Summer 8-2015

Solution of Nonlinear Time-Dependent PDE Through Componentwise Approximation of Matrix Functions

Alexandru Cibotarica
University of Southern Mississippi

The University of Southern Mississippi

SOLUTION OF NONLINEAR TIME-DEPENDENT PDE THROUGH COMPONENTWISE APPROXIMATION OF MATRIX FUNCTIONS

by

Alexandru Cibotarica

Abstract of a Dissertation
Submitted to the Graduate School
of The University of Southern Mississippi
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy

August 2015

ABSTRACT

SOLUTION OF NONLINEAR TIME-DEPENDENT PDE THROUGH
COMPONENTWISE APPROXIMATION OF MATRIX FUNCTIONS

by Alexandru Cibotarica

August 2015

Exponential propagation iterative (EPI) methods provide an efficient approach to the solution of large stiff systems of ODE, compared to standard integrators. However, the bulk of the computational effort in these methods is due to products of matrix functions and vectors, which can become very costly at high resolution due to an increase in the number of Krylov projection steps needed to maintain accuracy. In this dissertation, it is proposed to modify EPI methods by using Krylov subspace spectral (KSS) methods, instead of standard Krylov projection methods, to compute products of matrix functions and vectors. This improvement allowed the benefits of KSS methods observed in linear PDE to be extended to the nonlinear case. Numerical experiments demonstrate that this modification causes the number of Krylov projection steps to become dramatically reduced, thus improving efficiency and scalability.

The University of Southern Mississippi

SOLUTION OF NONLINEAR TIME-DEPENDENT PDE THROUGH COMPONENTWISE APPROXIMATION OF MATRIX FUNCTIONS

by

Alexandru Cibotarica

A Dissertation
Submitted to the Graduate School
of The University of Southern Mississippi
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy

Approved:

Dr. James V. Lambers
Committee Chair

Dr. Ching-Shyang Chen

Dr. Haiyan Tian

Dr. Huiqing Zhu

Dr. Karen S. Coats
Dean of the Graduate School

August 2015

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. James V. Lambers, whose knowledge and wisdom guided me throughout the process of writing this dissertation. His efforts, patience and encouragements have helped me achive this acomplishment in my life. For this, I will be forever grateful and indebted to him. I would also like to thank the members of my committe for their valuable input and criticism, and my colleagues from the mathematics department for kind words and support. Last but not least, I want to thank my wife who's been the source of my insipiration and motivation.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Figure

LIST OF TABLES

Table

NOTATION AND GLOSSARY

General Usage and Terminology

The blackboard fonts are used to denote standard sets of numbers: $\mathbb{R}$ for the field of real numbers, $\mathbb{C}$ for the complex field. The italicized capital letters, $A, B, \cdots$ are used to denote matrices. Functions are denoted by the letter f or greek letters. Lower case letters such as i, j, k, l, m, n are used to denote indices, while lower case bold letters are used to denote vectors. Norms are typeset using double pairs of lines, e.g., $||\cdot||$.

Chapter 1

INTRODUCTION

The increase in computing power over the last decade has made it possible to use mathematical models with higher spatial resolution. However, these models have introduced greater stiffness into the system of ordinary differential equations (ODEs) that is obtained from the spatial discretization of a time-dependent partial differential equation (PDE). This stiffness poses problems for both explicit and implicit time-stepping methods. For explicit methods, the time step is severely restricted; i.e. a very small time step is required to be used, 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 [16].

Consider an autonomous, stiff system of ODE

$$\mathbf{y}' = F(\mathbf{y}), \quad \mathbf{y}(t_0) = \mathbf{y}_0, \tag{1.1}$$

such as one that would arise from spatial discretization of a PDE. Finding a solution to (1.1) requires the computation of matrix function-vector products of the form $\mathbf{w} = \varphi(A\tau)\mathbf{b}$, where φ is a smooth function, A is an ill-conditioned matrix, τ is a parameter determined by the time step, and $\mathbf{b}$ is a vector. Exponential propagation iterative (EPI) methods, introduced by Tokman et al. [14, 16], are designed to reduce the number of Krylov projection steps needed to compute such matrix function-vector products. One approach [14, 10] to computing $\mathbf{w}$ for a general nonsymmetric matrix A is to use Krylov projections obtained from the Arnoldi algorithm. The Arnoldi iteration produces an orthonormal basis $V_m = [\mathbf{v}_1, \ldots, \mathbf{v}_m]$, where $\mathbf{v}_1 = \mathbf{b}/\|\mathbf{b}\|_2$, of the Krylov subspace $K_m = \text{span}(\mathbf{b}, A\mathbf{b}, \ldots, A^{m-1}\mathbf{b})$ and an upper Hessenberg matrix H_m

$$H_m = V_m^T(A)V_m. \tag{1.2}$$

Then the matrix function-vector product can be approximated as follows

$$\varphi(A\tau)\mathbf{b} \approx V_m V_m^T f(A\tau)V_m V_m^T \mathbf{b}. \tag{1.3}$$

