Krylov Subspace Spectral Methods with Coarse-Grid Residual Correction for Solving Time-Dependent, Variable Coefficient PDEs

Haley Dozier and James V. Lambers

Abstract Krylov Supspace Spectral (KSS) methods provide an efficient approach to the solution of time-dependent, variable coefficient partial differential equations by using an interpolating polynomial with frequency-dependent interpolation points to approximate a solution operator for each Fourier coefficient. KSS methods are high-order accurate time-stepping methods that also scale effectively to higher spatial resolution. In this paper, we will demonstrate the effectiveness of using coarse-grid residual correction, generalized to the time-dependent case, to improve the accuracy and efficiency of KSS methods. Numerical experiments demonstrate the effectiveness of this correction.

1 Introduction

Consider a time-dependent, variable-coefficient PDE, such as

$$u_t + Lu = 0, \quad 0 < x < 2\pi, \quad t > 0 \tag{1}$$

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

where L is a second order, self-adjoint, positive definite differential operator, such as a Sturm-Liouville operator. This type of problem often poses difficulties for both implicit and explicit time-stepping methods due to the lack of scalability of these methods caused by stiffness. That is, unless the chosen time-step is sufficiently small, the computed solutions might exhibit nonphysical behavior with large input sizes [7].

Haley Dozier
Department of Mathematics, The University of Southern Mississippi, 118 College Drive #5045, Hattiesburg, MS 39406 USA e-mail: Haley.Dozier@usm.edu

James V. Lambers
Department of Mathematics, The University of Southern Mississippi, 118 College Drive #5045, Hattiesburg, MS 39406 USA e-mail: James.Lambers@usm.edu

Krylov Subspace Spectral (KSS) methods [13] are designed specifically for solving time-dependent, variable-coefficient problems. The main idea behind KSS methods is to use an interpolating polynomial with frequency-dependent interpolation points to approximate the solution operator for each Fourier coefficient. As a result, KSS methods exhibit a high order of accuracy and stability. The dilemma is that this approach is only practical when applied to the high frequency components of the solution, and so a less efficient approach, such as standard Krylov projection, must be used on the low frequency components.

We have found that with the addition of coarse-grid residual correction, we can eliminate low frequency components of the error by restricting the problem to a coarser grid, and then using KSS methods on that coarser grid. In this paper, an overview of KSS methods in its current form will be given, as well as a description of how the addition of coarse-grid residual correction can be added to improve the accuracy of KSS. Numerical results will demonstrate this improvement.

The outline of this paper is as follows, Section 2 reviews KSS methods. Section 3 will discuss the addition of coarse-grid residual correction to KSS. Numerical results will be presented in Section 4, and conclusions will be given in Section 5.

2 Krylov Subspace Spectral Methods

We start by examining the parabolic PDE $u_t + Lu = 0$ on $(0, 2\pi)$ with periodic boundary conditions $u(0,t) = u(2\pi,t)$. The solution of this PDE can be represented by the Fourier series

$$u(x,t) = \frac{1}{\sqrt{2\pi}} \sum_{\omega=-\infty}^{\infty} e^{i\omega x} \hat{u}(\omega,t). \tag{3}$$

To find the Fourier coefficients of the solution of the solution at time t^{n+1}, we can represent them using the standard inner product on $(0, 2\pi)$,

$$\hat{u}(\omega,t_{n+1}) = \left\langle \frac{1}{\sqrt{2\pi}} e^{i\omega x}, e^{-L\triangle t} u(x,t_n) \right\rangle, \tag{4}$$

where $e^{-L\triangle t}$ is the exact solution operator.

The main idea behind KSS methods, as first described in [12], is to independently approximate all Fourier coefficients of the solution using an approximation of the exact solution operator that is tailored to each Fourier coefficient. To approximate the exact solution operator, spatially discretizing the right hand side of (4) leads to

$$\hat{u}(\omega,t_{n+1}) \approx \left(\frac{\Delta x}{\sqrt{2\pi}} e^{i\omega \mathbf{x}} \right)^H \left(e^{-L_N \Delta t} \mathbf{u}(x,t_n) \right) \tag{5}$$

where L_N is a $N \times N$ symmetric positive definite matrix obtained from spatial discretization of L using finite differences, and $\mathbf{x}$ is a vector of equally spaced

