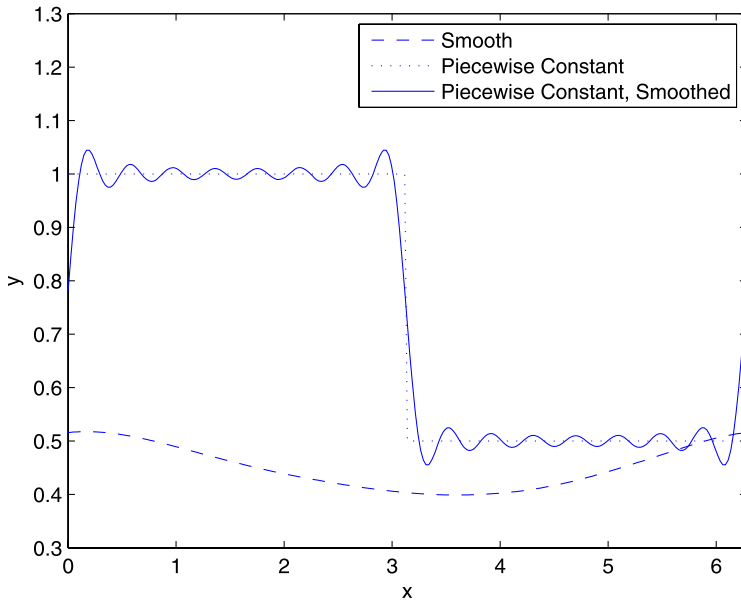
**Fig. 3** **a** *Top plot*: Estimates of relative error in the approximate solution of (7.9), (7.10), (7.11) at  $T = 1$ . Solutions are computed using the Crank-Nicolson method (solid curve), and a Krylov subspace spectral method with 2 Gaussian quadrature nodes (dashed curve). **b** *Bottom plot*: Estimates of relative error in the approximate solution of (7.12), (7.13), (7.14) at  $t = 1$ . Solutions are computed using the Crank-Nicolson method (solid curve), and a Krylov subspace spectral method with 2 Gaussian quadrature nodes (dashed curve). In both cases, both methods use  $N = 64$  grid points per dimension are used, and time steps  $\Delta t = 2^{-j}$ ,  $j = 0, \dots, 6$

**Table 1** Estimates of relative error in the approximate solution of (7.9), (7.10), (7.11) at  $T = 1$ . Error is the relative difference, in the 2-norm sense, between solutions computed with successive time steps, since no exact solution is available. Both methods use  $N = 64$  grid points, and time steps  $\Delta t = 2^{-j}$ ,  $j = 0, \dots, 6$

Method	$\Delta t$	Error	Order
KSS(2)	1	4.8e-3	2.34
	1/2	1.4e-3	
	1/4	3.4e-4	
	1/8	7.9e-5	
	1/16	1.7e-5	
	1/32	3.3e-6	
Crank-Nicolson	1	2.7e-2	1.99
	1/2	4.7e-3	
	1/4	1.1e-3	
	1/8	2.7e-4	
	1/16	6.8e-5	
	1/32	1.7e-5	

**Table 2** Estimates of relative error in the approximate solution of (7.9), (7.13), (7.14) at  $T = 1$ . Error is the relative difference, in the 2-norm sense, between solutions computed with successive time steps. Both methods use  $N = 64$  grid points per dimension, and time steps  $\Delta t = 2^{-j}$ ,  $j = 0, \dots, 6$

Method	$\Delta t$	Error	Order
KSS(2)	1	1.6e-4	2.42
	1/2	4.9e-5	
	1/4	1.1e-5	
	1/8	2.2e-6	
	1/16	4.1e-7	
	1/32	7.6e-8	
Crank-Nicolson	1	4.7e-3	1.99
	1/2	1.1e-3	
	1/4	2.6e-4	
	1/8	6.6e-5	
	1/16	1.6e-5	
	1/32	4.1e-6	



