Solution of nonlinear time-dependent PDEs through componentwise approximation of matrix functions

Alexandru Cibotarica, James V. Lambers, Elisabeth M. Palchak

Received 1 November 2015
Received in revised form 13 June 2016
Accepted 15 June 2016
Available online 23 June 2016

Keywords:
Exponential propagation iterative methods
Krylov subspace spectral methods
Stiff systems
Gaussian quadrature
Lanczos iteration

1. Introduction

Consider an autonomous, stiff system of ODEs of the form

$$\mathbf{y}' = F(\mathbf{y}), \quad \mathbf{y}(t_0) = \mathbf{y}_0,$$

such as one that would arise from spatial discretization of a PDE. Stiffness poses problems for both explicit and implicit time-stepping methods. For explicit methods, the time step is severely restricted, while for implicit methods, an ill-conditioned system must be solved during each time step, for which an iterative method requires many iterations or a specially developed preconditioner [27].

Exponential propagation iterative (EPI) methods, due to Tokman et al. [22,27] are designed to reduce the number of Krylov projection steps needed to compute any required matrix function–vector products of the form $\mathbf{w} = \varphi(\mathbf{A} \tau) \mathbf{b}$, where $\varphi$ is a function, $\mathbf{A}$ is an ill-conditioned matrix, $\tau$ is a parameter determined by the time step, and $\mathbf{b}$ is a vector. One approach to computing $\mathbf{w}$ for a given symmetric matrix $\mathbf{A}$ is to apply the Lanczos algorithm to $\mathbf{A}$ with the initial vector $\mathbf{b}$, until we obtain a matrix $X_j$ with orthonormal columns and a tridiagonal matrix $T_j$ such that $X_j^T \mathbf{A} X_j = T_j$. Then, we can compute the approximation
\[ w_j = |b|_2 x_j \varphi(T_j \tau) e_1, \]

where \( e_1 = [1, 0, \ldots, 0]^T \). Since each column \( x_j \) of the matrix \( X_j \) is of the form \( x_k = p_{k-1}(A)b \), where \( p_{k-1}(A) \) is a polynomial of degree \( n \) in \( A \), \( w_j \) is the product of a polynomial in \( A \) of degree \( j - 1 \) and \( b \). Since the matrix \( A \) arises from a stiff PDE, the eigenvalues of \( A \) are not clustered, which means that a large number of Lanczos iterations might be required in order to obtain a good approximation of \( w \).

The difficulty that time-stepping methods have with stiffness is due to the coupling of the components of the solution with widely varying frequencies. Another problem is the use of the same function, whether polynomial or rational, to approximate all of these components of \( \varphi(A \tau)b \), when such a function cannot effectively approximate \( \varphi(\lambda \tau) \) on a large interval except at high degree, which is computationally expensive. A solution to this problem is to use Krylov subspace spectral (KSS) methods [16,18], which is a component-wise approach in which each Fourier coefficient of the solution is computed using an approximation of the solution operator that is tailored to that component. Specifically, KSS methods use an interpolating polynomial with frequency-dependent interpolation points to approximate the function \( \varphi \). As a result, these methods demonstrate a high order of accuracy and stability like that of implicit methods.

Until now, KSS methods have been used mainly in linear PDEs on \( d \)-dimensional boxes, for \( d = 1, 2, 3 \), with either periodic or homogeneous boundary conditions. A successful implementation of KSS methods for nonlinear PDEs was used by Guidotti et al. when a one-node KSS method was applied to nonlinear diffusion equations from image processing [11] to obtain first-order accuracy in time. However, in order to achieve higher-order accuracy for nonlinear PDEs, in addition to using more nodes, it is also necessary to account for the nonlinearity more carefully than with a simple linearization at each time step. This can be accomplished by combining KSS methods with EPI methods.

In this paper, such a combination is presented, for the purpose of solving systems of ODEs of the form (1) that are obtained through spatial discretization of nonlinear PDEs, or systems of nonlinear PDEs, defined on rectangular domains with periodic, homogeneous Dirichlet, or homogeneous Neumann boundary conditions. The proposed method includes the following features that distinguish it from previous work on EPI or KSS methods:

- Instead of applying a Krylov projection method (e.g., see [13–15]) for computing approximations of expressions of the form \( y = \varphi(A \tau)b \), where \( A \) is an \( N \times N \) matrix, \( b \) is an \( N \)-vector, \( \tau \) is a scaling factor derived from the time step, and \( \varphi \) is a smooth function, such a method is applied only to a low-frequency approximation of \( b \), in order to avoid the larger number of iterations that these methods typically incur at higher spatial resolution. Furthermore, for advection-dominated problems, denoising is applied to the Krylov subspace basis produced by this iteration, to remove the obstacle to convergence that is presented by spurious high-frequency oscillations that occur in the basis vectors.

- For the high-frequency portion of the vector \( b \), application of \( \varphi(\tau A) \) is performed using a KSS method, as described in [23]. In this particular KSS method, each Fourier component of the output vector \( y \) is approximated using its own block Gaussian quadrature rule, except that the quadrature nodes are obtained through high-frequency analysis of block Lanczos iteration, which yields formulas for approximation of the nodes.

This approach differs from KSS methods from [16,18], in which block Lanczos iteration is performed for each Fourier component and the resulting block tridiagonal matrices are diagonalized to obtain the Gaussian quadrature nodes and weights, and from KSS methods from [17], in which asymptotic analysis is used to approximate only the extremal nodes, while the interior nodes are prescribed using equal spacing. The benefit of the approach used in [23] is that it combines the accuracy of the approach of [16,18] with the efficiency of the approach of [17].

In [23], formulas for the nodes were given for a 1-D, self-adjoint second-order operator with periodic boundary conditions. In this paper, a similar analysis is applied to other operators, that illustrate generalizations to other boundary conditions, higher spatial dimension, non-self-adjoint operators, and systems of coupled equations. This paper also generalizes the work of [23] from linear to nonlinear PDEs.

- Once the nodes are determined as described above, it is necessary to construct and apply frequency-dependent interpolating polynomials of the matrix \( A \) to the high-frequency portion of the vector \( b \). This paper provides implementation details for this task, and explains how it can be accomplished using approximately half of the number of Fourier transforms that a straightforward implementation would require.

The outline of the paper is as follows. Section 2 gives an overview of KSS methods. Section 3 reviews their acceleration based on asymptotic analysis of recursion coefficients, first presented in [23], and extends the analysis to new cases. Section 4 provides a brief overview of EPI methods, and demonstrates the spurious high-frequency oscillations that can occur when using standard Krylov projection within an EPI method. Section 5 describes how KSS and EPI methods are combined. Numerical results are presented in Section 6, and conclusions are stated in Section 7.

2. Krylov subspace spectral methods

To review the essential aspects of KSS methods, as first described in [16], we consider the parabolic PDE \( u_t + Lu = 0 \) on the interval [0, 2\( \pi \)], where \( L \) is a Sturm–Liouville operator, with appropriate initial conditions and periodic boundary conditions. The idea behind KSS methods is that the Fourier coefficients of the solution \( \tilde{u}(x, t_{n+1}) \) are obtained by applying the exact solution operator to the previously computed solution \( \tilde{u}(x, t_n) \). These Fourier coefficients are given by
We consider quadrature constant and the algorithm for discretizing a 1-D, second-order differential operator with smooth leading coefficient (top plot) and discontinuous leading coefficient (bottom plot), where $\mathbf{u} = \mathbf{v}$ is a discretization of $e^{2i\omega}$ (solid curve) or $e^{4i\omega}$ (dashed curve).

$$
\tilde{u}(\omega, t_{n+1}) = \left\{ \frac{1}{\sqrt{2\pi}} e^{i\omega x} e^{-L\Delta t} \tilde{u}(x, t_n) \right\}.
$$

(3)

where $\langle \cdot, \cdot \rangle$ denotes the standard inner product on $[0, 2\pi]$ and $e^{-L\Delta t}$ is the solution operator of the PDE.

As a result of the spatial discretization of (3), we obtain the following bilinear form

$$
\mathbf{u}^T \varphi(A) \mathbf{v},
$$

(4)

where $\mathbf{u} = \frac{1}{\sqrt{2\pi}} e^{i\omega x}$ and $\mathbf{v} = \tilde{u}(x, t_n)$ are $N$-vectors, $A = LN$, where $LN$ is a spectral discretization of $L$, and $\varphi(\lambda) = e^{-\lambda \Delta t}$.

Since the matrix $A$ is symmetric positive definite, it has real eigenvalues $b = \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N = a > 0$, and corresponding orthonormal eigenvectors $\mathbf{q}_j$, $j = 1, \ldots, N$. As a result, we have the following spectral decomposition of (3):

$$
\mathbf{u}^T \varphi(A) \mathbf{v} = \sum_{j=1}^{N} \varphi(\lambda_j) \mathbf{u}^T \mathbf{q}_j \mathbf{q}_j^T \mathbf{v}.
$$

(5)

As mentioned by Golub and Meurant in [6], (3) can also be viewed as a Riemann–Stieltjes integral

$$
\mathbf{u}^T \varphi(A) \mathbf{v} = \int_a^b \varphi(\lambda) d \alpha(\lambda),
$$

(6)

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

$$
\alpha(\lambda) = \begin{cases} 
0, & \text{if } \lambda < a \\
\sum_{j=1}^{N} u_j v_j, & \text{if } \lambda_i \leq \lambda < \lambda_{i-1}, \\
\sum_{j=1}^{N} u_j v_j, & \text{if } b \leq \lambda.
\end{cases}
$$

This allows approximation using Gaussian quadrature rules, where the nodes and weights are obtained using the Lanczos algorithm applied to $A$ with initial vectors $\mathbf{u}$ and $\mathbf{v}$ [6].