points in $[0,2\pi)$ with spacing $\Delta x = 2\pi/N$. If we let $\mathbf{u} = \frac{\Delta x}{\sqrt{2\pi}} e^{i\omega \mathbf{x}}$, $\mathbf{v} = u(\mathbf{x}, t_n)$, and $\phi(L_N) = e^{-L_N \Delta t}$, then we can represent the right side of (5) by the bilinear form

$$\mathbf{u}^H \phi(L_N) \mathbf{v}. \tag{6}$$

The matrix L_N is symmetric positive definite, and therefore has positive, real eigenvalues $b = \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_N = a$, and orthonormal eigenvectors $\mathbf{q}_j$ where $j = 1, \ldots, N$. Then the spectral decomposition of (6) is

$$\mathbf{u}^H \phi(L_N) \mathbf{v} = \sum_{j=1}^{N} \phi(\lambda_j) \mathbf{u}^H \mathbf{q}_j \mathbf{q}_j^H \mathbf{v}. \tag{7}$$

From here, we define the measure $\alpha(\lambda)$ by

$$\alpha(\lambda) = \begin{cases} 0 & \text{if } \lambda < a \\ \sum_{j=i}^{N} \mathbf{u}^H \mathbf{q}_j \mathbf{q}_j^H \mathbf{v} & \text{if } \lambda_i \leq \lambda \leq \lambda_{i-1} \\ \sum_{j=1}^{N} \mathbf{u}^H \mathbf{q}_j \mathbf{q}_j^H \mathbf{v} & \text{if } b \leq \lambda. \end{cases} \tag{8}$$

Now, as shown in [5], the bilinear form in (6) can be expressed as a Riemann-Stieltjes integral

$$\mathbf{u}^H \phi(L_N) \mathbf{v} = \int_a^b \phi(\lambda)\, d\alpha(\lambda). \tag{9}$$

To approximate this integral, Gaussian quadrature is used because it has a high degree of accuracy, and the weights are guaranteed to be positive if the measure $\alpha(\lambda)$ is positive and increasing [5]. After applying Gaussian quadrature to (9) the following approximation can be obtained

$$\int_a^b \phi(\lambda)\, d\alpha(\lambda) = \sum_{j=1}^{K} \phi(\lambda_j) w_j + error \tag{10}$$

where the nodes are λ_j and weights are w_j, for $j = 1, \ldots, K$. This quadrature rule is exact for polynomials of degree up to $2K-1$ [5].

In the case where $\mathbf{u} = \mathbf{v}$, the nodes and weights for Gaussian quadrature can be obtained using the symmetric Lanczos algorithm applied to L_N using initial vector $\mathbf{u}$. In the case where $\mathbf{u} \neq \mathbf{v}$, the weights for Gaussian quadrature, w_j, are not always guaranteed to be positive real numbers. This occurrence can destabilize the quadrature rule as shown in [1]. In this case, we consider a block approach:

$$[\mathbf{u}\ \mathbf{v}]^H \phi(L_N) [\mathbf{u}\ \mathbf{v}]. \tag{11}$$

We can represent this matrix as the Riemann-Stieltjes integral

$$\int_a^b \phi(\lambda)\, d\mu(\lambda) = \begin{bmatrix} \mathbf{u}^H \phi(L_N) \mathbf{u} & \mathbf{u}^H \phi(L_N) \mathbf{v} \\ \mathbf{v}^H \phi(L_N) \mathbf{u} & \mathbf{v}^H \phi(L_N) \mathbf{v} \end{bmatrix}, \tag{12}$$

where $\mu(\lambda)$ is a 2×2 matrix with entries of the form $\alpha(\lambda)$ from (8) [5]. Then a quadrature rule approximates the integral (12) as follows:

$$\int_a^b \phi(\lambda)\, d\mu(\lambda) \approx \sum_{j=1}^{2K} \phi(\lambda_j)\mathbf{v}_j\mathbf{v}_j^H + error. \tag{13}$$

where each λ_j is a scalar and each $\mathbf{v}_j$ is a 2-vector.