**Fig. 4** The functions used as the coefficient  $a(x)$  in the problem (7.15). The dashed curve is the smooth coefficient  $a_s(x)$ . The dotted curve is the piecewise constant coefficient  $a_{pc}(x)$ . The solid curve is the Fourier interpolant of  $a_{pc}(x)$ , denoted  $\tilde{a}_{pc}(x)$

$$\bullet \frac{\partial u}{\partial t}(x, y, t) - \Delta u(x, y, t) - E^- g_{3,2}(x, y)u(x, t) = 0, \quad (x, y) \in (0, 2\pi) \times (0, 2\pi), \quad t > 0, \tag{7.12}$$

$$u(x, y, 0) = E^+ g_{3,3}(x, y), \quad (x, y) \in (0, 2\pi) \times (0, 2\pi), \tag{7.13}$$

$$u(x, y, t) = u(x + 2\pi, t) = u(x, y + 2\pi, t), \quad t > 0. \tag{7.14}$$

**Table 3** Accuracy and efficiency of a 2-node Krylov subspace spectral method (denoted by KSS(2)) and the MATLAB ODE solver `ode23s`, applied to the problem (7.15) with  $a(x) = a_s(x)$ . Error is the relative difference, in the 2-norm sense, between the exact solution  $u(x, t) = \cos(x - t)$  and the computed solution at  $T = 1$ .  $N$  denotes the number of grid points.  $\Delta t$  denotes the maximum time step permitted. Time is execution time in seconds

Method	$N$	$\Delta t$	Error	Time	Order
KSS(2)	64	1/32	4.7e-7	0.04	2.81
	64	1/64	6.6e-8	0.08	
	64	1/128	9.5e-9	0.17	
	128	1/64	2.3e-8	0.13	2.36
	128	1/128	4.3e-9	0.26	
	128	1/256	8.7e-10	0.51	
	256	1/64	2.0e-8	0.23	
	256	1/128	3.4e-9	0.46	2.44
	256	1/256	6.8e-10	0.95	
	ode23s	64	1/4	2.8e-3	0.05
64		1/8	7.1e-4	0.09	
64		1/16	1.8e-4	0.18	
128		1/4	2.7e-3	0.12	1.99
128		1/8	6.7e-4	0.23	
128		1/16	1.7e-4	0.45	
256		1/4	2.7e-3	0.42	
256		1/8	6.8e-4	0.73	1.99
256		1/16	1.7e-4	1.49	

The results are shown in Fig. 3 and Tables 1 and 2, and compared to those obtained using the Crank-Nicolson method. In the 2-D case, the variable coefficient of the PDE is smoothed to a greater extent than in the 1-D case, because the prescribed decay rate of the Fourier coefficients is imposed in both the  $x$ - and  $y$ -directions. This results in greater accuracy in the 2-D case, which is consistent with the result proved in [16] that the local truncation error varies linearly with the variation in the coefficients.

As with previous results (see [16]), greater accuracy is obtained with Krylov subspace spectral methods, in both one and two space dimensions. The Crank-Nicolson method is second-order accurate in both time and space; in the results presented in Fig. 3, the Krylov subspace spectral method is converging with temporal order 2.34 in 1-D, and 2.42 in 2-D.

In 1-D, even greater accuracy can be obtained by prescribing an additional node that approximates the smallest eigenvalue of the spatial differentiation operator  $L(x, D)$  in (1.1). This is not the case in 2-D; this remains under investigation.

### 7.3 Performance Comparison

We compare the performance of a 2-node Krylov subspace spectral method to ODE solvers available in MATLAB. We consider the problem

$$u_t = (a(x)u_x)_x + F(x, t), \quad 0 < x < 2\pi, \quad t > 0, \quad (7.15)$$

**Table 4** Accuracy and efficiency of a 2-node Krylov subspace spectral method (denoted by KSS(2)) and the MATLAB ODE solver ode23s, applied to the problem (7.15) with  $a(x) = a_{pc}(x)$ . Error is the relative difference, in the 2-norm sense, between the exact solution  $u(x, t) = \cos(x - t)$  and the computed solution at  $T = 1$ .  $N$  denotes the number of grid points.  $\Delta t$  denotes the maximum time step permitted. Time is execution time in seconds