Fig. 1 demonstrates why integrals of the form (5) can be approximated accurately with a small number of nodes in the case where $A$ is a discretization of a differential operator and the vector $\mathbf{u}$ is used to extract a particular Fourier coefficient of $f(A)\mathbf{v}$. We examine the distribution $d\alpha(\lambda)$ in the case where $\mathbf{u} = \mathbf{v} = e^{i\omega x}$ for small and large values of $\omega$, and for $A$ discretizing a differential operator of the form $-\partial_x a(x) \partial_x$, with $a(x) > 0$ being a smooth function or a piecewise constant function. In either case, $d\alpha(\lambda)$ is mostly concentrated within a portion of the interval of integration $[a, b]$. Gaussian quadrature rules for such integrals naturally target these relevant portions [16,17].

In the case where $\mathbf{u} \neq \mathbf{v}$, the presence of a negative weight would destabilize the quadrature rule [1]. Alternatively, we consider the approximation of the $2 \times 2$ matrix integral

$$
\begin{bmatrix} \mathbf{u} & \mathbf{v} \end{bmatrix}^T \varphi(A) \begin{bmatrix} \mathbf{u} & \mathbf{v} \end{bmatrix}.
$$

(7)

We use the most general $K$-node quadrature formula, as described in [6], to get an approximation for (6) of the form

$$
\int_a^b \varphi(\lambda) d\mu(\lambda) = \sum_{j=1}^{2K} \varphi(\lambda_j) \mathbf{v}_j \mathbf{v}_j^T + \text{error},
$$

(8)

where, for each $j$, $\lambda_j$ is a scalar and $\mathbf{v}_j$ is a 2-vector. Each node $\lambda_j$ is an eigenvalue of the matrix.
\[
T_K = \begin{bmatrix}
M_1 & B_1^T & 0 \\
B_1 & M_2 & B_2^T \\
& \ddots & \ddots \\
& & B_{K-1} & M_K \\
\end{bmatrix},
\]

which is a block-tridiagonal matrix of order \(2K\). The vector \(v_j\) consists of the first two elements of the corresponding normalized eigenvector. The matrices \(M_j\) and \(B_j\) are computed using the block Lanczos algorithm [8].

The block KSS method starts by defining
\[
R_0(\omega) = \left[ \hat{e}_{\omega} \ u^0 \right],
\]
where \(\hat{e}_{\omega}\) is a discretization of \(\frac{1}{\Delta t} e^{i \omega t}\) and \(u^0\) is the computed solution at time \(t_n\). The QR factorization of \(R_0(\omega)\) yields \(R_q(\omega) = X_1(\omega)B_1(\omega)\). Then, block Lanczos iteration is applied to the discretized operator \(L_N\) with initial block \(X_1(\omega)\), producing a block tridiagonal matrix \(T_K(\omega)\) of the form (9), where each entry is a function of \(\omega\). Then, each Fourier coefficient of the solution at time \(t_{n+1}\) can be expressed as
\[
[\hat{u}^{n+1}]_\omega = \left[ B_0^T E_{12} e^{-\Delta t K(\omega) \Delta t} E_{12} B_0 \right]_{12}, \quad E_{12} = \left[ \hat{e}_1 \ \hat{e}_2 \right].
\]

This algorithm has temporal accuracy \(O(\Delta t^{2K-1})\) for parabolic problems [16]. Even higher-order accuracy, \(O(\Delta t^{4K-2})\), is obtained for the second-order wave equation [18]. Furthermore, under appropriate assumptions on the coefficients of the PDE, the 1-node KSS method is unconditionally stable [16,18]. More generally, the temporal order of accuracy is \(O(\Delta t^{2K-1})\), where \(d\) is the highest order of a time derivative in the PDE; this order of accuracy has been observed with the Schrödinger equation [20] and Maxwell’s equations [19].

The choice to compute coefficients of the solution in the basis \(e^{i \omega t}, \omega \in \mathbb{Z}\) is due to the periodic boundary conditions imposed on the sample problem used in this section. As will be seen in the next section, the choice of basis is determined by the domain and boundary conditions. While an orthogonal basis is most convenient, orthogonality is not essential; what is essential is efficient transformation between physical and frequency space.

3. Asymptotic analysis of block Lanczos iteration

Let \(u\) be a discretization of the solution \(u\) on a uniform \(N\)-point grid. Then, KSS methods use the initial block \(R_0 = \left[ \hat{e}_{\omega} \ u \right]\), for each \(\omega = -N/2 + 1, \ldots, N/2\). We then use block Lanczos iteration to compute the block tridiagonal matrix \(T_K\) in (9). In [23] it is shown for the PDE \(u_t + Lu = 0\), with \(L = -(p(x)u_x)x + q(x)u\) and periodic boundary conditions, that every (nonzero) off-diagonal entry of \(M_j\) or \(B_j\), for \(j = 1, 2, \ldots\), converges to zero as \(|\omega| \to \infty\), given sufficient regularity of the solution and coefficients of \(L\).

It follows that in this high-frequency limit, the block tridiagonal matrix \(T_K\) produced by block Lanczos applied to \(R_0\) converges to the matrix that would be obtained by applying “non-block” Lanczos iteration to the two columns of \(R_0\) separately, and then alternating rows and columns of the tridiagonal matrices produced by these iterations. That is, the eigenvalue problem for this matrix decouples. For finite \(\omega\), the block Gaussian quadrature nodes can at least be estimated by computing the eigenvalues of these smaller, tridiagonal matrices [23].

More precisely, we can obtain approximations of half of the block Gaussian quadrature nodes for all Fourier components by applying “non-block” Lanczos iteration to the matrix \(L_N\) with initial vector \(u\), the computed solution, as is done in Krylov projection methods such as those described in [13–15]. These nodes will be referred to as frequency-independent nodes. Because this iteration does not depend on \(\omega\), the frequency-independent nodes need only be computed once for each vector \(u\) for which an expression of the form \(\phi(L_N \tau)u\) is required. To estimate the other half of the nodes, we can perform an asymptotic analysis of Lanczos iteration applied to \(L_N\) with initial vector \(\hat{e}_\omega\) [23]; these are called frequency-dependent nodes.

As an example, we consider the case where the matrix \(A\) comes from a spectral discretization of the operator \(Lu = -pu_{xx} + q(x)u\) on \([0, 2\pi]\), where \(p\) is a constant, and assuming periodic boundary conditions [23]. Carrying out three iterations, which corresponds to a fifth-order accurate KSS method for a parabolic PDE, yields the recursion coefficients as functions of the wave number \(\omega\), after neglecting lower-order terms:
\[
\begin{bmatrix}
\alpha_1 & 0 & 0 \\
0 & \alpha_2 & \sqrt{\Delta x/2\pi} \\
0 & \sqrt{\Delta x/(2\pi)} & 0 \\
\end{bmatrix}
\approx
\begin{bmatrix}
p\omega^2 & \sqrt{\Delta x/(2\pi)}\|q\|_2 & 0 \\
\sqrt{\Delta x/(2\pi)}\|q\|_2 & 2p|\omega|\|q_x\|_2/\|q\|_2 & 0 \\
0 & 2p|\omega|\|q_x\|_2/\|q\|_2 & p\omega^2 \\
\end{bmatrix},
\]

where \(\Delta x = 2\pi/N\), with \(N\) the number of grid points, and \(q = q(x)\), where \(x\) is a \(N\)-vector of equally spaced grid points in \([0, 2\pi]\) with spacing \(\Delta x\). We also define \(\bar{q}(x) = q(x) - \bar{q}\), where \(\bar{q}\) is the average value of \(q(x)\) on \([0, 2\pi]\), and let \(\bar{q} = \bar{q}(x)\). The vector \(q_x\) is the spatial discretization of \(q_x\).

It follows that the nodes can easily be estimated as
\[
\lambda_{1,\omega} = p\omega^2, \quad \lambda_{2,\omega} = p\omega^2 \pm \sqrt{\beta_1^2 + \beta_2^2}, \quad i = 2, 3.
\]
When the matrix $A$ is a finite-difference representation of the underlying differential operator, the block Gaussian quadrature nodes can be represented more accurately if formulas for the eigenvalues of symmetric Toeplitz matrices are used for the leading-order terms in the nodes. For example, in (12), $p\omega^2$ is replaced by $2p(N/\pi)^2(1 - \cos(\pi \omega/N))$, where $N$ is the number of grid points.

3.1. Asymptotic analysis for self-adjoint operators

In this subsection, we illustrate the process of estimating frequency-dependent nodes for self-adjoint differential operators, for which spatial discretization yields symmetric matrices to which the symmetric Lanczos algorithm can be applied.

For our first example, we consider the case of homogeneous Neumann boundary conditions, with the same operator. Let $x$ be a vector of $N$ uniformly spaced grid points in $[0, 2\pi)$, with spacing $\Delta x = 2\pi/N$. We denote by $m_j$ the $j$th normalized Lanczos vector, and by $v_j$ the $j$th un-normalized Lanczos vector. Given the initial values $m_0 = 0, u = \cos(\omega x)$, where $\omega = k/2, k = 0, 1, \ldots, N - 1$, and the differential operator $Lu = -pu_{xx} + q(x)u$ that is represented by a matrix $A$, we first find

$$m_1 = \frac{u}{\|u\|_2} = \frac{\cos(\omega x)}{\|\cos(\omega x)\|_2} = \frac{\cos(\omega x)}{\sqrt{\pi/\Delta x}}.$$ 

Then the first iteration of the Lanczos algorithm proceeds as follows:

$$v_1 = Am_1 \approx \sqrt{\frac{\Delta x}{\pi}} [p\omega^2 \cos(\omega x) + q \cos(\omega x)],$$

which yields

$$\alpha_1 = m_1^T v_1 \approx \frac{1}{\pi} \int_0^{2\pi} (\omega^2 p \cos^2(\omega x) + q \cos^2(\omega x)) \, dx.$$ 

(13)

Since $\int_0^{2\pi} \cos^2(\omega x) \, dx = \pi$, (13) simplifies to

$$\alpha_1 \approx p\omega^2 + \bar{q},$$

where $\bar{q}$ is the average value of $q(x)$ on $[0, 2\pi]$.