The block Lanczos algorithm applied to L_N using initial block $[\mathbf{u}\ \mathbf{v}]$ yields the nodes and weights for block Gaussian quadrature [6]. Specifically, block Lanczos produces the block tridiagonal matrix with 2×2 blocks

$$\mathscr{T}_K = \begin{bmatrix} M_1 & B_1^T & & & \\ B_1 & M_2 & B_2^T & & \\ & \ddots & \ddots & \ddots & \\ & & & B_{K-1} & M_K \end{bmatrix}, \tag{14}$$

where each B_j is upper triangular. The eigenvalues of $\mathscr{T}_K$ are used as the nodes λ_j in (13), and $\mathbf{v}_j\mathbf{v}_j^H$ are the matrix-valued "weights", where $\mathbf{v}_j$ consists of the first two components of the normalized eigenvector corresponding to λ_j.

A time step of block KSS, as seen in [12], proceeds as follows. First, we define

$$R_0(\omega) = [\hat{\mathbf{e}}_\omega\ \mathbf{u}^n] \tag{15}$$

where $\hat{\mathbf{e}}_\omega$ is a discretization of $\frac{\Delta x}{\sqrt{2\pi}}e^{i\omega\mathbf{x}}$ on a uniform N-point grid, and $\mathbf{u}^n$ is the computed solution at time t_n (these are $\mathbf{u}$ and $\mathbf{v}$ in (11) above). The QR factorization of (15) leads to

$$R_0(\omega) = X_1(\omega)B_0(\omega) \tag{16}$$

with

$$X_1(\omega) = \left[\hat{\mathbf{e}}_\omega\ \frac{\mathbf{u}_\omega^n}{||\mathbf{u}_\omega^n||^2}\right], \quad B_0(\omega) = \begin{bmatrix} 1 & \hat{\mathbf{e}}_\omega{}^H\mathbf{u}^n \\ 0 & ||\mathbf{u}_\omega^n||^2 \end{bmatrix},$$

where

$$\mathbf{u}_\omega^n = \mathbf{u}^n - \hat{\mathbf{e}}_\omega\hat{\mathbf{e}}^H\mathbf{u}^n = \mathbf{u}^n - \hat{\mathbf{e}}_\omega\hat{u}(\omega,t_n). \tag{17}$$

Block Lanczos is then applied to the discretized operator, L_N, with initial block $X_1(\omega)$. From block Lanczos, we obtain our M_j and B_j so that we can produce the matrix $\mathscr{T}_K(\omega)$ with the same form as (14), the entries of which depend on ω. Then, each Fourier coefficient of the solution at time t_{n+1} can be approximated by

$$[\hat{\mathbf{u}}^{n+1}]_\omega = [B_0^H(\omega)E_{12}^H e^{-\mathscr{T}_K(\omega)\triangle t}E_{12}B_0(\omega)]_{12}, \quad E_{12} = [\,\mathbf{e}_1\ \mathbf{e}_2\,] \tag{18}$$

By applying an inverse Fast Fourier Transform (FFT) to the vector of Fourier coefficients, we obtain the vector $\mathbf{u}^{n+1}$, which approximates the solution $u(x,t_{n+1})$. In [12] it was shown that this algorithm has local temporal accuracy of $O(\triangle t^{2K-1})$ for the parabolic problem and in [9] it was shown to have local temporal accuracy $O(\triangle t^{4K-2})$ for the second-order wave equation.

To improve the efficiency of block KSS methods, asymptotic analysis of block Lanczos iteration was performed in [3, 14]. It was shown that at high frequencies, the eigenvalue problem for $\mathscr{T}_K(\omega)$ approximately decouples, so that the Gaussian quadrature nodes could instead be estimated by performing "non-block" Lanczos on L_N with initial vectors $\hat{\mathbf{e}}_\omega$ and $\mathbf{u}^n$, which yields *frequency-dependent* and *frequency-independent nodes*, respectively.

This improves efficiency for two reasons. First, the frequency-independent nodes need only be computed once per time step, and shared by all quadrature rules of the form (13) for each ω. Second, the entries of the Jacobi matrix obtained by applying Lanczos with to L_N initial vector $\hat{\mathbf{e}}_\omega$ can easily be estimated in terms of the coefficients of the underlying differential operator L.

With these enhancements taken into account, the following algorithm from [3] describes a time step of KSS on $[t_n, t_{n+1}]$ to solve $u_t + Lu = 0$, on an N-point uniform grid, with periodic boundary conditions, and $O(\Delta t^{2K-1})$ accuracy in time.