Method	$N$	$\Delta t$	Error	Time	Order
KSS(2)	64	1/512	1.3e-5	0.63	
	64	1/1024	1.9e-6	1.28	2.82
	64	1/2048	2.6e-7	2.57	
	128	1/512	2.1e-4	1.16	
	128	1/1024	3.6e-5	2.14	2.60
	128	1/2048	5.7e-6	4.10	
	256	1/512	3.7e-3	1.86	
	256	1/1024	6.7e-4	3.67	2.53
	256	1/2048	1.1e-4	7.27	
	ode23s	64	1/64	1.2e-5	0.70
64		1/128	3.0e-6	1.42	2.00
64		1/256	7.5e-7	2.79	
128		1/32	4.8e-5	0.90	
128		1/64	1.2e-5	1.82	2.00
128		1/128	3.0e-6	3.64	
256		1/16	1.9e-4	1.54	
256		1/32	4.8e-5	3.05	1.99
256		1/64	1.2e-5	6.45	

where

$$F(x, t) = \sin(x - t) + (a(x) \sin(x - t))_x. \tag{7.16}$$

With periodic boundary conditions and the initial condition  $u(x, 0) = \cos x$ , the exact solution is  $\cos(x - t)$ . For  $a(x)$ , we use the following:

- A smooth function  $a_s(x) = E^+ f_{0,3}^0(x)$ , as defined in (7.1).
- A piecewise constant function

$$a_{pc}(x) = \begin{cases} 1, & 0 \leq x < \pi, \\ 1/2, & \pi \leq x < 2\pi. \end{cases} \tag{7.17}$$

- The Fourier interpolant of  $a_2(x)$ , with the interpolation points  $x_j = jh, j = 0, 1, \dots, 31$ , where  $h = \pi/16$ . We denote this interpolant by  $\tilde{a}_{pc}(x)$ .

All three coefficients are shown in Fig. 4.

For stiff problems such as this, many of the ODE solvers included in MATLAB are not suitable, if one is hoping to achieve high-order accuracy. The solvers of interest are ode45, ode23s, and ode15s. The first of these, ode45, is capable of achieving high accuracy in a reasonable amount of computing time if we are solving (7.15) over a relatively small interval in time, such as  $[0, 1]$ . However, as  $T$  increases to higher orders of magnitude, this is no longer the case, and it ceases to be competitive with the stiff solvers such as ode23s. Therefore, we only report results for the two stiff solvers.

**Table 5** Accuracy and efficiency of a 2-node Krylov subspace spectral method (denoted by KSS(2)) and the MATLAB ODE solver `ode23s`, applied to the problem (7.15) with  $a(x) = \tilde{a}_{pc}(x)$ . Error is the relative difference, in the 2-norm sense, between the exact solution  $u(x, t) = \cos(x - t)$  and the computed solution at  $T = 1$ .  $N$  denotes the number of grid points.  $\Delta t$  denotes the maximum time step permitted. Time is execution time in seconds

Method	$N$	$\Delta t$	Error	Time	Order
KSS(2)	64	1/512	9.1e-7	0.68	
	64	1/1024	1.2e-7	1.35	3.25
	64	1/2048	1.7e-8	2.67	
	128	1/512	8.0e-7	1.13	
	128	1/1024	1.1e-7	2.23	2.87
	128	1/2048	1.5e-8	4.45	
	256	1/512	7.8e-7	1.96	
	256	1/1024	1.1e-7	3.89	2.90
	256	1/2048	1.4e-8	7.68	
	ode23s	64	1/64	1.2e-5	0.71
64		1/128	3.0e-6	1.44	1.99
64		1/256	7.6e-7	2.75	
128		1/32	4.8e-5	0.88	
128		1/64	1.2e-5	1.77	2.00
128		1/128	3.0e-6	3.54	
256		1/16	1.9e-4	1.44	
256		1/32	4.8e-5	2.87	1.99
256		1/64	1.2e-5	5.72	

### 7.3.1 Short-time Performance

We begin with results for a short interval in time. Table 3 demonstrates the accuracy and efficiency of a 2-node Krylov subspace spectral method, compared to that of `ode23s`, for the case of a smooth coefficient  $a(x) = a_s(x)$ , on a short time interval  $[0, 1]$ . For comparable execution times, the Krylov subspace spectral method is more accurate by several orders of magnitude, with a higher convergence rate, and the gap in both accuracy and efficiency increases with the number of grid points. It should also be noted that as the number of grid points doubles, the execution time of `ode23s` increases by a factor somewhat greater than two, while that of the Krylov subspace spectral method increases by a factor significantly less than two.