The next step in the Lanczos algorithm is to update

$$v_1 = v_1 - \beta_0 m_0 - \alpha_1 m_1$$

$$\approx \sqrt{\frac{\Delta x}{\pi}} [p\omega^2 \cos(\omega x) + q \cos(\omega x)] - (p\omega^2 + \bar{q}) \frac{\cos(\omega x)}{\sqrt{\pi/\Delta x}}$$

$$\approx (q - \bar{q}) \frac{\cos(\omega x)}{\sqrt{\pi/\Delta x}} = \tilde{q} \cos(\omega x),$$

(14)

where multiplication of vectors is component-wise. From (14), we obtain

$$\beta_1 = \|v_1\|_2 \approx \frac{\|\tilde{q} \cos(\omega x)\|_2}{\sqrt{\pi/\Delta x}}.$$ 

Continuing this process, we obtain, after neglecting lower-order terms,

$$\begin{bmatrix} \alpha_1 & \beta_1 & 0 \\ \beta_1 & \alpha_2 & \beta_2 \\ 0 & \beta_2 & \alpha_3 \end{bmatrix} \approx \begin{bmatrix} p\omega^2 & \|q \cos(\omega x)\|_2 & 0 \\ \|q \cos(\omega x)\|_2 & \sqrt{\pi/\Delta x} & p\omega^2 \\ 0 & 2p\omega \|q \sin(\omega x)\|_2 & \|q \cos(\omega x)\|_2 \\ 2p\omega \|q \sin(\omega x)\|_2 & \|q \cos(\omega x)\|_2 & p\omega^2 \end{bmatrix}.$$ 

Then, we obtain the approximate quadrature nodes using (12).

We now generalize to a two-dimensional domain $[0, 2\pi]^2$, with periodic boundary conditions in both directions. Let $\Delta x = \Delta y = 2\pi/N$, where $N$ is the number of grid points per dimension, and let $x$ be a $N^2 \times 2$ matrix of $x$- and $y$-coordinates of equally spaced grid points in $[0, 2\pi]^2$ with spacing $\Delta x$ and $\Delta y$. Given the initial values $m = 0, u = e^{i\omega x}$, where $\omega = (\omega_1, \omega_2)$ with $\omega_1, \omega_2 = -N/2 + 1, \ldots, N/2$, and the differential operator $L = -p\Delta + q(x, y)$ that is represented by a matrix $A$, we first find
\[ m_1 = \frac{e^{ix\bar{\omega}}}{\|e^{ix\bar{\omega}}\|_2} = \frac{\Delta x}{2\pi} e^{ix\bar{\omega}}. \]

Then, the first iteration of the Lanczos algorithm proceeds as follows:

\[ v_1 = A m_1 = \frac{\Delta x}{2\pi} p \|\bar{\omega}\|_2^2 e^{ix\bar{\omega}} + \frac{\Delta x}{2\pi} q e^{ix\bar{\omega}} \]

which yields

\[ \alpha_1 = v^H_1 v_1 = \frac{\|ar{\omega}\|_2^2}{4\pi^2} \int_0^{2\pi} \int_0^{2\pi} [p \|ar{\omega}\|_2^2 q] dy dx = \frac{\Delta x}{2\pi} \bar{q} e^{ix\bar{\omega}}. \]

where \( \bar{q} \) is the average value of the function \( q(x, y) \) over the rectangle \([0, 2\pi]^2\). We then update \( v_1 \) to obtain

\[ v_1 = v_1 - \beta_0 m_0 - \alpha_1 m_1 \]

\[ \approx \frac{\Delta x}{2\pi} \left[ \|ar{\omega}\|_2^2 \left( p e^{ix\bar{\omega}} + q e^{ix\bar{\omega}} - (p \|ar{\omega}\|_2^2 + \bar{q}) e^{ix\bar{\omega}} \right) \right] \]

\[ \approx \frac{\Delta x}{2\pi} (\bar{q} - \bar{q}) e^{ix\bar{\omega}} = \frac{\Delta x}{2\pi} \bar{q} e^{ix\bar{\omega}}. \]  \hspace{1cm} (15)

From (15) we obtain

\[ \beta_1 = \|v_1\|_2 \approx \frac{\Delta x}{2\pi} \|\bar{q}\|_2. \]

Continuing this process, we obtain, after neglecting lower-order terms,

\[ \begin{bmatrix} \alpha_1 & \bar{\beta}_1 & 0 \\ \beta_1 & \alpha_2 & \bar{\beta}_2 \\ 0 & \beta_2 & \alpha_3 \end{bmatrix} \approx \begin{bmatrix} p \|ar{\omega}\|_2^2 & \Delta x \|\bar{q}\|_2/(2\pi) \\ \Delta x \|\bar{q}\|_2/(2\pi) & p \|ar{\omega}\|_2^2 \\ 2p \|\bar{q} \cdot \bar{\omega}\|_2/s & 2p \|\bar{q} \cdot \bar{\omega}\|_2/s \end{bmatrix} \]

where \( \nabla q \) is the spatial discretization of \( \nabla q \). Then, as in (12), we obtain the estimated quadrature nodes

\[ \lambda_{1,\omega} = p \|\omega\|_2^2, \quad \lambda_{i,\omega} = p \|\omega\|_2^2 \pm \sqrt{\beta_1^2 + \beta_2^2}, \quad i = 2, 3. \]

3.2. Non-self-adjoint operators

When the spatial differential operator \( L \) is not self-adjoint, spatial discretization yields an unsymmetric matrix \( A \). Therefore, Arnoldi iteration is best used for the approximation of \( \varphi(A^\tau) b \) instead of unsymmetric Lanczos iteration, which can suffer from “serious breakdown” [9].

Arnoldi iteration produces an upper Hessenberg matrix \( H_m \) and matrix \( V_m \) with orthonormal columns such that

\[ AV_m = V_m H_m + h_{m+1,m} e_{m+1}^H. \]  \hspace{1cm} (16)

By analogy with (11), to approximate \( u = \varphi(A^\tau) b \), we could compute each discrete Fourier component \( [\hat{u}]_{\omega} \) of \( u \), corresponding to wave number \( \omega \), by applying block Arnoldi iteration [26] to \( A \), with initial block \( R_0(\omega) \) as defined in (10), and after \( m \) iterations that yield a block upper Hessenberg matrix \( H_m(\omega) \), we obtain the approximation

\[ [\hat{u}]_{\omega} = \left[ B_0^H E_{H}^H E_1(\omega^\tau) E_{12} B_0 \right]_{12}, \]  \hspace{1cm} (17)

where \( B_0 \) and \( E_{12} \) are as defined in Section 2.

The underlying quadrature rule is a block version of an Arnoldi quadrature rule as presented in [3]. Using the same approach as in [5, Theorem 1], it can be shown that for \( u^H v = 0 \), if \( m \) iterations of block Arnoldi are carried out on \( A \) with initial block \( [u \ v] \) to produce a block Hessenberg matrix \( H_m(\omega) \), then the quadrature rule

\[ u^H g(A) v \approx \|u\|_2 \|v\|_2 e_1^H g(H_m(\omega)) e_2 \]

is exact when \( g \) is a polynomial of degree \( 2m - 1 \) or less.

As in the case of block Lanczos, the eigenvalue problem for \( H_m(\omega) \) approximately decouples for high frequencies, due to the decay of the Fourier coefficients of \( b \). Therefore, we can approximate the frequency-dependent eigenvalues, which are used as interpolation points for a polynomial approximation of \( \varphi(\lambda) \), by applying non-block Arnoldi iteration, described
above, with initial vector \( \hat{e}_0 \), in the case of periodic boundary conditions, or an appropriate discretization of a sine or cosine function for homogeneous Dirichlet or Neumann boundary conditions, respectively.

In this subsection, we illustrate the use of Arnoldi iteration for selection of frequency-dependent interpolation points, in the high-frequency case. First, we consider a 2-D ADR (advection-diffusion-reaction) problem on \([0, 2\pi]^2\), with periodic boundary conditions (see Section 6.3). We will apply Arnoldi iteration to a matrix \( A \) that is a spatial discretization of the differential operator

\[
Lu = p \Delta u + q(u_x + u_y) + \phi(x, y)u,
\]

where \( p \) and \( q \) are constants. As before, we use the initial vector \( \mathbf{v}_0 = e^{ix\hat{\omega}} \), where \( \hat{\omega} = (\omega_1, \omega_2) \) contains the wave numbers, which are integers in \((-N/2, N/2)\), and \( x \) is a \( N^2 \times 2 \) matrix consisting of the coordinates of equally-spaced grid points, with spacing \( \Delta x = \Delta y = 2\pi/N \) where \( N \) is the number of grid points per dimension. We begin the first iteration of the Arnoldi algorithm by finding \( \mathbf{m}_1 \):

\[
\mathbf{m}_1 = \frac{e^{ix\hat{\omega}}}{\|e^{ix\hat{\omega}}\|_2} = \frac{\Delta x}{2\pi} e^{ix\hat{\omega}}.
\]