Using (1.2), we can rewrite (1.3) as

$$V_m^T \varphi(A\tau)V_m \approx \varphi(V_m^T A\tau V_m) \approx \varphi(H_m\tau), \tag{1.4}$$

producing the approximation

$$\varphi(A\tau)\mathbf{b} \approx V_m \varphi(H_m \tau) V_m^T \mathbf{b}. \tag{1.5}$$

Using the fact that $V_m^T \mathbf{b} = \|\mathbf{b}\|_2 \mathbf{e}_1$, we can express (1.5) as

$$\mathbf{w} = \varphi(A\tau)\mathbf{b} \approx \|\mathbf{b}\|_2 V_m \varphi(H_m \tau)\mathbf{e}_1, \tag{1.6}$$

where $\mathbf{e}_1 = \begin{bmatrix} 1 & 0 & \cdots & 0 \end{bmatrix}^T$. Since the matrix A arises from a stiff PDE, the eigenvalues of A are not clustered, which means that a large number of Arnoldi or Lanczos iterations might be required in order to obtain a good approximation of $\mathbf{w}$. In the case where (1.1) is a linear system of ODE, one way to solve this problem is by using an outer iteration

$$\mathbf{w}_j^{m+1} \approx e^{-A\Delta t}\mathbf{w}_j^m, \quad m = 0, 1, \ldots, \quad \mathbf{w}_j^0 = \mathbf{v}, \tag{1.7}$$

for some $\Delta t \ll t$. In this case, Δt must be chosen very small, which might not be practical.

The coupling of the components of the solution with different frequencies is the main difficulty that time-stepping methods have with stiffness. Another problem is that explicit time-stepping methods use a polynomial function and implicit time-stepping methods use a rational function to approximate all the components of $\varphi(A\tau)\mathbf{b}$, which cannot be done effectively on a large interval except at high degree, resulting in high computational expense. A solution to this problem is to use Krylov subspace spectral (KSS) methods [12, 18], that use an interpolating polynomial with frequency-dependent interpolating points to approximate the function φ, resulting in a component-wise approach to the problem. Thus, each Fourier coefficient of the solution is computed using an approximation of the solution operator that is tailored to that component. As a consequence, these methods demonstrate a high order of accuracy and stability like that of implicit methods.

The need to compute component-dependent nodes and weights of block Gausian quadrature rules [12, 19] accounts for most of the computational expense in KSS methods. Through an asymptotic analysis of the recursion coefficients produced by block Lanczos iteration [5], Lambers was able to obtain a much faster version of KSS [19], by using quadrature nodes that are prescribed based on estimates of the extremal nodes of these block Gaussian rules.

After performing a more thorough asymptotic analysis, we are able to estimate *all* of the block Gaussian nodes, not just the extremal ones. As a result, more efficient implementation of the central idea behind KSS methods is possible. Compared to traditional Krylov subspace-based approaches to computing $\varphi(A)\mathbf{b}$ [9, 10, 11, 25], KSS methods are not only highly accurate and stable, but also scalable with respect to the number of grid points used in the spatial discretization of the underlying PDE.

Untill now, KSS methods have been used mainly on linear PDE on n-dimensional boxes, for $n = 1, 2, 3$, with either periodic or homogeneous boundary conditions. A succesful implementation of KSS methods for nonlinear PDE was used by Guidotti, et al. [7] when a one-node KSS method was applied to nonlinear diffusion equations from image processing to obtain first-order accuracy in time. However, in order to achieve higher-order accuracy for nonlinear PDE, 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.

We present such a combination in this dissertation, for solving systems of ODE of the form (1.1) that are obtained through spatial discretization of nonlinear PDE, or systems of nonlinear PDE, defined on rectangular domains with periodic, homogeneous Dirichlet, or homogeneous Neumann boundary conditions. This method includes the following features:

- Instead of applying a Krylov projection method (e.g. see [9, 10, 11]) for computing approximations of expressions of the form $\mathbf{y} = \varphi(\tau A)\mathbf{b}$, where A is an $N \times N$ matrix, $\mathbf{b}$ is an N-vector, τ is a scaling factor derived from the time step, and φ is a smooth function, such a method is applied only to a low-frequency approximation of $\mathbf{b}$, in order to avoid the larger number of iterations that these methods typically incur at higher spatial resolution. Furthermore, 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 $\mathbf{b}$, application of $\varphi(\tau A)$ is performed using a KSS method, as described in [15]. In this particular KSS method, each Fourier component of the output vector $\mathbf{y}$ 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 [12, 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 [19], 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 [15] is that it combines the accuracy of the approach of [12, 18] with the efficiency of the approach of [19].

  In [15], formulas for the nodes were given for a 1-D, self-adjoint second-order operator

with periodic boundary conditions. In this dissertation, 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.