Table 4 demonstrates the accuracy and efficiency of a 2-node Krylov subspace spectral method, compared to that of `ode23s`, for the case of a piecewise constant coefficient  $a(x) = a_{pc}(x)$ , on a short time interval  $[0, 1]$ . For the coarsest grid, the Krylov subspace spectral method is slightly better in terms of both accuracy and efficiency, but this advantage is lost to `ode23s` as the number of grid points increases. Even though its rate of convergence is higher, the error at larger time steps is too great to overcome without substantial refinement of the temporal grid.

Table 5 demonstrates the accuracy and efficiency of a 2-node Krylov subspace spectral method, compared to that of `ode23s`, for the case of a Fourier interpolant of a piecewise constant coefficient,  $a(x) = a_{pp}(x)$ , on a short time interval  $[0, 1]$ . For comparable execution times, the Krylov subspace spectral method achieves much higher accuracy and a faster rate

**Table 6** Accuracy and efficiency of a 2-node Krylov subspace spectral method (denoted by KSS(2)) and the MATLAB ODE solver `ode23s`, applied to the problem (7.15) with  $a(x) = a_s(x)$ . Error is the relative difference, in the 2-norm sense, between the exact solution  $u(x, t) = \cos(x - t)$  and the computed solution at  $T = 100$ .  $N$  denotes the number of grid points.  $\Delta t$  denotes the maximum time step permitted. Time is execution time in seconds

Method	$N$	$\Delta t$	Error	Time	Order
KSS(2)	64	1/32	2.4e-7	4.26	2.72
	64	1/64	3.6e-8	8.44	
	64	1/128	5.5e-9	16.82	
	128	1/32	1.6e-7	7.07	2.66
	128	1/64	2.5e-8	14.03	
	128	1/128	4.0e-9	29.25	
	256	1/32	1.7e-7	12.38	
	256	1/64	2.3e-8	24.74	2.80
	256	1/128	3.5e-9	50.10	
	ode23s	64	1/4	3.1e-3	4.34
64		1/8	7.9e-4	8.71	
64		1/16	2.0e-4	17.18	
128		1/2	1.3e-2	5.62	2.01
128		1/4	3.2e-3	11.24	
128		1/8	8.0e-4	22.72	
256		1	5.0e-2	9.15	
256		1/2	1.3e-2	18.32	1.96
256		1/4	3.3e-3	36.62	

of convergence, and the gap in accuracy and efficiency increases with the number of grid points. For a fixed time step, doubling the grid points roughly triples the execution time of `ode23s`, while the execution time for the Krylov subspace spectral method increases by a factor of less than two.

### 7.3.2 Long-time Performance

Now, we solve the same problems, with the same methods, over a much longer time interval. Table 6 demonstrates the accuracy and efficiency of a 2-node Krylov subspace spectral method, compared to that of `ode23s`, for the case of a smooth coefficient  $a(x) = a_s(x)$ , on a long time interval  $[0, 100]$ . As in the case of a short time interval, it is no contest; for comparable execution time, the Krylov subspace spectral method achieves much greater accuracy and a higher convergence rate than `ode23s`. Furthermore, as the number of grid points doubles, for a fixed time step, the execution time for `ode23s` approximately triples, while it increases by a factor of slightly less than two for the Krylov subspace spectral method.