We then have

\[
\mathbf{v}_1 = A\mathbf{m}_1 \approx \frac{\Delta x}{2\pi} (-\|\hat{\omega}\|^2 p e^{ix\hat{\omega}} + qi e^{ix\hat{\omega}} (\omega_1 + \omega_2) + \phi e^{ix\hat{\omega}})
\]

where the vector \( \phi \) is the spatial discretization of \( \phi(x, y) \). In the next step, we compute

\[
h_{11} = \mathbf{m}_1^H \mathbf{v}_1
\]

\[
\approx \frac{1}{4\pi^2} \int_0^{2\pi} \int_0^{2\pi} e^{-ix\hat{\omega}} e^{ix\hat{\omega}} (-\|\hat{\omega}\|^2 p + qi(\omega_1 + \omega_2) + \bar{\phi}) dxdy
\]

\[
\approx -\|\hat{\omega}\|^2 p + \bar{q}(\omega_1 + \omega_2) + \bar{\phi},
\]

where, as before, \( \bar{q} \) and \( \bar{\phi} \) are the averages of \( q(x, y) \) and \( \phi(x, y) \) over the rectangle \([0, 2\pi]^2\), respectively. We then update \( \mathbf{v}_1 \) as follows:

\[
\mathbf{v}_1 = \mathbf{v}_1 - h_{11} \mathbf{m}_1 \approx \frac{\Delta x}{2\pi} e^{ix\hat{\omega}} (\phi - \bar{\phi}) \approx \frac{\Delta x}{2\pi} e^{ix\hat{\omega}} \bar{\phi},
\]

where, as before, \( \phi \) is the spatial discretization of \( \phi(x, y) \). Next, we compute

\[
h_{21} = \|\mathbf{v}_1\|_2 \approx \frac{\Delta x}{2\pi} \|\bar{\phi}\|_2
\]

We conclude the first outer iteration of the Arnoldi algorithm by computing

\[
\mathbf{m}_2 = \frac{\mathbf{v}_1}{h_{21}} = \frac{e^{ix\hat{\omega}} \bar{\phi}}{\|\bar{\phi}\|_2}.
\]

Continuing in this fashion, we set \( \mathbf{v}_2 = A\mathbf{m}_2 \) and then in the second outer iteration we obtain

\[
h_{12} = \mathbf{m}_1^H \mathbf{v}_2
\]

\[
\approx \frac{1}{2\pi \Delta x \|\bar{\phi}\|_2} \int_0^{2\pi} \int_0^{2\pi} \{ p(-\|\hat{\omega}\|^2 \bar{\phi} + 2i\nabla \phi \cdot \hat{\omega} + \Delta \phi) + qi\bar{\phi}(\omega_1 + \omega_2) \\
+ q(\phi_x + \phi_y) + \phi \bar{\phi} \} dxdy,
\]

\[
\approx \frac{\Delta x \|\bar{\phi}\|_2}{2\pi},
\]

\[
h_{22} = \mathbf{m}_2^H \mathbf{v}_2
\]

\[
\approx -p\|\hat{\omega}\|_2^2 - \frac{p(\|\phi_x\|^2 + \|\phi_y\|^2)}{\|\bar{\phi}\|_2} + q(\omega_1 + \omega_2) + \bar{\phi}\bar{\phi},
\]

where \( \bar{\phi}\bar{\phi} \) is the average of the function \( \phi \), with weight function \( \bar{\phi}^2 \), on the rectangle \([0, 2\pi]^2\). In the next step we update \( \mathbf{v}_2 \) as follows:
\[ v_2 = \text{Am}_2 - h_{12}\text{m}_1 - h_{22}\text{m}_2 \] (26)

\[ \approx \frac{1}{\|\phi\|^2} \left[ p e^{i\omega \lambda} (2i\nabla \phi \cdot \omega + \Delta \phi) + q e^{i\phi \omega} (\phi_x + \phi_y) + e^{i\omega \lambda} \tilde{\phi} \right] \\
- \frac{\Delta x^2 \|\phi\|^2 e^{i\omega \lambda}}{4\pi^2} - \frac{e^{i\omega \lambda} \tilde{\phi}}{\|\phi\|^2} \left( - p (\|\phi_x\|^2 + 2\|\phi_y\|^2) + \|\phi\|^2 + \tilde{\phi} \right) . \]

Dropping the lower order \( \omega \)-terms in (26) yields

\[ v_2 \approx \frac{2ip \nabla \phi \cdot \omega e^{i\omega \lambda}}{\|\phi\|^2}. \] (27)

Continuing, we then compute

\[ h_{32} = \|v_2\|_2 \approx \frac{2p \|\nabla \phi \cdot \omega\|^2}{\|\phi\|^2}, \]

\[ \text{m}_3 = v_2 / h_{32}, \]

\[ v_3 = \text{Am}_3, \]

\[ h_{13} = \text{m}^H_3 v_3 \]

\[ \approx \frac{i}{2\pi \Delta x \|\nabla \phi \cdot \omega\|^2} \left[ \int \int \left[ p (\|\phi\|^2 + 2i\nabla \phi \cdot \nabla \phi) + 2i\nabla \phi \cdot \omega \right] \\
+ \Delta (\nabla \phi \cdot \omega) + q i ((\nabla \phi \cdot \omega) x + (\nabla \phi \cdot \omega) y) \\
+ qi (\nabla \phi \cdot \omega)(\omega_1 + \omega_2) + \phi (\nabla \phi \cdot \omega) \right] dx dy \\
\approx 0, \]

\[ h_{23} = \text{m}^H_2 v_3 \]

\[ \approx \frac{2p \|\nabla \phi \cdot \omega\|^2}{\|\phi\|^2}, \]

\[ h_{33} = \text{m}^H_3 v_3 \]

\[ \approx -p \|\omega\|^2 - \frac{p (\|\nabla \phi \cdot \omega\|^2 + \|\nabla \phi \cdot \omega\|^2)}{\|\phi\|^2} + \tilde{\phi} (\omega_1 + \omega_2) + \tilde{\phi} (\nabla \phi \cdot \omega)^2 . \]

We see that the matrix \( H_3 \) is well approximated by a matrix that is complex symmetric and tridiagonal. Furthermore, as the diagonal entries are all equal except for lower-order terms in \( \|\omega\|^2 \), we can readily estimate the eigenvalues of \( H_3 \), for a 3rd-order method, by

\[ \lambda_{1,2} = h_{11} \pm h_{12} , \]

whereas for a 5th-order method, the eigenvalues of \( H_3 \) can be estimated using

\[ \lambda_1 = h_{11} , \quad \lambda_{2,3} = h_{11} \pm \sqrt{h_{12}^2 + h_{23}^2} . \]

Next, we consider a system of coupled PDEs, based on a linearization of the 2-D Brusselator equation (see Section 6.4). The system is

\[ u_t = \alpha \Delta u + pu + \phi v, \]

\[ v_t = \alpha \Delta v + q v + \psi u , \]

with appropriate initial conditions and periodic boundary conditions. It is assumed that \( \alpha \) is a constant, and all other coefficients are variable. Following an approach described in [19] for constructing basis functions for a system of coupled PDEs, it can be shown that at high frequencies, these basis functions can be approximated by \((e^{i\omega \lambda}, 0)\) and \((0, e^{i\omega \lambda})\), for each wave number \( \omega \).

Therefore, given the operator \( L \) and the initial vector \( v_0 \) defined by

\[ L = \begin{bmatrix} \alpha \Delta + p & \phi \\
\psi & \alpha \Delta + q \end{bmatrix}, \quad v_0 = \begin{bmatrix} e^{i\omega \lambda} \\
0 \end{bmatrix}, \]

\[ v_0 = \begin{bmatrix} e^{i\omega \lambda} \\
0 \end{bmatrix} \cdot \begin{bmatrix} e^{i\omega \lambda} \end{bmatrix}. \] (28)
with a matrix A representing a spatial discretization of L, we apply the Arnoldi algorithm to A with the initial vector \( \psi_0 \) being the first vector given in (28); the iteration with the second vector is similar. Proceeding as in the previous analysis, we obtain the following entries of \( H_3 \):

\[
h_{11} = \mathbf{m}_1^\dagger \mathbf{v}_1 \approx \frac{1}{4\pi^2} \int_0^{2\pi} \int_0^{2\pi} \left( -\alpha \| \mathbf{w} \|_2^2 + p \right) dx \, dy \approx -\alpha \| \mathbf{w} \|_2^2 + \tilde{p}.
\]

\[
h_{21} = \| \mathbf{v}_1 \|_2 \approx \frac{\sqrt{\| \mathbf{p} \|_2^2 + \| \mathbf{w} \|_2^2}}{2\pi / \Delta x},
\]

\[
h_{12} = \mathbf{m}_1^\dagger \mathbf{v}_2 \approx \frac{\| \mathbf{p} \|_2^2 + \| \mathbf{w} \|_2^2}{2\pi / \Delta x \| \mathbf{p} \|_2^2 + \| \mathbf{w} \|_2^2},
\]

\[
h_{22} = \mathbf{m}_2^\dagger \mathbf{v}_2 \approx \| \mathbf{v}_2 \|_2 \approx \frac{2\alpha \sqrt{\| \nabla \mathbf{p} \cdot \mathbf{w} \|_2^2 + \| \mathbf{p} \|_2^2 + \| \nabla \mathbf{w} \cdot \mathbf{w} \|_2^2}}{| \mathbf{p} \|_2^2 + \| \mathbf{w} \|_2^2},
\]

\[
h_{32} = \| \mathbf{v}_2 \|_2 \approx \frac{2\alpha \sqrt{\| \nabla \mathbf{p} \cdot \mathbf{w} \|_2^2 + \| \mathbf{p} \|_2^2 + \| \nabla \mathbf{w} \cdot \mathbf{w} \|_2^2}}{| \mathbf{p} \|_2^2 + \| \mathbf{w} \|_2^2},
\]

\[
h_{13} \approx \frac{i\phi(\mathbf{w} \cdot \mathbf{w})}{2\pi / \Delta x \| \nabla \mathbf{p} \cdot \mathbf{w} \|_2^2 + \| \mathbf{p} \|_2^2 + \| \nabla \mathbf{w} \cdot \mathbf{w} \|_2^2},
\]

\[
h_{23} \approx \frac{2\alpha (\| \nabla \mathbf{p} \cdot \mathbf{w} \|_2^2 + \| \nabla \mathbf{w} \cdot \mathbf{w} \|_2^2 + \| \mathbf{p} \|_2^2 + \| \nabla \mathbf{w} \cdot \mathbf{w} \|_2^2)}{\sqrt{\| \mathbf{p} \|_2^2 + \| \mathbf{w} \|_2^2 + \| \nabla \mathbf{w} \cdot \mathbf{w} \|_2^2} \sqrt{\| \nabla \mathbf{p} \cdot \mathbf{w} \|_2^2 + \| \mathbf{p} \|_2^2 + \| \nabla \mathbf{w} \cdot \mathbf{w} \|_2^2}},
\]

\[
h_{33} \approx -\alpha \| \mathbf{w} \|_2^2 - \frac{\alpha (\| \nabla \mathbf{p} \cdot \mathbf{w} \|_2^2 + \| \nabla \mathbf{w} \cdot \mathbf{w} \|_2^2 + \| \mathbf{p} \|_2^2 + \| \nabla \mathbf{w} \cdot \mathbf{w} \|_2^2)}{\sqrt{\| \mathbf{p} \|_2^2 + \| \mathbf{w} \|_2^2 + \| \nabla \mathbf{w} \cdot \mathbf{w} \|_2^2} \sqrt{\| \nabla \mathbf{p} \cdot \mathbf{w} \|_2^2 + \| \mathbf{p} \|_2^2 + \| \nabla \mathbf{w} \cdot \mathbf{w} \|_2^2}}
\]

As expected, the matrix \( H_3 \) does not have any kind of symmetry; however, it is worth noting that except for lower-order terms in \( \| \mathbf{w} \|_2 \), \( h_{32} \) and \( h_{23} \) are equal, as are all of the diagonal entries.