- 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 $\mathbf{b}$. This dissertation 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 this dissertation is as follows. Chapter 2 gives a description of KSS methods. Chapter 3 discusses the acceleration of the KSS methods based on a thourough asymptotic analysis of the recursion coefficients for different cases. Chapter 4 provides a brief description of the EPI methods, and shows the high-frequency oscillations that can occur when using standard Krylov projection with an EPI method. Also, Chapter 4 describes how KSS and EPI methods are combined. Numerical results are presented in Chapter 5, and conclusions are stated in Chapter 6.

Chapter 2

Krylov Subspace Spectral Methods

To review the essential aspects of KSS methods, as first described in [12], 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 computed 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

$$\hat{u}(\omega, t_{n+1}) = \left\langle \frac{1}{\sqrt{2\pi}} e^{i\omega x}, e^{-L\Delta t} \tilde{u}(x, t_n) \right\rangle, \tag{2.1}$$

where ω is an integer representing the wave number, $\langle \cdot, \cdot \rangle$ 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 (2.1), we obtain the following bilinear form

$$\mathbf{u}^T f(A)\mathbf{v}, \tag{2.2}$$

where $\mathbf{u} = \frac{1}{\sqrt{2\pi}} e^{i\omega x}$ and $\mathbf{v} = \tilde{u}(x, t_n)$ are N-vectors, $A = L_N$ is an $N \times N$ symmetric positive definite matrix that comes from discretizing the operator L, and $f(\lambda) = e^{-\lambda t}$.

The matrix A 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 (2.1)

$$\mathbf{u}^T f(A)\mathbf{v} = \sum_{j=1}^{N} f(\lambda_j) \mathbf{u}^T \mathbf{q}_j \mathbf{q}_j^T \mathbf{v}. \tag{2.3}$$

As mentioned by Golub and Meurant in [4], (2.1) can also be viewed as a Rieman-Stieltjes integral

$$\mathbf{u}^T f(A)\mathbf{v} = \int_a^b f(\lambda)\, d\alpha(\lambda), \tag{2.4}$$

where

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

We can approximate the integral in (2.4) using Gaussian quadrature rules, where the nodes and weights are obtained using the Lanczos algorithm applied to A with initial vectors $\mathbf{u}$ and $\mathbf{v}$ [4].

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 2×2 matrix integral

$$\left[\begin{array}{cc} \mathbf{u} & \mathbf{v} \end{array} \right]^H f(A) \left[\begin{array}{cc} \mathbf{u} & \mathbf{v} \end{array} \right]. \tag{2.5}$$

The expression in (2.5) can be regarded as a matrix-valued Riemann-Stieltjes integral

$$\int_a^b f(\lambda)\, d\mu(\lambda) = \left[\begin{array}{cc} \mathbf{u}^H f(A)\mathbf{u} & \mathbf{u}^H f(A)\mathbf{v} \\ \mathbf{v}^H f(A)\mathbf{u} & \mathbf{v}^H f(A)\mathbf{v} \end{array} \right]$$

where $\mu(\lambda)$ is a 2×2 matrix, each entry of which is a measure of the form $\alpha(\lambda)$.

We use the most general K-node quadrature formula, as described in [4], to obtain an approximation for (2.4) of the form

$$\int_a^b f(\lambda)\, d\mu(\lambda) = \sum_{j=1}^{2K} f(\lambda_j)\mathbf{v}_j\mathbf{v}_j^H + error, \tag{2.6}$$

where, for each j, λ_j is a scalar and $\mathbf{v}_j$ is a 2-vector. Each node λ_j is an eigenvalue of the matrix

$$\mathcal{T}_K = \left[\begin{array}{cccc} M_1 & B_1^H & & \\ B_1 & M_2 & B_2^H & \\ & \ddots & \ddots & \ddots \\ & & B_{K-1} & M_K \end{array} \right], \tag{2.7}$$

which is a block-tridiagonal matrix of order $2K$. The vector $\mathbf{v}_j$ 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 [5]:

$X_0 = 0$, $R_0 = [\mathbf{u}, \mathbf{v}]$, $R_0 = X_1 B_0$ (QR factorization)
for $n = 1, 2, \ldots, K$
  $V = AX_n$
  $M_n = X_n^H V$
  $R_n = V - X_{n-1}B_{n-1}^H - X_n M_n$
  $R_n = X_{n+1}B_n$ (QR factorization)
end