1. Perform K iterations of Lanczos on L_N with initial vector $\mathbf{u}^n$ to obtain Jacobi matrix T_K. The eigenvalues $\lambda_1, \ldots, \lambda_K$ of T_K are the frequency-independent nodes.
2. For each $\omega = -N/2+1, \ldots, N/2$, compute the frequency-dependent nodes $\lambda_{1,\omega}, \ldots, \lambda_{K,\omega}$ from analytically computed estimates of the entries of $T_K(\omega)$, obtained through K iterations of Lanczos on L_N with initial vector $\hat{\mathbf{e}}_\omega$ [3, 14].
3. For each $\omega = -N/2+1, \ldots, N/2$, compute the polynomial interpolant

$$p_{2K-1,\omega}(\lambda) = \sum_{j=0}^{2K-1} c_{j,\omega} \lambda^j$$

   of $\phi(\lambda) = e^{-\lambda \Delta t}$, with interpolation points $\lambda_1, \ldots, \lambda_K, \lambda_{1,\omega}, \ldots, \lambda_{K,\omega}$.
4. Each Fourier coefficient of the solution at time t_{n+1} is then computed as follows:

$$[\mathbf{u}^{n+1}]_\omega = \sum_{j=0}^{2K-1} c_{j,\omega} \hat{\mathbf{e}}_\omega^H L^j \mathbf{u}^n. \tag{19}$$

   FFTs are used to compute Fourier coefficients of $L^j \mathbf{u}^n$, but by performing Newton interpolation in step 3 with the frequency-independent nodes listed first, the number of FFTs can be reduced from $2K$ in (19) to K [3].
5. Perform an inverse FFT to obtain the solution at time t_{n+1}.

3 KSS with Coarse-Grid Residual Correction

The "smoothing" property [2] is a phenomenon that many iterative methods possess. This property describes a method in which the rapid decrease in error in earlier iterations is due to the elimination of higher frequency error. Methods with the smoothing property are often not as effective at eliminating low frequency error.

Similarly, due to the work of Cibotarica, Lambers, and Palchak in [3, 14], KSS methods are already highly effective at computing the high frequency components of the solution, but since the asymptotic analysis in these works applied only to high-frequency components, they are not as effective at eliminating low frequency error.

Solving a PDE on a coarse grid is an effective way to eliminate low-frequency error in linear systems that arise from the spatial discretization of elliptic partial differential equations. To apply this multigrid-inspired technique to a time-dependent problem of the form $u_t + Lu = 0$, we first define the residual as $R = u_t + Lu$, and then solve a non-homogeneous version of the PDE to estimate the error for correction.

We therefore need three functions to implement coarse grid residual correction generalized to the time-dependent PDE:

- a function to restrict the problem to a coarser grid
- a function to discretize the spatial differential operator on the coarse grid, and
- a function to interpolate back to the fine grid

The Krylov Subspace Spectral method with Coarse-Grid Residual Correction (KSS-CG) proceeds as follows, during *each* time step:

1. use KSS as described in Section 2 to compute an initial solution and residual,
2. use a FFT to restrict the residual to a coarse grid,
3. compute a correction on the coarse grid by solving the same PDE, but with the residual as a source term,
4. use a FFT to transfer the correction to the fine grid, and
5. add the correction to the initial solution from step 1.

It should be noted that since the test cases we use have homogeneous Neumann boundary conditions instead of periodic, we will be using a discrete cosine transform that employs FFTs.

In [4] it is shown why multigrid methods are ineffective for nonelliptic problems such as the Helmholtz equation, as any choice of a relaxation parameter results in an amplification of some modes. In [10], KSS methods were applied to the Helmholtz equation, and difficulties arose due to a singularity in the integrand $\phi(\lambda) = 1/\lambda$ in (9) resulting from the indefiniteness of the underlying matrix.

In both cases, the PDE is being solved on the entire (spatial) domain through the solution of a single system of linear equations; by contrast, KSS-CG is not solving a nonelliptic PDE on the entire (space-time) domain simultaneously. As described above, the algorithm is essentially a time-stepping method, with residual correction performed after each time step on a coarser spatial grid.