For a 3rd-order method, our estimated nodes are

\[
\lambda_{1,2} = h_{11} \pm \sqrt{h_{12}h_{21}}.
\]

For a 5th-order method, we neglect \( h_{13} \), as it is of lower order, to obtain the estimated nodes

\[
\lambda_1 = h_{11}, \quad \lambda_{2,3} = h_{11} \pm \sqrt{h_{12}h_{21} + h_{23}h_{32}}.
\]

To justify neglecting \( h_{13} \), we perform a perturbation analysis. Since the diagonal entries are equal modulo lower-order terms, we consider the matrix

\[
H = \begin{bmatrix}
  h_{11} & h_{12} & h_{13} \\
  h_{21} & h_{11} & h_{23} \\
  0 & h_{32} & h_{11}
\end{bmatrix}
\]

and let \( \tilde{H} \) be equal to \( H \) except with \( h_{13} = 0 \). Then, the characteristic polynomials of \( \tilde{H} \) and \( H \) are

\[
\tilde{p}(\lambda) = (\lambda - h_{11})^3 - (h_{12}h_{21} + h_{23}h_{32})(\lambda - h_{11}), \quad p(\lambda) = \tilde{p}(\lambda) - h_{13}h_{12}h_{32}.
\]

Let \( \lambda_j \) be an eigenvalue of \( \tilde{H} \). From

\[
p(\lambda_j + \delta_j) = 0, \quad \tilde{p}(\lambda_j) = 0,
\]

and a Taylor expansion of \( p \) around \( \lambda_j \), we obtain, for \( j \neq 1 \),

\[
\delta_j \approx \frac{p(\lambda_j)}{p'(\lambda_j)} = \frac{h_{13}h_{12}h_{32}}{2(h_{12}h_{21} + h_{23}h_{32})} = O(\| \mathbf{w} \|_2) = O(\| \mathbf{w} \|_2^{-1}).
\]
and a similar result for \( j = 1 \), except for a constant factor. Fig. 2 confirms that the eigenvalues of \( H \) and \( \tilde{H} \) are in close agreement, with the error decaying to zero as \( \| \tilde{\omega} \|_2 \) increases, and at the same rate as in this analysis.

4. EPI methods

We now give a brief description of exponential propagation iterative (EPI) methods, introduced by Tokman [27]. Suppose that we have a nonlinear autonomous system of ODEs of the form (1). Then we use the Taylor expansion of \( F(y(t)) \) around \( y(t_n) \) to obtain

\[
\frac{dy}{dt} = F(y(t_n)) + A_n(y(t) - y(t_n)) + R(y(t)),
\]

(29)

where \( A_n = \frac{dF(y(t_n))}{dy} \) is the Jacobian of \( F(y(t)) \) and \( R(y(t)) \) is the nonlinear remainder function. Using an integrating factor \( e^{-A_n t} \) and integrating (29) over the time interval \([t_n, t_{n+1}]\) gives us the integral form of (1).

\[
y(t_{n+1}) = y(t_n) + [e^{A_n \Delta t} - I]A_n^{-1}F(y(t_n)) + \int_{t_n}^{t_{n+1}} e^{A_n(t_{n+1} - \tau)}R(y(\tau))d\tau.
\]

(30)

Then, the integral term is approximated numerically, which requires the approximation of products of matrix functions and vectors of the form \( \varphi(A\tau)b \).

These products are evaluated using a Krylov subspace approximation in the following way:

\[
\varphi(A\tau)b \approx \|b\|_2 V_m\varphi(H_m\tau)e_1,
\]

(31)

where \( H_m \) is an upper Hessenberg matrix which is given by \( H_m = V_m^TAV_m \), and \( V_m = [v_1 \ v_2 \ \cdots \ v_m] \), where \( \{v_1, v_2, \ldots, v_m\} \) is an orthonormal basis of the Krylov subspace \( K_m(A, b) \), which can be obtained using the Arnoldi algorithm [9]. The accuracy of the approximation in (31) depends on the number of the Krylov vectors constructed, the eigenvalues of \( A \), the magnitude of \( \tau \), and \( \varphi \).

For larger matrices \( A \) and larger time steps \( \tau \), the number of iterations \( m \) can increase substantially. When this occurs, convergence can be hindered by the appearance of spurious high-frequency oscillations in the columns of \( V_m \), even when the initial vector \( b \) represents a very smooth function. This is illustrated in Fig. 3. For this reason, in the numerical experiments performed in Section 6.2, a simple denoising process is applied after each matrix-vector multiplication, in which each Fourier coefficient whose magnitude is below a certain threshold is zeroed. In our implementation, we zeroed each element \( \hat{u}(\omega) \) of the vector of Fourier coefficients \( \hat{u} \) such that \( |\hat{u}(\omega)| < 10\epsilon \|\hat{u}\|_2 \), where \( \epsilon \) is the relative tolerance specified for the convergence of the Arnoldi iteration itself. It was found that denoising was unnecessary for the other test problems featured in Section 6, due to their more significant diffusion terms.

The effect of this denoising can be seen in Fig. 4. As can be seen in the results presented in Sections 6.2 and 6.3, this can substantially improve efficiency without sacrificing accuracy. However, it is important to select this threshold properly in order to realize this gain in efficiency without causing the Arnoldi iteration to break down due to near-linear dependence of the Krylov subspace basis vectors. Future work will include adaptive selection of this threshold.

The behavior of the unfiltered Krylov vectors is not surprising, as similar behavior is displayed by the unsmoothed Fourier method applied to hyperbolic PDEs [10]. In that work, the proposed remedies were to either use filtering, or increase the
number of grid points; the former remedy serves as the motivation for denoising in this context. It is worth noting that in both the unfiltered and filtered cases, no loss of orthogonality was observed in the Krylov vectors; that is, $1/\sqrt{m}V^T_mV_m - I_m\parallel_F$ was negligibly small (that is, on the order of $10^{-10}$ or smaller) in both cases.

5. KSS–EPI methods

The combination of KSS and EPI methods is easily described: whenever an EPI method computes a matrix function–vector product of the form $\psi(\tau)b$, for some function $\psi$, matrix $A$, scaling parameter $\tau$ based on the time step, and vector $b$, the following procedure is carried out in place of standard Krylov projection as in (31):

1. Use an FFT to decompose $b = b_L + b_H$, where $b_L$ consists of low-frequency components and $b_H$ contains all other (high-frequency) components. This is accomplished by zeroing all Fourier coefficients of $b_L$ for which the absolute value of any wave number exceeds a selected threshold.
2. Use standard Krylov projection as in (31) to compute $\psi(\tau)b_L$.
3. Use KSS, with nodes prescribed as in Section 3, to compute $\psi(\tau)b_H$.
4. Combine the results of steps 2 and 3 to obtain $\psi(\tau)b$.

The decomposition of $b$ in step 1 must be chosen so that the computation in step 2 does not require many more Krylov projection steps than would be required for step 3, which is one more than the desired order of temporal accuracy. However, it is also important to not include too few low-frequency components in $b_L$, as the nodes prescribed in Section 3 are based on a high-frequency analysis and are therefore not effective choices at low frequencies [23].

We will denote by $N_c$ the cutoff point for the low-frequency components. Specifically, a Fourier coefficient $\hat{b}(\omega)$ of $b$ will be zeroed in $b_L$ if $\parallel\hat{b}(\omega)\parallel_\infty \geq N_c$. In this work, the $N_c$ value has been determined by experimentation. Future work will include development of an adaptive approach to this decomposition, based on criteria such as the smoothness of the solution and number of iterations required for convergence in step 2 from previous time steps.
We now elaborate on how step 3 can be performed more efficiently, by minimizing the number of FFTs. Recall from Section 2 that when \( K \) iterations of block Lanczos (or block Arnoldi, in the case where \( L \) is not self-adjoint) are performed in KSS methods for PDEs with a first-order time derivative, the temporal error is \( O(\Delta t^{2K-1}) \). In this case, there are \( 2K \) total quadrature nodes for each Fourier component, with wave number \( \omega \).

As discussed in Section 3, from the decoupling of the block tridiagonal matrix \( T_K \) in the high-frequency limit (or a block upper Hessenberg matrix, in the case of block Arnoldi iteration), half of these quadrature nodes depend on \( \omega \) and half do not. For each \( \omega \), the frequency-independent nodes are \( \{ \lambda_1, \lambda_2, \ldots, \lambda_K \} \) and the frequency-dependent nodes are \( \{ \lambda_{1,\omega}, \lambda_{2,\omega}, \ldots, \lambda_{K,\omega} \} \).