The block KSS method starts by defining

$$R_0(\omega) = \left[\begin{array}{cc} \hat{\mathbf{e}}_\omega & \mathbf{u}^n \end{array} \right], \tag{2.8}$$

where $\hat{\mathbf{e}}_\omega$ is a discretization of $\frac{1}{\sqrt{2\pi}}e^{i\omega x}$ and $\mathbf{u}^n$ is the computed solution at time t_n. The QR factorization of $R_0(\omega)$ yields $R_0(\omega) = X_1(\omega)B_0(\omega)$, with

$$X_1(\omega) = \begin{bmatrix} \hat{\mathbf{e}}_\omega & \frac{\mathbf{u}_\omega^n}{\|\mathbf{u}_\omega^n\|_2} \end{bmatrix} \tag{2.9}$$

and

$$B_0(\omega) = \begin{bmatrix} 1 & \hat{\mathbf{e}}_\omega^H \mathbf{u}^n \\ 0 & \|\mathbf{u}_\omega^n\|_2 \end{bmatrix},$$

where

$$\mathbf{u}_\omega^n = \mathbf{u}^n - \hat{\mathbf{e}}_\omega \hat{\mathbf{e}}_\omega^H \mathbf{u}^n. \tag{2.10}$$

Then, block Lanczos iteration is applied to the discretized operator L_N with initial block $X_1(\omega)$, producing a block tridiagonal matrix $\mathcal{T}_K(\omega)$ of the form (2.7), where each entry is a function of ω. Then, each Fourier coefficient of the solution at time t_{n+1} can be expressed as

$$[\hat{\mathbf{u}}^{n+1}]_\omega = \left[B_0^H E_{12}^H e^{-\mathcal{T}_K(\omega)\Delta t} E_{12} B_0 \right]_{12}, \quad E_{12} = \begin{bmatrix} \mathbf{e}_1 & \mathbf{e}_2 \end{bmatrix}. \tag{2.11}$$

This algorithm has temporal accuracy $O(\Delta t^{2K-1})$ for parabolic problems [12]. 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 [12, 18].

It may seem that KSS methods appear to be prohibitively expensive, due to the computation of a large number of Krylov subspaces, when compared to the Krylov subspace methods such as those in [9, 10, 11, 25]. However, the latter methods require a number of Arnoldi or Lanczos iterations that increase with the number of grid points in order to preserve the same level of accuracy. At the same time, the Krylov subspaces generated by KSS methods are closely related by the wave number ω, which allows elimination of redundant computations [19]. In the next chapter we will see how we can get individual approximations for each component through asymptotic analysis, to make KSS methods even more efficient.

Chapter 3

Asymptotic Analysis of Block Lanczos Iteration

The main idea in KSS methods is to compute each Fourier component of the solution using an approximation that is optimal for that component. In particular, each component of the solution uses its own polynomial approximation of $S(L_N;\Delta t) = e^{-L_N \Delta t}$, where the function S is based on the solution operator of the PDE

$$\mathbf{u}_t + L\mathbf{u} = 0,$$

and L_N is the discretization of the spatial differential operator. These polynomial approximations are obtained by interpolation of the function $S(\lambda;\Delta t)$ at selected nodes for each component. Then, the computed solution has the form [19]

$$\mathbf{u}^{n+1} = S(L_N;\Delta t)\mathbf{u}^n = \sum_{j=0}^{2K} D_j(\Delta t) L_N^j \mathbf{u}^n,$$

where $D_j(\Delta t)$ is a matrix that is diagonal in the chosen basis. The diagonal entries are the coefficients of these interpolating polynomials in the monomial basis, with each row corresponding to a particular component. The interpolation points of the original block KSS method [12, 18] were obtained by performing block Lanczos iteration and then diagonalizing a $2K \times 2K$ matrix–for each component. In this chapter, we develop a much faster way of obtaining interpolation points, by studying the behavior of block Lanczos in the limit as $|\omega| \to \infty$, where ω is the wave number.

3.1 The Block Case

The block Lanczos algorithm is as follows:

$X_0 = 0$, $R_0 = [\mathbf{u}, \mathbf{v}]$, $R_0 = X_1 B_0$ (QR factorization)

for $n = 1, 2, \ldots, K$
  $V = AX_n$
  $M_n = X_n^H V$
  if $n < K$

$R_n = V - X_{n-1}B_{n-1}^H - X_nM_n$
$R_n = X_{n+1}B_n$ (QR factorization)
end
end

Let $\mathbf{u}^n$ be a discretization of the solution at time $t_n = n\Delta t$ on an uniform N-point grid. For the initial $R_0 = \left[\begin{array}{cc} e^{i\omega\mathbf{x}}/\sqrt{2\pi} & \mathbf{u}^n \end{array}\right]$, we start the first iteration of the block Lanczos algorithm by finding the QR-factorization of R_0:

$$\left[\begin{array}{cc} \mathbf{a}_1 & \mathbf{a}_2 \end{array}\right] = \left[\begin{array}{cc} \mathbf{q}_1 & \mathbf{q}_2 \end{array}\right] \left[\begin{array}{cc} r_{11} & r_{12} \\ 0 & r_{22} \end{array}\right], \tag{3.1}$$

where

$$r_{11} = \|\mathbf{a}_1\|_2 = \left\| \frac{e^{i\omega\mathbf{x}}}{\sqrt{2\pi}} \right\|_2 = 1, \tag{3.2}$$