3.1 Using KSS-CG to solve a Parabolic PDE

Consider the parabolic PDE

$$u_t + Lu = 0, \quad (0, 2\pi) \times (0, \infty), \tag{20}$$

$$u(x,0) = f(x), \quad 0 < x < 2\pi, \tag{21}$$

with either periodic or homogeneous boundary conditions. In this section, we restrict ourselves to one space dimension for concreteness; a 2-D problem is considered in the results section.

As previously stated, multigrid can be used to improve the accuracy of iterative methods that have the smoothing property. After KSS is applied during a single time step as described in Section 2, we are left with a relatively smooth error. To perform residual correction, first the solution computed from KSS is used to find the residual, $R(x,t) = u_t(x,t) + Lu(x,t)$. This entails using the time derivative of the solution operator, in this case $S(t) = e^{-Lt}$, to compute u_t. That is, the same Gaussian quadrature rules are used as with computing the solution itself, but with the integrand $f(\lambda) = -\lambda e^{-\lambda t}$.

To restrict the residual to a coarse grid, the low-frequency components of its discrete Fourier transform are extracted. Once the residual is restricted to the coarse grid, the differential operator L must also be restricted to the coarse grid. Then, the non-homogeneous equation

$$e_t + Le = R(x,t) \tag{22}$$

must be solved where e is the error, and the initial condition is $e(x,0) = e_0 = 0$. It follows that

$$e(x,t) = \int_0^t e^{-L(t-s)} R(x,s)\, ds. \tag{23}$$

If we use Gaussian quadrature to approximate the integral in (23), we obtain the error estimate

$$e(x,\Delta t) = \int_0^{\Delta t} e^{-L(\Delta t - s)} R(x,s) ds \approx \sum_{k=1}^{m} w_k e^{-L(\Delta t - s_k)} R(x,s_k) \tag{24}$$

where the s_k are the Gauss-Legendre points, transformed to the interval $[0,\Delta t]$, and the w_k are the weights transformed to the same interval. To correct the solution, the newly obtained error estimate can be interpolated back to the fine grid by padding its discrete Fourier transform with zeros.

A straightforward modification of the above algorithm to perform multiple coarse-grid corrections would have the drawback that with each correction, the total number of quadrature nodes in time would increase substantially, because the residual of each term in each correction would have to be evaluated at m times, where m is the number of nodes used in the quadrature rule in (24). To avoid the resulting increase in computational expense, future work will focus on coarsening in both space and time to make multiple corrections practical.

3.2 Using KSS-CG to solve a Hyperbolic Problem

Consider the hyperbolic PDE

$$u_{tt} = Lu, \quad (0,2\pi)\times(0,\infty), \tag{25}$$

$$u(x,0) = f(x), \quad u_t(x,0) = g(x), \quad 0 < x < 2\pi. \tag{26}$$

The solution operator for this problem can be expressed as a matrix of functions of the operator L:

$$\begin{bmatrix} u(x,t+\Delta t) \\ u_t(x,t+\Delta t) \end{bmatrix} = \begin{bmatrix} \cos(\sqrt{-L}\triangle t) & \frac{1}{\sqrt{-L}}\sin(\sqrt{-L}\triangle t) \\ -\sqrt{-L}\sin(\sqrt{-L}\triangle t) & \cos(\sqrt{-L}\triangle t) \end{bmatrix} \begin{bmatrix} u(x,t) \\ u_t(x,t) \end{bmatrix}. \tag{27}$$

The entries of the propagator matrix in (27) indicate which functions are the integrands in the Riemann-Stieltjes integrals that are used to compute the Fourier coefficients of the solution [11].

The residual, R, computed at each time step is

$$R = u_{tt} - Lu,$$

where the second time derivative of the solution operator from Section 2 is used to compute u_{tt}. That is, the second derivatives of the matrix functions in (27) with respect to Δt are used as integrands in the required Riemann-Stieltjes integrals. Then, the error used to update the solution is obtained by solving

$$e_{tt} = Le + R(x,t)$$

which yields

$$e(x,t_{n+1}) = \int_0^{\Delta t} \frac{1}{\sqrt{-L}}\sin(\sqrt{-L}(\triangle t - s))R(x,s)\,ds. \tag{28}$$

This error estimate and its time derivative are then interpolated back to the fine grid by padding their discrete Fourier transforms with zeros, as in the parabolic case.