The frequency-independent nodes are obtained by applying Arnoldi (or Lanczos, as appropriate) iteration to \( b_H \), as described in Section 4, and computing the eigenvalues of \( H_K \). The frequency-dependent nodes are estimated using the coefficients of the differential operator on which \( A \) is based, as described in Section 3.

The Fourier component of \( \psi(At) b_H \) corresponding to the wave number \( \omega \) is obtained by computing the same Fourier component of \( p_{2K-1}(At)b_H \), where \( p_{2K-1} \) is the polynomial interpolant of \( \psi(\lambda) \) with interpolation points \( \{ \lambda_1, \lambda_{1,\omega} \}_{i=1}^K \). Expressing this interpolant in Newton form, we have

\[
p_{2K-1}(\lambda) = \sum_{j=1}^K \varphi[\lambda_1, \ldots, \lambda_j] \prod_{i=1}^{j-1} (\lambda - \lambda_i)
+ \sum_{j=1}^K \varphi[\lambda_1, \ldots, \lambda_K, \lambda_{1,\omega}, \ldots, \lambda_{j,\omega}] \prod_{i=1}^{j-1} (\lambda - \lambda_{i,\omega}) \prod_{k=1}^{K} (\lambda - \lambda_k).
\]

Arranging the interpolation points in the order indicated above allows us to reduce the number of FFTs needed. Using the relation from Lanczos iteration,

\[
AX_K = X_K T_K + r_K e_K^T,
\]

we define

\[
v = p_{K-1}(A)b_H = [\varphi[\lambda_1] + \varphi[\lambda_1, \lambda_2](A-\lambda_1 I) + \cdots + \varphi[\lambda_1, \lambda_2, \ldots, \lambda_K](A-\lambda_1 I) \cdots (A-\lambda_{K-1} I) ] b_H \\
= ||b_H|| X_K p_{K-1}(H_K) e_1,
\]

\[
w = q_K(A)b_H = (A-\lambda_1 I) \cdots (A-\lambda_K I) b_H \\
= \beta_1 \beta_2 \cdots \beta_{K-1} r_K \\
= \beta_1 \cdots \beta_K X_{K+1} e_{K+1},
\]

and

\[
\hat{p}_{K-1}(\lambda) = \varphi[\lambda_1, \ldots, \lambda_K, \lambda_{1,\omega}] + \varphi[\lambda_1, \ldots, \lambda_{1,\omega}](\lambda - \lambda_{1,\omega}) + \cdots + \varphi[\lambda_1, \ldots, \lambda_K, \lambda_{\omega}](\lambda - \lambda_{1,\omega}) \cdots (\lambda - \lambda_{K-1,\omega}) \\
= C_{K-1}^0 + \cdots + C_{1,\omega}^0 + C_0^\omega.
\]

Then, using \( \mathcal{F} \) to denote the discrete Fourier transform, we have

\[
\psi(At)b_H \approx p_{2m-1}(At)b_H = v + \hat{p}_{K-1}(A)w = v + \mathcal{F}^{-1} \sum_{j=0}^{K-1} [C_j^\omega] \mathcal{F} A^j w,
\]

and it can easily be seen that the solution at each time step requires \( K \) FFTs and one inverse FFT. The coefficients \( C_j^\omega \), \( j = 0, 1, \ldots, K - 1 \), of the power form of \( \hat{p}_{K-1} \) can easily be obtained by repeatedly applying nested multiplication to the last \( K \) terms of the Newton form of \( p_{2K-1}(\lambda) \).

As discussed in Section 2, the total number of quadrature nodes, \( 2K \), yields a temporal order of accuracy of \( O(\Delta t^{2K-1}) \). Therefore, a lower bound for \( K \) is dictated by the order of accuracy of the underlying EPI method. While efficient implementation is most readily achieved if the same value of \( K \) is chosen for each Fourier coefficient, it is possible to vary \( K \) adaptively, using the error estimate

\[
E_{\omega, K} = \tau^{2K} \varphi^{(2K)}(\xi) (2K)! w^H (A - \lambda_{1,\omega}) \cdots (A - \lambda_{K,\omega}) \hat{e}_\omega,
\]

which is derived from the error formula for block Gaussian quadrature [7].
6. Numerical results

In this section we compare several versions of EPI methods, as applied to four test problems. The versions differ in the way in which they compute matrix function–vector products of the form $\varphi(At)b$:

- Standard Krylov projection, as in (31), hereafter referred to as “Krylov–EPI”, either with or without denoising as in Section 4,
- Using the KSS approach, as described in Section 5, hereafter referred to as “KSS–EPI”
- Newton interpolation using Leja points [2], hereafter referred to as “LEJA”, and
- Adaptive Krylov projection [24], hereafter referred to as “AKP”.

For the first test problem, as it is the only one with a symmetric Jacobian, rational Krylov subspace reduction with optimized shifts [4], hereafter referred to as “RKS".

All of these approaches are used in the context of two EPI methods. The first is a 3rd-order, 2-stage EPI method [27]

$$
Y_1 = y_n + \frac{1}{3} h a_{11} \varphi_1 \left(\frac{1}{3} h A \right) F(y_n),
$$

(35)

$$
y_{n+1} = y_n + h \varphi_1(hA) F(y_n) + 3 h b_1 \varphi_2(hA)[F(Y_1) - F(y_n) - A(Y_1 - y_n)],
$$

where $a_{11} = 9/4$ and $b_1 = 32/81$, and

$$
R(Y_1) = F(Y_1) - F(y_n) - A(Y_1 - y_n).
$$

For this method,

$$
\varphi_1(\lambda) = \frac{e^\lambda - 1}{\lambda}, \quad \varphi_2(\lambda) = \frac{e^\lambda - \lambda - 1}{\lambda^2}, \quad \varphi_3(\lambda) = \frac{e^\lambda (6 - \lambda) - (6 + 5\lambda + 2\lambda^2)}{\lambda^3}.
$$

The second is a 5th-order, 3 stage EPI method [28]

$$
Y_1 = y_n + h a_{11} \psi_1 (g_{11} h A) F(y_n),
$$

(36)

$$
Y_2 = y_n + h a_{21} \psi_1 (g_{21} h A) F(y_n) + h a_{22} \psi_2 (g_{22} h A) R(Y_1),
$$

$$
y_{n+1} = y_n + h b_1 \psi_1 (g_{31} h A) F(y_n) + h b_2 \psi_2 (g_{32} h A) R(Y_1) + h b_3 \psi_3 (g_{33} h A) [-2R(Y_1) + R(Y_2)],
$$

where

$$
\psi_i(z) = \sum_{j=1}^{j} p_{ij} \psi_j(z), \quad i = 1, 2, 3,
$$

and the coefficients $g_{ij}$, $a_{ij}$, $b_j$, and $p_{ij}$ are obtained from the description of the EPIRK5s3 method in [28].

Errors reported are the relative errors computed with respect to an “exact” solution that is obtained using the MATLAB ODE solver ode15s with the smallest allowable time step. For all test problems, various grid sizes are used to demonstrate the effect of increased resolution on performance; throughout this section, $N$ refers to the number of grid points per dimension.

For all methods used to compute matrix function–vector products, efficiency will be measured in two ways: (1) total execution time required to integrate over the entire time interval, and (2) the average number of matrix–vector products (plus the average number of FFTs, in the case of KSS–EPI, or the average number of linear system solves, in the case of RKS) performed for each evaluation of a matrix function–vector product of the form $\varphi(At)b$. The phrase “number of iterations” is used throughout this section to refer to this quantity.

The implementations of KSS–EPI, Krylov–EPI and RKS determine convergence of Krylov projection through comparison of consecutive iterates. For this reason, the tolerance varies with the time step. Specifically, as the time step is reduced by a factor of 2, the tolerance is reduced by a factor of $2^p$, where $p$ is the order of the EPI method. The tolerance used for the largest time step is $10^{-4}$ for 3rd order, and $5 \times 10^{-8}$ for 5th order. The slightly larger tolerance for 5th order is chosen so as to avoid roundoff issues at smaller time steps. The implementations of AKP and LEJA do not determine convergence in this manner, so their tolerance is set to $10^{-8}$ for all time steps.

6.1. Diffusive problem

The first test problem is the two-dimensional Allen–Cahn equation given by

$$
\frac{\partial u}{\partial t} = \alpha \nabla^2 u + u - u^3, \quad x, y \in [0, 1], \quad t \in [0, 0.2]
$$

(37)

with $\alpha = 0.1$, using homogeneous Neumann boundary conditions and initial conditions given by
Fig. 5. Relative error plotted against execution time for solving the Allen–Cahn equation (37) using the 3rd-order EPI method (35). Matrix function–vector products are computed using KSS–EPI (magenta solid curves), Krylov–EPI (blue solid curves), AKP (red dashed curves), LEJA (green dashed–dotted curves), and RKSR (black dotted curves), on grids with $N = 50$, $150$ and $300$ points per dimension. Time steps used are $\Delta t = (0.2)^{-p}$, for $p = 0, 1, 2, 3, 4$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

$$u_0(x, y) = 0.4 + 0.1 \cos(2\pi x) \cos(2\pi y).$$

The $\nabla^2$ term is discretized using a centered finite difference. For KSS–EPI, the low-frequency portion $b_1$ consists of all components with wave numbers $\omega_i \leq 7$, $i = 1, 2$. The low value of this threshold is due to the smoothness of the initial data. That is, the value of $N_c$ for this problem is 7.

Figs. 5 and 7 show the error vs. time performance for the various approaches to matrix–function–vector multiplication, used within the 3rd- and 5th-order EPI methods, respectively. We can see from these figures that the errors for both Krylov–EPI and KSS–EPI methods are essentially the same, but the computation time is different. For both orders, only for the grid size $N = 50$ grid points per dimension is the efficiency of Krylov–EPI comparable to that of KSS–EPI, while it is significantly slower for $N = 150$ and $N = 300$. Most significantly, as the number of grid points increases, the increase in computational expense is much more pronounced with Krylov–EPI, due to the increase in Krylov projection steps needed for convergence as can be seen in Figs. 6 and 8. For both methods, the same matrix is being used for Lanczos iteration, but in KSS–EPI, the initial vector is only a low-frequency approximation of that used for Krylov–EPI, thus drastically reducing the number of iterations needed.