$$\mathbf{q}_1 = \frac{\mathbf{a}_1}{r_{11}} = \frac{e^{i\omega\mathbf{x}}}{\sqrt{2\pi}}, \tag{3.3}$$

$$r_{12} = \langle \mathbf{q}_1, \mathbf{a}_2 \rangle \approx \frac{1}{\sqrt{2\pi}} \int_0^{2\pi} e^{-i\omega\mathbf{x}} \mathbf{u}^n dx = \hat{u^n}(\omega), \tag{3.4}$$

where $\hat{u^n}(\omega)$ is a coefficient of the Fourier interpolant of $\mathbf{u}^n$,

$$r_{22} = \|\mathbf{a}_2 - r_{12}\mathbf{q}_1\|_2 = \left\| \mathbf{u}^n - \frac{1}{\sqrt{2\pi}} \hat{u^n}(\omega) e^{i\omega\mathbf{x}} \right\|_2, \tag{3.5}$$

$$\mathbf{q}_2 = \frac{\mathbf{a}_2 - r_{12}\mathbf{q}_1}{r_{22}} = \frac{\mathbf{u}^n - \frac{1}{\sqrt{2\pi}} \hat{u^n}(\omega) e^{i\omega\mathbf{x}}}{\left\| \mathbf{u}^n - \frac{1}{\sqrt{2\pi}} \hat{u^n}(\omega) e^{i\omega\mathbf{x}} \right\|_2}. \tag{3.6}$$

If we let $\mathbf{u}^n - \frac{1}{\sqrt{2\pi}} \hat{u^n}(\omega) e^{i\omega\mathbf{x}} = \mathbf{u}_\omega^n$, then (3.6) can be written as

$$\mathbf{q}_2 = \frac{\mathbf{u}_\omega^n}{\|\mathbf{u}_\omega^n\|_2}. \tag{3.7}$$

Thus, the QR-factorization of R_0 yields the two matrices

$$X_1 = \left[\begin{array}{cc} \frac{e^{i\omega\mathbf{x}}}{\sqrt{2\pi}} & \frac{\mathbf{u}_\omega^n}{\|\mathbf{u}_\omega^n\|_2} \end{array}\right] \text{ and } B_0 = \left[\begin{array}{cc} 1 & \hat{u^n}(\omega) \\ 0 & \|\hat{u^n}(\omega)\|_2 \end{array}\right]. \tag{3.8}$$

The next step is to find M_1

$$M_1 = X_1^H L_N X_1, \tag{3.9}$$

where $Lu = pu_{xx} + q(x)u$, and p is a constant. Using the value of X_1 from (3.8) into (3.9) yields

$$M_1 = \begin{bmatrix} -\omega^2 p + \bar{q} & \widehat{\frac{L_N \mathbf{u}_\omega^n}{\|\mathbf{u}_\omega^n\|_2}} \\ \widehat{\frac{L_N \mathbf{u}_\omega^n}{\|\mathbf{u}_\omega^n\|_2}} & R(L_N, \mathbf{u}_\omega^n) \end{bmatrix}, \tag{3.10}$$

where $R(L_N, \mathbf{u}_\omega^n) = \dfrac{\langle \mathbf{u}_\omega^n, L_N \mathbf{u}_\omega^n \rangle}{\langle \mathbf{u}_\omega^n, \mathbf{u}_\omega^n \rangle}$ is the Rayleigh quotient of the operator L_N and $\mathbf{u}_\omega^n$ and $\bar{q}$ is the average of the function q. As $|\omega|$ increases, the Fourier coefficients of a function go to zero as long as the function is at least piecewise continuous, and therefore the non-diagonal entries of M_1 become negligible, i.e.,

$$M_1 \approx \begin{bmatrix} -\omega^2 p + \bar{q} & 0 \\ 0 & R(L_N, \mathbf{u}_\omega^n) \end{bmatrix}. \tag{3.11}$$

In the next step of the block Lanczos algorithm, we obtain R_1 as follows

$$R_1 = L_N X_1 - X_1 M_1 = \begin{bmatrix} \dfrac{1}{\sqrt{2\pi}} e^{i\omega \mathbf{x}} \tilde{q} & \dfrac{L_N \mathbf{u}_\omega^n}{\|\mathbf{u}_\omega^n\|_2} - \mathbf{u}_\omega^n R(L_N, \mathbf{u}_\omega^n) \end{bmatrix}. \tag{3.12}$$