4 Numerical Results

In this section, the effectiveness of KSS with coarse grid residual correction (KSS-CG) will be demonstrated. The following approaches will be compared:

- KSS method as described in Section 2.
- KSS method with with coarse grid residual correction, as described in Section 3, using 2 Gaussian quadrature nodes (KSS-CG2). This will only be done in the parabolic case.
- KSS method with with coarse grid residual correction, as described in Section 3, using 4 Gaussian quadrature nodes (KSS-CG5). This will only be done in the hyperbolic case.

- KSS method with with coarse grid residual correction using 3 Gaussian quadrature nodes (KSS-CG3)
- Krylov projection as described in [8] (KP)
- KSS-EPI method as described in [3].

The errors reported are relative errors with respect to an "exact solution" using the MATLAB ODE solver `ode15s`, computed using the smallest allowable time step. For each test problem we use grid sizes of $N = 50, 150$ grid points per dimension to demonstrate how increased spatial resolution will affect the performance of each method. For KSS-CG, a grid with $N = 25$ grid points per dimension is used for residual correction.

4.1 Parabolic Problem

We first compare all five methods when solving the 2-D parabolic problem

$$u_t = \alpha \triangle u + (1 - 3u_0^2)u, \tag{29}$$

on the rectangle $[0,1]^2$ and for $0 < t < 0.2$, with initial condition

$$u(x,y,0) = u_0(x,y) = 0.4 + 0.1\cos(2\pi x)\cos(5\pi y) \tag{30}$$

and homogeneous Neumann boundary conditions.

Figures 1 and 2 show the error vs. time performance and error vs. time step performance for each approach used, with grid sizes $N = 50$ and 150 points per dimension, respectively. From these plots we can see that the errors for both the KSS-CG methods are smaller that the errors for any other method. On the larger grid size, seen in Figure 2, the difference in efficiency between each method is more pronounced. KSS-CG3 had the smallest relative error for each time-step, followed closely by KSS-CG2. Standard KSS and KSS-EPI methods demonstrated less computational time per time-step although comparatively the percentage increase in computational time between grid sizes for KSS is much larger than the percentage increase for both KSS-CG methods. This implies that for even larger grid sizes, KSS-CG may be more efficient than the other tested methods, as also observed in [3], though further numerical experiments would have to be performed to validate this theory. It is also important to note that although all KSS methods used are third-order accurate, KSS-CG achieved fourth-order accuracy.

4.2 Hyperbolic Problem

We now compare the performance of these methods when solving the hyperbolic problem

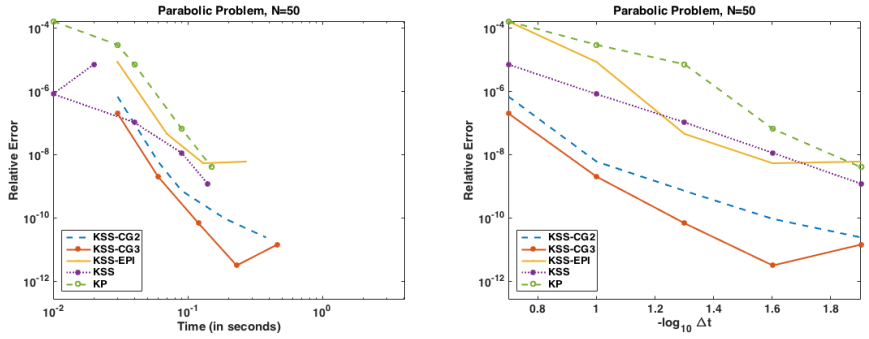

Fig. 1 *Time of each timestep for each method vs error with grid sizes $N = 50$ points per dimension. The blue dashed curve represents KSS-CG using 2 Gaussian quadrature nodes, the red solid and star curve represents KSS-CG using 3 Gaussian quadrature nodes, the yellow dash-dot curve represents KSS-EPI, the purple dot-star curve represents the KSS method with filtering (but without correction), and the green dash-circle curve represent standard Krylov Projection.*