Table 7 demonstrates the accuracy and efficiency of a 2-node Krylov subspace spectral method, compared to that of `ode23s`, for the case of a piecewise constant coefficient  $a(x) = a_{pc}(x)$ , on a long time interval  $[0, 100]$ . In this case, as the number of grid points increases, `ode23s` manages to deliver slightly better accuracy in somewhat less execution time, but its convergence rate slows considerably, while the Krylov subspace spectral method maintains

**Table 7** Accuracy and efficiency of a 2-node Krylov subspace spectral method (denoted by KSS(2)) and the MATLAB ODE solver `ode23s`, applied to the problem (7.15) with  $a(x) = a_{pc}(x)$ . Error is the relative difference, in the 2-norm sense, between the exact solution  $u(x, t) = \cos(x - t)$  and the computed solution at  $T = 100$ .  $N$  denotes the number of grid points.  $\Delta t$  denotes the maximum time step permitted. Time is execution time in seconds

Method	$N$	$\Delta t$	Error	Time	Order
KSS(2)	64	1/512	1.3e-5	66.36	2.82
	64	1/1024	1.9e-6	132.37	
	64	1/2048	2.6e-7	264.43	
	128	1/512	2.3e-4	111.84	2.60
	128	1/1024	4.0e-5	222.63	
	128	1/2048	6.3e-6	442.51	
	256	1/512	2.9e-3	200.85	
	256	1/1024	6.3e-4	399.53	2.36
	256	1/2048	1.1e-4	792.99	
	64	1/64	5.3e-5	69.11	
ode23s	64	1/128	2.3e-5	137.81	2.73
	64	1/256	1.2e-6	277.30	
	128	1/32	1.3e-4	89.12	
	128	1/64	1.1e-4	178.79	0.12
	128	1/128	1.4e-4	357.74	
	256	1/16	1.3e-3	147.19	
	256	1/32	3.3e-4	294.31	
	256	1/64	7.2e-4	590.40	0.42

a healthy convergence rate. Unfortunately, both methods suffer from the roughness of the coefficient.

Table 8 demonstrates the accuracy and efficiency of a 2-node Krylov subspace spectral method, compared to that of `ode23s`, for the case of a Fourier interpolant of a piecewise constant coefficient,  $a(x) = a_{pp}(x)$ , on a long time interval  $[0, 100]$ . As in the case of the short time interval, the Krylov subspace spectral method is able to achieve high accuracy even as the grid spacing increases, without decreasing the time step. The accuracy is comparable to that of the cases with a smooth coefficient, even though this coefficient exhibits steeper gradients and oscillations. Meanwhile, for comparable execution times, `ode23s` fails to achieve convergence at all.

### 7.3.3 Comparison with `ode15s`

The stiff solver `ode15s`, when used with sufficiently small values of the parameters `MaxStep` and `RelTol`, is capable of achieving very high accuracy for the problem (7.15) over long time intervals, in far less time than `ode23s`, thus presenting a challenge to Krylov subspace spectral methods to remain competitive. A close examination of the algorithm employed by `ode15s` reveals why it is so much faster than `ode23s`, and how Krylov subspace spectral methods can achieve a similar speedup.

As described in [20], `ode15s` solves an ODE of the form  $\mathbf{y}' = f(t, \mathbf{y})$ ,  $\mathbf{y}(t_0) = \mathbf{y}_0$  by computing the Jacobian of  $f$  at  $(t_0, \mathbf{y}_0)$ , which is then used in a Newton iteration to compute the solution at successive times, recomputing the Jacobian as needed. If  $f$  is independent

**Table 8** Accuracy and efficiency of a 2-node Krylov subspace spectral method (denoted by KSS(2)) and the MATLAB ODE solver `ode23s`, applied to the problem (7.15) with  $a(x) = \tilde{a}_{pc}(x)$ . Error is the relative difference, in the 2-norm sense, between the exact solution  $u(x, t) = \cos(x - t)$  and the computed solution at  $T = 100$ .  $N$  denotes the number of grid points.  $\Delta t$  denotes the maximum time step permitted. Time is execution time in seconds