To obtain X_2, we repeat the process of the block Lanczos algorithm and perform the QR-factorization of R_1

$$r_{11} = \|\tilde{q}\|_2, \tag{3.13}$$

$$\mathbf{q}_1 = \frac{\tilde{q} e^{i\omega \mathbf{x}}}{\sqrt{2\pi}\|\tilde{q}\|_2}, \tag{3.14}$$

where $\tilde{q} = q - \bar{q}$,

$$\begin{aligned} r_{12} &\approx \frac{1}{\sqrt{2\pi}\|\tilde{q}\|_2} \int_0^{2\pi} e^{-i\omega_x} \tilde{q} \left(\frac{L_N \mathbf{u}_\omega^n}{\|\mathbf{u}_\omega^n\|_2} - \mathbf{u}_\omega^n R(L_N, \mathbf{u}_\omega^n) \right) dx \\ &\approx \frac{\widehat{\tilde{q} \left(\dfrac{L_N \mathbf{u}_\omega^n}{\|\mathbf{u}_\omega^n\|_2} - \mathbf{u}_\omega^n R(L_N, \mathbf{u}_\omega^n) \right)}}{\|\tilde{q}\|_2}, \end{aligned} \tag{3.15}$$

where the multiplication of vectors is performed componentwise,

$$\begin{aligned}\mathbf{q}_2 &= \frac{\dfrac{L_N\mathbf{u}_\omega^n}{\|\mathbf{u}_\omega^n\|_2} - \mathbf{u}_\omega^n R(L_N,\mathbf{u}_\omega^n) - \dfrac{\widehat{\tilde{q}e^{i\omega\mathbf{x}}\tilde{q}\left(\dfrac{L_N\mathbf{u}_\omega^n}{\|\mathbf{u}_\omega^n\|_2} - \mathbf{u}_\omega^n R(L_N,\mathbf{u}_\omega^n)\right)}}{\sqrt{2\pi}\|\tilde{q}\|_2^2}}{\left\| \dfrac{L_N\mathbf{u}_\omega^n}{\|\mathbf{u}_\omega^n\|_2} - \mathbf{u}_\omega^n R(L_N,\mathbf{u}_\omega^n) - \dfrac{\widehat{\tilde{q}e^{i\omega\mathbf{x}}\tilde{q}\left(\dfrac{L_N\mathbf{u}_\omega^n}{\|\mathbf{u}_\omega^n\|_2} - \mathbf{u}_\omega^n R(L_N,\mathbf{u}_\omega^n)\right)}}{\sqrt{2\pi}\|\tilde{q}\|_2^2} \right\|_2} \\ &= \frac{r_{1,\omega}}{\|r_{1,\omega}\|_2}. \end{aligned} \tag{3.16}$$

Therefore, X_2 is given by

$$X_2 = \begin{bmatrix} \dfrac{\tilde{q}e^{i\omega\mathbf{x}}}{\sqrt{2\pi}\|\tilde{q}\|_2} & \dfrac{r_{1,\omega}}{\|r_{1,\omega}\|_2} \end{bmatrix}. \tag{3.17}$$

We then find M_2 as follows

$$M_2 = X_2^H L_N X_2 = \begin{bmatrix} -\omega^2 p + \bar{q}_{\tilde{q}^2} & \dfrac{\widehat{\tilde{q}L_N r_{1,\omega}}}{\|\tilde{q}\|_2\|r_{1,\omega}\|_2} \\ \overline{\dfrac{\widehat{\tilde{q}L_N r_{1,\omega}}}{\|\tilde{q}\|_2\|r_{1,\omega}\|_2}} & R(L_N, r_{1,\omega}) \end{bmatrix}, \tag{3.18}$$

where $\bar{q}_{\tilde{q}^2}$ is the average value of $q(x)$ with respect to the weight function $\tilde{q}^2(x)$ and is given by $\bar{q}_{\tilde{q}^2} = \dfrac{\int_0^{2\pi} q(x)\tilde{q}^2(x)dx}{\int_0^{2\pi}\tilde{q}^2(x)dx}$.

Since the off-diagonal entries of M_2 are Fourier coefficients, as $|\omega|$ increases, the Fourier coefficients of a function go to zero. Therefore M_2, just as M_1, becomes approximately diagonal, i.e.,

$$M_2 \approx \begin{bmatrix} -\omega^2 p + \bar{q}_{\tilde{q}^2} & 0 \\ 0 & R(L_N, r_{1,\omega}) \end{bmatrix}. \tag{3.19}$$

Continuing this process, it can be seen that every (nonzero) off-diagonal entry of M_j or B_j, for $j = 1,2,\ldots$, is a Fourier coefficient of some function that is a differential operator applied to u. Therefore, as long as the Fourier coefficients of $\mathbf{u}^n$ decay to zero at a sufficiently high rate as $|\omega| \to \infty$, these off-diagonal entries will also decay to zero.