From Figs. 6 and 8, it can be seen that for KSS–EPI, the number of overall iterations (matrix–vector multiplications + FFTs) shows almost no sensitivity to the grid size, compared to Krylov–EPI, AKP and Leja interpolation, all of which exhibit substantial growth as the number of grid points increases. While the number of matrix–vector multiplications and system solves for RKSR is also essentially independent of the grid size, the increase in computation time is again more substantial than for KSS–EPI. Specifically, between $N = 50$ and $N = 300$, the time for RKSR increased by a factor of approximately 100, while the time for KSS–EPI increased by a factor of roughly 10 (while the total number of grid points increased by a factor of 36). It should be noted that the results shown for RKSR were achieved only after considerable experimentation with a parameter, the length of an equidistributed sequence [4, Theorem 4.4], which determines maximum allowable number of shifts. If this value is chosen to be too small, convergence does not occur as $\Delta t \to 0$, and if it is chosen too large, then some...
Fig. 6. Average number of matrix–vector products, shown on a logarithmic scale, per matrix function–vector product evaluation for each method when solving the Allen–Cahn equation (37) using the 3rd-order EPI method (35). For KSS–EPI, FFTs are also included, and for RKSR, linear system solves are also included. For each method, bars correspond to grid sizes of $N = 50, 150, 300$ points per dimension, from left to right. Left plot: $\Delta t = 0.2$. Right plot: $\Delta t = 0.0125$.

Fig. 7. Relative error plotted against execution time for solving the Allen–Cahn equation (37) using the 5th-order EPI method (36). Matrix function–vector products are computed using KSS–EPI (magenta solid curves), Krylov–EPI (blue solid curves), AKP (red dashed curves), LEJA (green dashed–dotted curves), and RKSR (black dotted curves), on grids with $N = 50, 150$ and $300$ points per dimension. Time steps used are $\Delta t = (0.2)2^{-p}$, for $p = 0, 1, 2, 3, 4$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)
efficiency is lost. Furthermore, it was observed that the optimal value of this parameter depends on the grid size. Although AKP is slightly more accurate than KSS–EPI for 5th order, this is more than offset by the superior efficiency of KSS–EPI, particularly at larger grid sizes and larger time steps.

Table 1 illustrates the sensitivity of the results, in terms of both accuracy and efficiency, to the threshold value $N_c$ that is used to obtain the low- and high-frequency portions $b_L$ and $b_H$ of each vector $b$ in each computation of the form $\varphi(At)b$. As $N_c = 7$ was used in the results presented above, we examine nearby values. As can be seen in the table, the results are not unduly affected by changes in this parameter.

Nonetheless, it is still necessary to develop an efficient, adaptive approach to selecting a reasonably optimal value of $N_c$. If it is chosen too large, then unnecessary Krylov projection steps may be taken. On the other hand, if it is chosen too small, then the accuracy can suffer because the interpolation points used by KSS are only reasonable approximations of block Gaussian quadrature nodes at sufficiently high frequencies, in view of the analysis used to obtain them.

### 6.2. Advective problem

The second test problem is the one-dimensional Burgers’ equation

$$u_t + uu_x = \nu u_{xx}, \quad x \in [0, 1], \quad t \in [0, 1]$$

with $\nu = 0.03$, using Dirichlet boundary conditions and initial condition

$$u_0(x) = \sin^2(3\pi x)(1 - x)^{3/2}.$$

For KSS–EPI, the low-frequency portion $b_L$ consists of all components with wave numbers $\omega \leq N_c$. A higher threshold ($N_c = 40$ for 3rd-order, $N_c = 80$ for 5th-order) is used in this problem than for the Allen–Cahn equation, as the initial data is less smooth and the PDE is advection-dominated rather than diffusion-dominated.

For this test problem, both the Krylov–EPI and KSS–EPI methods have similar error for the same time steps, as can be seen in Figs. 9 and 11. The difference in computation time is already significant for $N = 500$ between KSS–EPI and Krylov–EPI for both 3rd and 5th order, but as the grid size increases, we can see that KSS–EPI is showing far superior
Fig. 9. Relative error plotted against execution time for solving Burgers’ equation (38) using the 3rd-order EPI method (35). Matrix function–vector products are computed using KSS–EPI with denoising (magenta solid curves), KSS–EPI without denoising (black dotted curves), Krylov–EPI (blue solid curves), AKP (red dashed curves), and LEJA (green dashed–dotted curves), on grids with \( N = 500, 1500 \) and 3000 points. Time steps used are \( \Delta t = (0.1) 2^{-p} \), for \( p = 0, 1, 2, 3, 4 \). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Fig. 10. Average number of matrix–vector products, shown on a logarithmic scale, per matrix function–vector product evaluation for each method when solving Burgers’ equation (38) using the 3rd-order EPI method (35). For KSS–EPI and KSS–EPI denoised (abbreviated “dn”), FFTs are also included. For each method, bars correspond to grid sizes of \( N = 500, 1500, 3000 \) points, from left to right. Left plot: \( \Delta t = 0.1 \). Right plot: \( \Delta t = 0.00625 \).
Fig. 11. Relative error plotted against execution time for solving Burgers’ equation (38) using the 5th-order EPI method (36). Matrix function–vector products are computed using KSS–EPI with denoising (magenta solid curves), KSS–EPI without denoising (black dotted curves), Krylov–EPI (blue solid curves), AKP (red dashed curves), and LEJA (green dashed–dotted curves), on grids with \( N = 500, 1500 \) and 3000 points. Time steps used are \( \Delta t = (0.1)^{2-p} \), for \( p = 0, 1, 2, 3, 4 \). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Fig. 12. Average number of matrix–vector products, shown on a logarithmic scale, per matrix function–vector product evaluation for each method when solving Burgers’ equation (38) using the 5th-order EPI method (36). For KSS–EPI and KSS–EPI denoised (abbreviated “dn”), FFTs are also included. For each method, bars correspond to grid sizes of \( N = 500, 1500, 3000 \) points, from left to right. Left plot: \( \Delta t = 0.1 \). Right plot: \( \Delta t = 0.00625 \).
Fig. 13. Relative error plotted against execution time for solving the 2-D ADR equation (39) using the 3rd-order EPI method (35). Matrix function–vector products are computed using KSS–EPI with denoising (magenta solid curves), Krylov–EPI (blue solid curves), AKP (red dashed curves), and LEJA (green dashed–dotted curves), on grids with N = 50, 150 and 300 points per dimension. Time steps used are Δt = (0.01)2−p, for p = 0, 1, 2, 3, 4. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

efficiency compared to the Krylov–EPI method. As shown in Figs. 10 and 12, this is again due to the increasing number of Krylov projection steps needed for Krylov–EPI.

Denoising applied to KSS–EPI is advantageous for this problem, unlike with the Allen–Cahn equation. As seen in Figs. 10 and 12, for KSS–EPI without denoising, the number of iterations is increasing with the number of grid points, though not as rapidly as with the other methods. However, with denoising included, the insensitivity of the number of iterations to the grid size is restored. It is worth mentioning that both Leja interpolation and AKP are more accurate and more efficient than Krylov–EPI. At the same time, both Leja interpolation and AKP are more accurate but much slower than KSS–EPI, in terms of computation time and number of matrix–vector products.

6.3. 2D ADR problem

The third test problem is the two-dimensional advection–diffusion–reaction equation

\[ u_t = \epsilon (u_{xx} + u_{yy}) - \alpha (u_x + u_y) + \gamma u (u - \frac{1}{2})(1 - u) \quad x, y \in [0, 1], \quad t \in [0, 0.1] \tag{39} \]

with \( \epsilon = 0.01, \alpha = -10, \) and \( \gamma = 1. \) We used periodic boundary conditions with initial conditions given by

\[ u_0(x) = \sin(4\pi x) \cos(6\pi y). \]

For KSS–EPI, the low-frequency portion \( \mathbf{b}_l \) consists of all components with wave numbers \( |\omega| \leq N_s = 20. \)

For this test problem, both Leja interpolation and AKP are more efficient than Krylov–EPI and KSS–EPI, as we can see from Fig. 13. While Krylov–EPI is similar in efficiency to KSS–EPI at the smaller grid size \( (N = 50) \), we again observe the much greater increase in needed Krylov projection steps (Fig. 14) for Krylov–EPI as \( N \) increases. However, for this problem, Krylov–EPI does not need as many projection steps compared to, for example, Burgers’ equation, so the advantage
is much less pronounced. It should also be noted that compared to the previous test problems, none of the methods were particularly sensitive to the number of grid points in terms of number of iterations; this is due to the relative insignificance of the diffusive term, in view of the small coefficient of $\epsilon = 0.01$. As such, this is not the kind of problem for which KSS–EPI is expected to yield a significant advantage over other methods.

6.4. System of coupled PDEs

For our final test problem, we consider the 2-D Brusselator equation [12,21]

$$
\begin{align*}
  u_t &= 1 + uv^2 - 4u + \alpha \nabla^2 u, \quad x, y \in [0, 1], \quad t \in [0, 0.1], \\
  v_t &= 3u - u^2 v + \alpha \nabla^2 u,
\end{align*}
$$

with $\alpha = 0.2$, homogeneous Dirichlet boundary conditions, and initial data

$$
  u(x, y, 0) = \sin(6\pi x) \sin(7\pi y), \quad v(x, y, 0) = \sin(5\pi x).
$$

The $N_c$ value for the Brusselator equation is 30.