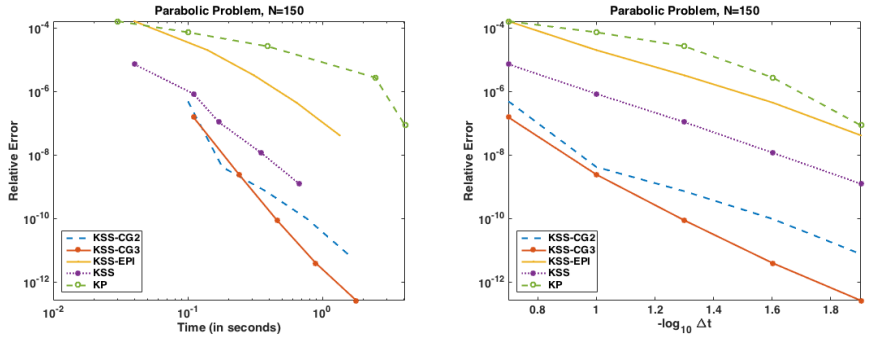

Fig. 2 *Time of each timestep for each method vs error with grid sizes $N = 150$ points per dimension. The blue dashed curve represents KSS-CG using 2 Gaussian quadrature nodes, the red solid and star curve represents KSS-CG using 3 Gaussian quadrature nodes, the yellow dash-dot curve represents KSS-EPI, the purple dot-star curve represents the KSS method with filtering (but without correction), and the green dash-circle curve represent standard Krylov Projection.*

$$u_{tt} = \alpha \triangle u + (1 - 3u_0^2)u, \tag{31}$$

on the rectangle $[0,1]^2$ and for $0 < t < 2$, with initial conditions

$$u(x,y,0) = u_0(x,y), \quad u_t(x,y,0) = 1,$$

where $u_0(x,y)$ is as defined in (30). For this problem, all KSS methods used are sixth-order accurate, and therefore we use KSS-CG with 3 and 4 Gaussian quadrature nodes for the correction.

We can see from the graph of the smaller grid size in Figure 3 that all methods yield similar computational time in the last time-step, yet KSS-CG3 and KSS-CG4

yield substantially higher accuracy. As the grid size increases from $N = 50$ grid points per dimension to $N = 150$, the accuracy of Krylov projection and KSS-EPI decreased while the accuracy of KSS and both KSS-CG methods increased, as can be seen in Figure 4. Most significantly, the KSS-CG methods are far more efficient and scalable than Krylov Projection or KSS-EPI. As in the parabolic case, both KSS-CG methods exhibited higher-order accuracy: eighth-order instead of sixth over the smaller time steps.

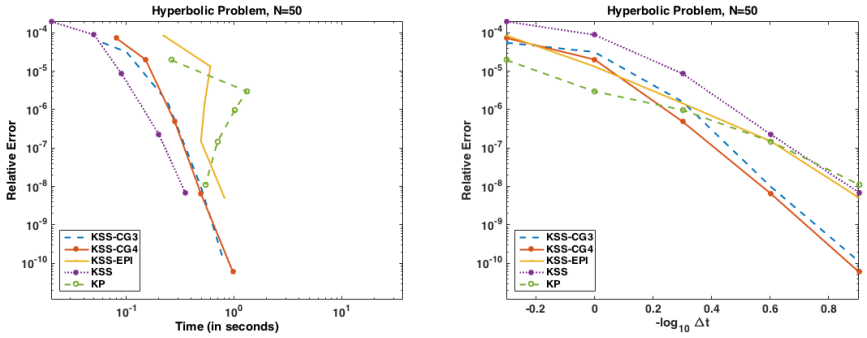

Fig. 3 *Time of each timestep for each method vs error with grid size $N = 50$ points per dimension. The blue dashed curve represents KSS-CG using 3 Gaussian quadrature nodes, the red solid and star curve represents KSS-CG using 5 Gaussian quadrature nodes, the yellow dash-dot curve represents KSS-EPI, the purple dot-star curve represents the KSS method with filtering (but without correction), and the green dash-circle curve represent standard Krylov Projection.*