In the case where the leading coefficient $p(x)$ of L is *not* constant, the blocks B_0 and M_1 are the same as in the case where p is constant, except that the $(1,1)$ entry of M_1 is $\bar{p}\omega^2 + \bar{q}$. We then have

$$R_1 = \left[\begin{array}{cc} \omega^2 \tilde{\mathbf{p}} \hat{\mathbf{e}}_\omega & \dfrac{L_N \mathbf{u}_\omega}{\|\mathbf{u}_\omega\|_2} - R(L_N, \mathbf{u}_\omega) \dfrac{\mathbf{u}_\omega}{\|\mathbf{u}_\omega\|_2} \end{array} \right] + \text{lower order terms}, \tag{3.20}$$

where $\tilde{\mathbf{p}}$ contains the values of $\tilde{p}(x)$ from $x_j = j\Delta x$, $j = 0, 1, \ldots, N-1$. It can be seen that as in the case of constant p, when we compute the QR factorization $R_1 = X_2 B_1$, the $(1,2)$ entry of B_1, modulo lower-order terms, will be a Fourier coefficient of

$$\mathbf{w}_1 = \tilde{\mathbf{p}} \left(\frac{L_N \mathbf{u}_\omega}{\|\mathbf{u}_\omega\|_2} - R(L_N, \mathbf{u}_\omega) \frac{\mathbf{u}_\omega}{\|\mathbf{u}_\omega\|_2} \right),$$

which will approach 0 as $|\omega| \to \infty$. Continuing this process reveals that the behavior is the same as in the case where p is constant.

It follows that in this high-frequency limit, the block tridiagonal matrix $\mathcal{T}_K$ produced by block Lanczos applied to R_0 as defined above 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. Therefore, by reordering the rows and columns of $\mathcal{T}_K$ in such a way that odd-numbered and even-numbered rows and columns are grouped together, we find that the eigenvalue problem for this matrix *decouples*, and the block Gaussian quadrature nodes can be obtained by computing the eigenvalues of these smaller, tridiagonal matrices [15]. For finite ω, we can then use non-block Lanczos to at least estimate the true block Gaussian quadrature nodes.

3.2 The Non-Block Case

The decoupling observed in the preceding discussion reveals that 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 $\mathbf{u}^n$, the computed solution, as is done in standard Krylov projection methods such as those described in [9, 10, 11]. These nodes will be referred to as $frequency-independent$ nodes. To estimate the other half of the nodes, we perform an asymptotic analysis of Lanczos iteration applied to L_N with initial vector $\hat{\mathbf{e}}_\omega$; these are called $frequency-dependent$ nodes. The algorithm for Lanczos iteration is given as follows:

$\beta_0 = 0$, $\mathbf{m}_0 = \mathbf{0}$, $\mathbf{m}_1 = \mathbf{u}/\|\mathbf{u}\|_2$
for $n = 1, 2, \ldots, K$
    $\mathbf{v}_n = A\mathbf{m}_n$
    $\alpha_n = \mathbf{m}_n^H \mathbf{v}_n$

$\mathbf{v}_n = \mathbf{v}_n - \beta_{n-1}\mathbf{m}_{n-1} - \alpha_n\mathbf{m}_n$
$\beta_n = \|\mathbf{v}_n\|_2$
$\mathbf{m}_{n+1} = \mathbf{v}_n/\beta_n$
end

Thus, if $\mathbf{u} = \frac{1}{\sqrt{2\pi}}e^{i\omega\mathbf{x}}$ and $Lu = p\mathbf{u}_{xx} + q(x)\mathbf{u}$, where p is a constant, then the first iteration of the Lanczos algorithm is as follows

$$\mathbf{m} = \frac{\mathbf{u}}{\|\mathbf{u}\|} = \frac{\frac{1}{\sqrt{2\pi}}e^{i\omega\mathbf{x}}}{\|\frac{1}{\sqrt{2\pi}}e^{i\omega\mathbf{x}}\|_2} = \frac{e^{i\omega\mathbf{x}}}{\|e^{i\omega\mathbf{x}}\|} = e^{i\omega\mathbf{x}}. \tag{3.21}$$

$$\mathbf{v} = L\mathbf{m} = p(e^{i\omega\mathbf{x}})_{xx} + qe^{i\omega\mathbf{x}} = p(-\omega^2 e^{i\omega\mathbf{x}}) + qe^{i\omega\mathbf{x}} = e^{i\omega\mathbf{x}}(-\omega^2 p + q). \tag{3.22}$$

$$\alpha_1 = \mathbf{m}^H\mathbf{v} \approx \int_0^{2\pi} e^{-i\omega x}e^{i\omega\mathbf{x}}(-\omega^2 p + q)dx = -\omega^2 p + \bar{q}, \tag{3.23}$$

where $\bar{q} = \frac{1}{2\pi}\int_0^{2\pi} q\,dx$ is the average value of the function $q(x)$ over the interval $[0, 2\pi]$. In the next step, ew update $\mathbf{v}$ and obtain β_1 as follows

$$\begin{aligned} \mathbf{v} = \mathbf{v} - \beta_0\mathbf{l} - \alpha_1\mathbf{m} &= e^{i\omega\mathbf{x}}(-\omega^2 p + q) - e^{i\omega\mathbf{x}}(-\omega^2 p + \bar{q}) \\ &= e^{i\omega\mathbf{x}}(q - \bar{q}) = e^{i\omega\mathbf{x}}\tilde{q}, \end{aligned} \tag{3.24}$$

where $\tilde{q} = q - \bar{q}$.