As we can see from Figs. 15 and 17, both Krylov–EPI and KSS–EPI yield similar accuracy, with a slight edge to Krylov–EPI, though for the finest grid with $N = 300$, this edge is offset by KSS–EPI’s greater efficiency. While KSS–EPI again needs fewer Krylov projection steps compared to Krylov–EPI as $N$ increases, the overall number of iterations (matrix–vector multiplications + FFTs) is actually greater for KSS–EPI in many cases (see Figs. 16 and 18), thus causing KSS–EPI to be only slightly more efficient than Krylov–EPI. However, it is worth noting that while the number of iterations needed by KSS–EPI was greater than for other methods in many of the experiments, it is still essentially insensitive to the number of grid points, as shown in Figs. 16 and 18, while all other methods exhibited a substantial increase.

6.5. Discussion of efficiency

The major components of the computational cost of KSS–EPI stem from Krylov projection that is applied to low-frequency parts, and FFTs that are applied to both the low- and high-frequency parts. Specifically, suppose the EPI method is of order $p$, and $q$ Krylov projection steps are needed for convergence of the low-frequency part. Then, the task of evaluating $\varphi(A \tau b)$ requires $q + p$ matrix–vector multiplications, $(p + 3)/2$ FFTs and 2 inverse FFTs if denoising is not used, and $q + p$ matrix–vector multiplications, $q + (p + 3)/2$ FFTs and $q + 2$ inverse FFTs if denoising is used. Denoising, therefore, is only worthwhile if the value of $q$ can be substantially reduced, as in the case of Burgers’ equation.

From these four test problems, we see that the performance of KSS–EPI, with or without denoising, varied considerably in comparison to Leja interpolation and adaptive Krylov projection. The following observations are worth making.

- Of the methods used, KSS–EPI is unique in that it requires Fourier transforms, which are the most computationally expensive tasks performed in the processing of the high-frequency portion of the solution. The number of transforms is related to the desired order of accuracy, as can be seen in (34). When not many matrix–vector products are needed to achieve convergence for Leja interpolation or standard or adaptive Krylov projection, the cost of these transforms becomes more significant, thus reducing or eliminating the advantage of a KSS–EPI approach. However, these transforms can be performed in parallel, even if only a very small number of processors are available.
Fig. 15. Relative error plotted against execution time for solving the Brusselator equation (40) using the 3rd-order EPI method (35). Matrix function–vector products are computed using KSS–EPI without denoising (magenta solid curves), Krylov–EPI (blue solid curves), AKP (red dashed curves), and LEJA (green dashed–dotted curves), on grids with \( N = 50, 150 \) and 300 points per dimension. Time steps used are \( \Delta t = (0.01)2^{-p} \), for \( p = 0, 1, 2, 3, 4 \). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Fig. 16. Average number of matrix–vector products, shown on a logarithmic scale, per matrix function–vector product evaluation for each method when solving the Brusselator equation (40) using the 3rd-order EPI method (35). For KSS–EPI, FFTs are also included. For each method, bars correspond to grid sizes of \( N = 50, 150, 300 \) points per dimension, from left to right. Left plot: \( \Delta t = 0.01 \). Right plot: \( \Delta t = 0.000625 \).
Fig. 17. Relative error plotted against execution time for solving the Brusselator equation (40) using the 5th-order EPI method (36). Matrix function–vector products are computed using KSS–EPI without denoising (magenta solid curves), Krylov–EPI (blue solid curves), AKP (red dashed curves), and LEJA (green dashed–dotted curves), on grids with $N = 50, 150$ and $300$ points per dimension. Time steps used are $\Delta t = (0.01)2^{-p}$, for $p = 0, 1, 2, 3, 4$. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Fig. 18. Average number of matrix–vector products, shown on a logarithmic scale, per matrix function–vector product evaluation for each method when solving the Brusselator equation (40) using the 5th-order EPI method (36). For KSS–EPI, FFTs are also included. For each method, bars correspond to grid sizes of $N = 50, 150, 300$ points per dimension, from left to right. Left plot: $\Delta t = 0.01$. Right plot: $\Delta t = 0.000625$. 
Table 2: Average ratio of computation time on the finest grid to computation time on the coarsest grid, across all time steps, for each of the methods and test problems featured in this section. The ‘order’ column indicates the order of the EPI method used. The ‘N^d’ column indicates the ratio of number of grid points, N^d, on the finest grid to number of grid points on the coarsest grid, where d is the number of spatial dimensions. For Burgers’ and ADR 2-D, KSS–EPI includes denoising.

<table>
<thead>
<tr>
<th>Equation</th>
<th>Order</th>
<th>N^d</th>
<th>Krylov–EPI</th>
<th>AKP</th>
<th>LEJA</th>
<th>RKSR</th>
<th>KSS–EPI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allen–Cahn</td>
<td>3</td>
<td>36</td>
<td>304</td>
<td>288</td>
<td>56</td>
<td>105</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>36</td>
<td>345</td>
<td>271</td>
<td>58</td>
<td>94</td>
<td>19</td>
</tr>
<tr>
<td>Burgers’</td>
<td>3</td>
<td>6</td>
<td>200</td>
<td>117</td>
<td>21</td>
<td>–</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>6</td>
<td>187</td>
<td>116</td>
<td>20</td>
<td>–</td>
<td>4</td>
</tr>
<tr>
<td>ADR 2-D</td>
<td>3</td>
<td>36</td>
<td>186</td>
<td>35</td>
<td>6</td>
<td>–</td>
<td>30</td>
</tr>
<tr>
<td>Brusselator</td>
<td>3</td>
<td>36</td>
<td>262</td>
<td>85</td>
<td>24</td>
<td>–</td>
<td>51</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>36</td>
<td>220</td>
<td>64</td>
<td>19</td>
<td>–</td>
<td>30</td>
</tr>
</tbody>
</table>

- In this paper, KSS was used in conjunction with standard Krylov projection for the low-frequency part of the solution. However, there is no reason why KSS could not be combined with an alternative approach to matrix function–vector products, such as Leja interpolation or adaptive Krylov projection. This will be explored in future work, as it requires examination of error estimation and stopping criteria for these methods in order to determine whether their convergence can be accelerated if it is known that the initial vector represents a smooth function, due to the elimination of high-frequency components.

Table 2 provides a measure of scalability of the various methods featured in this section. For each test problem and each method used, the computation time taken on the finest grid (N = 3,000 points for Burgers’ equation, N = 300 points per dimension for all other equations) is divided by the time taken on the coarsest grid (N = 500 points for Burgers’ equation, N = 50 points per dimension for all other equations). This ratio is then compared to the ratio of the number of points in each of these grids. It can be seen that KSS–EPI, with denoising for the advection-dominated problems, is the only method for which the computation time grows less than linearly for each test problem (although it grows more rapidly for the Brusselator equation with the 3rd-order EPI method, this only occurs for the largest time steps). For all other methods, such slow growth in computation time is achieved only for some of the test problems, or not at all, in the case of Krylov–EPI.

7. Conclusions

We have demonstrated that when solving stiff systems of nonlinear ODEs derived from PDEs, the growth in the computational cost that results from an increase in the number of grid points can be significantly reduced by performing a relatively low and grid-insensitive number of Krylov projection steps and FFTs on low- and high-frequency portions of the solution separately, instead of a number of Krylov projection steps on the entire solution that grows substantially with the number of grid points. By employing a componentwise approach to the computation of \( \varphi(\tau A)b \) as in KSS methods, in which each component of the solution with respect to an appropriate orthonormal basis is computed using an individualized approximation of the function \( \varphi \), the Krylov subspace dimension can be bounded independently of the grid size and instead determined by the desired temporal order of accuracy (for the high-frequency part) and by the number of Krylov projection steps required on a coarse grid (for the low-frequency part).

Future work on the combination of KSS and EPI methods will focus on the computation of low-frequency components of the solution. It will be necessary to develop an adaptive approach to determining the threshold \( N_c \) for retaining low-frequency components. Also, as mentioned in the previous section, combination with other methods for matrix function–vector products, including Leja interpolation and adaptive Krylov projection, will be investigated. Furthermore, it will be essential to generalize the approach demonstrated in this paper for rapidly estimating frequency-dependent nodes to low-frequency as well as high-frequency nodes, in view of the inherent coupling between high-frequency and low-frequency components. Given that these nodes tend to be smooth functions of the wave number, this generalization could be accomplished via interpolation.

As can be seen from the analysis in Section 3, the computation of the frequency-dependent nodes, while far more efficient and simplified compared to the approach from [16,18], is still labor-intensive (just analytically rather than numerically) and problem-specific. However, it can also be seen that the asymptotic behavior of these nodes is largely determined by the leading coefficient. Future work will need to include the developments of algorithms that can automatically produce viable frequency-dependent nodes directly from the coefficients of a differential operator, as well as the domain and boundary conditions. Patterns observed from the analysis in Section 3 yield a first step toward this goal.

The half-and-half split between the frequency-dependent and frequency-independent nodes is a consequence of the approximate decoupling of the block tridiagonal matrix \( T_k \) produced by block Lanczos, that has \( 2 \times 2 \) blocks. Since this results in two separate non-block Lanczos iterations, it is possible to terminate these iterations at different points, meaning that different splits are possible. It may be worthwhile to investigate such alternative splits in future work, but one drawback of such an uneven split is that it would not relate to any block Gaussian quadrature rule.
It is important to note that the decomposition of the block Gaussian quadrature nodes into frequency-dependent and frequency-independent nodes is not limited to cases in which Fourier decomposition is used. In [25], the same idea is used for PDEs defined on a disk, in which a Legendre polynomial expansion is used for the radial part of the solution. The decoupling of the eigenvalue problem for $T_k$ first observed in [23], and further exploited in this paper, occurs due to the rapid decay of the coefficients in whatever series expansion is used to represent the solution. Future work will include taking advantage of this behavior in order to generalize KSS methods to PDEs defined on non-rectangular domains.

Acknowledgements

The authors would like to thank Sarah Long for her assistance with the numerical experiments, as well as Chen Greif, Laurent Demanet and the anonymous referees for their helpful feedback which led to substantial improvement of this paper.

References