Method	$N$	$\Delta t$	Error	Time	Order
KSS(2)	64	1/512	7.5e-7	66.24	
	64	1/1024	1.0e-7	131.49	2.93
	64	1/2048	1.3e-8	262.65	
	128	1/512	6.9e-7	110.96	
	128	1/1024	9.3e-8	221.43	2.81
	128	1/2048	1.4e-8	441.42	
	256	1/512	6.8e-7	194.12	
	256	1/1024	9.1e-8	386.24	2.91
	256	1/2048	1.2e-8	765.14	
ode23s	64	1/64	1.6e-5	69.31	
	64	1/128	1.7e-5	137.59	1.87
	64	1/256	1.2e-6	277.50	
	128	1/32	6.3e-5	88.93	
	128	1/64	4.2e-5	179.17	-0.03
	128	1/128	6.6e-5	357.43	
	256	1/16	5.8e-4	147.54	
	256	1/32	5.2e-4	294.13	0.20
	256	1/64	4.4e-4	585.73	

of  $t$ , as in the problem (7.15), then no such recomputation is necessary. The most expensive portion of the Newton iterations, the solution of triangular systems obtained from the Jacobian, is performed in the LAPACK layer of MATLAB. Therefore, `ode15s` is much faster than the other ODE solvers such as `ode23s`, which rely more heavily on slower compiled code from MATLAB M-files, even though they perform fewer floating-point operations.

We therefore modify the implementation of Krylov subspace spectral methods in order to reduce execution time. First, we select a time step  $\Delta t$  and grid size  $N$ . Then, for each function of the form  $\cos \omega x$  or  $\sin \omega x$ , for  $\omega = 0, 1, \dots, N/2$  (excluding  $\omega = 0$  and  $\omega = N/2$  for the sines), we take a single time step of length  $\Delta t$  with the given function as initial data. These solutions are used to build a matrix that serves as a solution operator. Then, each time step consists of a matrix-vector multiplication, or a few matrix-vector multiplications and vector additions in the case of a source term, since unequal time steps are used to account for the source term as accurately as possible.

Table 9 demonstrates the accuracy and efficiency of a 2-node Krylov subspace spectral method, compared to that of `ode15s`, for the case of a smooth coefficient,  $a(x) = a_s(x)$ , on a short time interval  $[0, 1]$ . We see that the Krylov subspace spectral method delivers greater accuracy in far less time than `ode15s`, and that the gap in performance widens as the number of grid points increases, even though it must expend 4 times as much computational effort to compute the solution operator before taking any time steps.

Table 10 demonstrates the accuracy and efficiency of a 2-node Krylov subspace spectral method, compared to that of `ode15s`, for the case of a smooth coefficient,  $a(x) = a_s(x)$ , on a long time interval  $[0, 100]$ . This time, `ode15s` performs more effectively, especially on

**Table 9** Accuracy and efficiency of a 2-node Krylov subspace spectral method (denoted by KSS(2)) and the MATLAB ODE solver `ode15s`, applied to the problem (7.15) with  $a(x) = a_s(x)$ . Error is the relative difference, in the 2-norm sense, between the exact solution  $u(x, t) = \cos(x - t)$  and the computed solution at  $T = 1$ .  $N$  denotes the number of grid points.  $\Delta t$  denotes the maximum time step permitted. Time is execution time in seconds

Method	$N$	$\Delta t$	Error	Time	Order
KSS(2)	64	1/32	4.7e-7	0.06	2.81
	64	1/64	6.6e-8	0.07	
	64	1/128	9.6e-9	0.08	
	128	1/32	2.4e-7	0.22	2.74
	128	1/64	3.4e-8	0.22	
	128	1/128	5.4e-9	0.24	
	256	1/32	9.8e-8	0.84	2.56
	256	1/64	1.6e-8	0.86	
	256	1/128	2.8e-9	0.89	
	ode15s	64	1/1024	5.7e-7	0.50
64		1/2048	1.4e-7	1.54	
64		1/4096	3.6e-8	3.06	
128		1/16	6.0e-7	0.60	1.99
128		1/32	1.5e-7	1.20	
128		1/64	3.8e-8	3.88	
256		1/16	6.4e-7	0.97	2.00
256		1/32	1.6e-7	1.81	
256		1/64	4.0e-8	6.09	

finer grids, in terms of achieving greater accuracy in less time. However, the performance of the Krylov subspace spectral method is at least comparable. It is important to note that the approach employed by `ode15s` would be less practical for problems in higher space dimension, as the matrices involved would grow much larger and ill-conditioned. Furthermore, the implementation of the Krylov subspace spectral method used in these experiments would not perform as efficiently as a FORTRAN implementation of the algorithm used in the comparisons with `ode23s`, which does not require working with dense matrices.