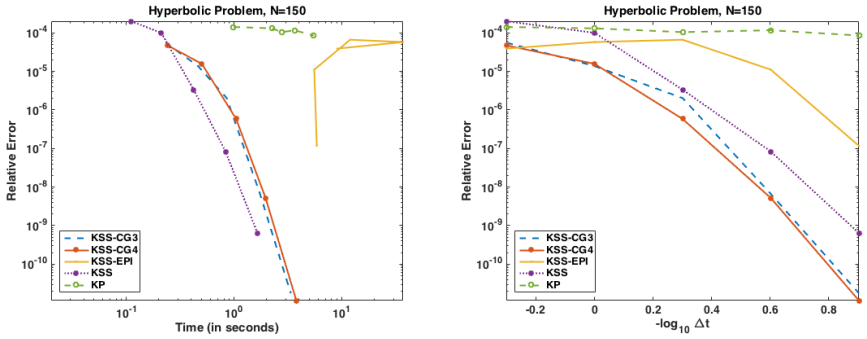

Fig. 4 *Time of each timestep for each method vs error with grid size $N = 150$ points per dimension. The blue dashed curve represents KSS-CG using 3 Gaussian quadrature nodes, the red solid and star curve represents KSS-CG using 5 Gaussian quadrature nodes, the yellow dash-dot curve represents KSS-EPI, the purple dot-star curve represents the KSS method with filtering (but without correction), and the green dash-circle curve represent standard Krylov Projection.*

5 Conclusion

It has been demonstrated that with coarse-grid residual correction, KSS methods become more accurate when solving time-dependent variable-coefficient PDEs, and also achieve a higher order of temporal accuracy. This represents a significant step forward in the evolution of KSS methods, as previous versions used the less efficient approaches of explicitly performing block Lanczos [14] or Krylov projection [3] to compute low-frequency components. Further optimization of KSS-CG will be needed to obtain faster computation time with sustained accuracy.

Future work on the combination of coarse-grid residual correction and KSS is needed to fully explore the effectiveness of this method. Topics for further research include generalizing KSS-CG to solve a wider variety of problems, including nonlinear PDEs in combination with EPI methods as in [3]. Future work must also focus on further grid coarsening, as in this paper only the next coarsest grid was used. An efficient way to use any number of corrections must be developed to fully realize the potential of KSS-CG; this would involve coarsening in time as well as space.

References

1. K. Atkinson. *An Introduction to Numerical Analysis* . 2nd ed., Wiley, (1989)
2. W. Briggs, V. E. Henson, and S. F. McCormick. *A Multigrid Tutorial*. Society for Industrial and Applied Mathematics, Philidelphia, PA 19104 (2000)
3. A. Cibotarica, J. V. Lambers, and E. M. Palchak. Solution of nonlinear time-dependent pde through componentwise approximation of matrix functions. *Journal of Computational Physics*, (2016)
4. O. G. Ernst and M. J. Gander. Why it is Difficult to Solve Helmholtz Problems with Classical Iterative Methods. *Lecture Notes in Computational Science and Engineering: Numerical Analysis of Multiscale Problems* 83:325–363 (2011)
5. G. H. Golub and G. Meurant. *Matrices, Moments and Quadrature with Applications*. Princeton University Press (2010)
6. G.H. Golub and R. Underwood. The block lanczos method for computing eigenvalues. *Mathematical Software III*, pages 361–377 (1977)
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. Approximations to the matrix exponential operator. *SIAM Journal of Numerical Analysis*. 34:1911–1925 (1996)
9. J. V. Lambers. An Explicit, Stable, High-Order Spectral Method for the Wave Equation Based on Block Gaussian Quadrature. IAENG Journal of Applied Mathematics 38:233–248 (2008)
10. J. V. Lambers. A Multigrid Block Krylov Subspace Spectral Method for Variable-Coefficient Elliptic PDE. *IAENG Journal of Applied Mathematics*, 39:236–246 (2009)
11. J. V. Lambers. A spectral time-domain method for computational electrodynamics. *Adv. Appl. Math. Mech.*, 1:781–798 (2009)
12. J. V. Lambers. Enhancement of krylov subspace spectral methods by block lanczos iteration. *Electronic Transactions on Numerical Analysis*, 31:86–109 (2008)
13. J. V. Lambers, "Krylov Subspace Methods for Variable-Coefficient Initial-Boundary Value Problems", Ph.D. thesis, SC/CM Program, Stanford University (2003)
14. E. M. Palchak, A. Cibotarica, and J. V. Lambers. Solution of time-dependent pde through rapid estimation of block gaussian quadrature nodes. *Linear Algebra and its Applications*, 468:233–259 (2015)