## 8 Conclusions

We have succeeded in designing an implementation of Krylov subspace spectral methods that requires  $O(N \log N)$  floating-point operations per time step. This implementation exploits the structure of differential operators to simultaneously compute recursion coefficients for quadrature rules with respect to measures that are defined for each wave number. Since these methods are based on perturbations of Krylov subspaces in the direction of the solution from the previous time step, we also developed a fast algorithm for updating these recursion coefficients using simple recurrence relations and applications of the spatial operator to the previous solution. This efficient process, in conjunction with the individual attention devoted to each component in order to improve stability, makes Krylov subspace spectral methods viable alternatives to implicit methods for solving parabolic PDE.

**Table 10** Accuracy and efficiency of a 2-node Krylov subspace spectral method (denoted by KSS(2)) and the MATLAB ODE solver ode15s, applied to the problem (7.15) with  $a(x) = a_s(x)$ . Error is the relative difference, in the 2-norm sense, between the exact solution  $u(x, t) = \cos(x - t)$  and the computed solution at  $T = 100$ .  $N$  denotes the number of grid points.  $\Delta t$  denotes the maximum time step permitted. Time is execution time in seconds

Method	$N$	$\Delta t$	Error	Time	Order
KSS(2)	64	1/32	5.0e-7	0.41	
	64	1/64	7.3e-8	0.76	2.76
	64	1/128	1.1e-8	1.46	
	64	1/256	1.6e-9	2.90	
	128	1/32	2.4e-7	0.78	
	128	1/64	3.4e-8	1.35	2.71
	128	1/128	5.0e-9	2.49	
	128	1/256	8.6e-10	4.88	
	256	1/32	1.0e-7	1.96	
	256	1/64	1.5e-8	3.16	2.59
ode15s	256	1/128	2.5e-9	5.47	
	256	1/256	4.6e-10	10.44	
	64	1/16	1.3e-7	0.80	
	64	1/32	4.3e-9	1.57	4.71
	64	1/64	1.9e-10	3.12	
	128	1/16	1.4e-7	1.03	
	128	1/32	4.4e-9	1.97	4.84
	128	1/64	1.7e-10	3.89	
	256	1/16	1.4e-7	1.58	
	256	1/32	4.5e-9	2.98	4.93
	256	1/64	1.5e-10	5.87	

It should be noted that the ideas presented in this paper apply directly to Krylov subspace spectral methods for wave propagation problems, as presented in [12], since they use exactly the same recursion coefficients. However, since most wave propagation problems are formulated as a first-order system of equations, rather than a scalar second-order equation, it is appropriate to extend Krylov subspace spectral methods to systems. This is a direction that is currently being explored.

In other ongoing research, to be detailed in [17], attempts are being made to reduce the overall operation count necessary to compute an accurate solution at a given time by a combination of reducing the number of time steps needed and improving the efficiency of each time step, using homogenizing transformations to precondition the problem.

The accuracy for discontinuous coefficients needs to be improved. To that end, we are considering the use of alternative differentiation operators that adjust the stencil in the presence of sharp gradients, such as those applied in CWENO schemes (see [18]). Early experimentation along these lines has yielded encouraging results. Another possibility is to use bases of trial functions other than trigonometric polynomials, such as orthogonal bases of compactly supported wavelets, or multiwavelet bases (see [1, 3]). Yet another direction worth investigating is the use of reprojection methods that reconstruct piecewise smooth functions from truncated Fourier series, for example see [7].

There are additional avenues worth exploring in order to improve efficiency even further. For example, by constructing Gauss-Kronrod rules using methods presented in [4], error estimates can be obtained, which can be used for step size control. Such step size control can be applied to individual Fourier components, thus allowing different components to be computed independently at different time scales. Furthermore, the computed solution can be differentiated *analytically* with respect to  $t$ , which provides a highly accurate residual that can be used for deferred correction, in the spirit of [14]. Preliminary results from this approach can be found in [15].

All of the ideas laid out in this section, in conjunction with those presented in this paper, are rapidly helping Krylov subspace spectral methods evolve to the point at which they are competitive with other, more established methods for solving a steadily growing variety of time-dependent problems.

## References

1. Alpert, B., Beylkin, G., Gines, D., Vozovoi, L.: Adaptive solution of partial differential equations in multiwavelet bases. *J. Comput. Phys.* **182**, 149–190 (2002)
2. Atkinson, K.: *An Introduction to Numerical Analysis*, 2nd edn. Wiley, New York (1989)
3. Beylkin, G., Keiser, J.M., Vozovoi, L.: A new class of time discretization schemes for the solution of nonlinear PDEs. *J. Comput. Phys.* **147**, 362–387 (1998)
4. Calvetti, D., Golub, G.H., Gragg, W.B., Reichel, L.: Computation of Gauss-Kronrod quadrature rules. *Math. Comput.* **69**, 1035–1052 (2000)
5. Dahlquist, G., Eisenstat, S.C., Golub, G.H.: Bounds for the error of linear systems of equations using the theory of moments. *J. Math. Anal. Appl.* **37**, 151–166 (1972)
6. Gautschi, W.: Orthogonal polynomials: applications and computation. *Acta Numerica* **5**, 45–120 (1996)
7. Gelb, A., Tanner, J.: Robust reprojection methods for the resolution of the Gibbs phenomenon. *Appl. Comput. Harmon. Anal.* **20**, 3–25 (2006)
8. Golub, G.H.: Some modified matrix eigenvalue problems. *SIAM Rev.* **15**, 318–334 (1973)
9. Golub, G.H.: Bounds for matrix moments. *Rocky Mt. J. Math.* **4**, 207–211 (1974)
10. Golub, G.H., Gutknecht, M.H.: Modified moments for indefinite weight functions. *Numer. Math.* **57**, 607–624 (1989)
11. Golub, G.H., Meurant, C.: Matrices, moments and quadrature. In: Griffiths, D.F., Watson, G.A. (eds.) *Proceedings of the 15th Dundee Conference, June–July 1993*. Longman Scientific & Technical (1994)
12. Guidotti, P., Lambers, J.V., Sølna, K.: Analysis of wave propagation in 1D inhomogeneous media. *Numer. Funct. Anal. Optim.* **27**, 25–55 (2006)
13. Hochbruck, M., Lubich, C.: On Krylov subspace approximations to the matrix exponential operator. *SIAM J. Numer. Anal.* **34**, 1911–1925 (1996)
14. Kress, W., Gustafsson, B.: Deferred correction methods for initial boundary value problems. *J. Sci. Comput.* **17**, 241–251 (2002)
15. Lambers, J.V.: *Krylov subspace methods for variable-coefficient initial-boundary value problems*. Ph.D. Thesis, Stanford University, SCCM Program (2003)
16. Lambers, J.V.: Krylov subspace spectral methods for variable-coefficient initial-boundary value problems. *Electron. Trans. Numer. Anal.* **20**, 212–234 (2005)
17. Lambers, J.V.: *Approximating eigenfunctions of variable-coefficient differential operators* (in preparation)
18. Levy, D., Puppo, G., Russo, G.: Central WENO schemes for hyperbolic systems of conservation laws. *M2AN* **33**(3), 547–571 (1999)
19. Sack, R.A., Donovan, A.F.: An algorithm for Gaussian quadrature given modified moments. *Numer. Math.* **18**, 465–478 (1971)
20. Shampine, L.F., Reichelt, M.W.: The MATLAB ODE suite. *SIAM J. Sci. Comput.* **18**, 1–22 (1997)