$$\beta_1 = \|e^{i\omega\mathbf{x}}\tilde{q}\|_2 = \|\tilde{q}\|_2, \tag{3.25}$$

We then continue by updating the values of $\mathbf{l}$ and $\mathbf{m}$

$$\mathbf{l} = e^{i\omega\mathbf{x}}, \tag{3.26}$$

$$\mathbf{m} = \frac{e^{i\omega\mathbf{x}}\tilde{q}}{\|\tilde{q}\|_2}. \tag{3.27}$$

The second iteration of the Lanczos algorithm yields

$$\mathbf{v} = \frac{1}{\|\tilde{q}\|_2}[p(-\omega^2 e^{i\omega\mathbf{x}}\tilde{q} + 2i\omega e^{i\omega\mathbf{x}}q_x + e^{i\omega\mathbf{x}}q_{xx}) + e^{i\omega\mathbf{x}}q\tilde{q}]. \tag{3.28}$$

$$\begin{aligned}\alpha_2 &\approx \frac{1}{\|\tilde{q}\|_2^2}\int_0^{2\pi}(-\omega^2 p\tilde{q}^2 + 2i\omega p\tilde{q}q_x + p\tilde{q}q_{xx} + q\tilde{q})dx \\ &\approx -\omega^2 p + \bar{q}_{\tilde{q}^2} - p\frac{\|q_x\|_2^2}{\|\tilde{q}\|_2^2},\end{aligned} \tag{3.29}$$

where $\int_0^{2\pi} 2ip\omega\tilde{q}q_x dx = 0$ and $\bar{q}_{\tilde{q}^2} = \int_0^{2\pi} q\tilde{q}^2 dx$ is the average value of $q(x)$ with respect to $\tilde{q}^2$ on the interval $[0, 2\pi]$. The dominant term in (3.29) is $-\omega^2 p$. The other terms are insignificant compared to this term, so if we drop the lower order terms, 3.29 can be rewritten as

$$\alpha_2 \approx -\omega^2 p. \tag{3.30}$$

We update $\mathbf{v}$ and obtain

$$\begin{aligned}\mathbf{v} &= \frac{1}{\|\tilde{q}\|_2}[p(-\omega^2 e^{i\omega\mathbf{x}}\tilde{q} + 2i\omega e^{i\omega\mathbf{x}}q_x + e^{i\omega\mathbf{x}}q_{xx}) + q\tilde{q}e^{i\omega\mathbf{x}}] \\ &\quad -e^{i\omega\mathbf{x}}\|\tilde{q}\|_2 - (-\omega^2 p)\frac{e^{i\omega x\tilde{q}}}{\|\tilde{q}\|_2}.\end{aligned} \tag{3.31}$$

Since the ω^2-terms in (3.31) cancel and we are only interested in the highest order ω-terms, eq. (3.31) can be written as

$$\mathbf{v} = \frac{2ip\omega e^{i\omega\mathbf{x}}q_x}{\|\tilde{q}\|_2}. \tag{3.32}$$

Then, we update β_2 to obtain

$$\beta_2 = \left\| \frac{2ip\omega e^{i\omega\mathbf{x}}q_x}{\|\tilde{q}\|_2} \right\|_2 = \frac{2p\omega\|q_x\|_2}{\|\tilde{q}\|_2}. \tag{3.33}$$

Updating $\mathbf{l}$ and $\mathbf{m}$ we obtain

$$\mathbf{l} = \frac{e^{i\omega\mathbf{x}}\tilde{q}}{\|\tilde{q}\|_2} \tag{3.34}$$

$$\mathbf{m} = \frac{ie^{i\omega\mathbf{x}}q_x}{\|q_x\|_2}. \tag{3.35}$$

The third iteration of the Lanczos algorithm yields

$$\mathbf{v} = i\frac{1}{\|q_x\|_2}[p(-\omega^2 e^{i\omega\mathbf{x}}q_x + 2i\omega e^{i\omega\mathbf{x}}q_{xx} + e^{i\omega\mathbf{x}} + e^{i\omega\mathbf{x}}q_{xxx}) + e^{i\omega\mathbf{x}}qq_x], \tag{3.36}$$

$$\alpha_3 \approx \frac{1}{\|q_x\|_2^2}\int_0^{2\pi}[p(-\omega^2 q_x^2 e^{i\omega\mathbf{x}} + 2i\omega q_x q_{xx} + q_x q_{xxx}) + q q_x^2]dx. \tag{3.37}$$

Simplifying (3.37) we obtain

$$\alpha_3 \approx -\omega^2 p + \bar{q}_{q_x^2} + p\frac{\|q_{xx}\|_2^2}{\|q_x\|_2^2}. \tag{3.38}$$

Similarly to (3.29), we drop the lower order terms in (3.38) to obtain

$$\alpha_3 \approx -\omega^2 p. \tag{3.39}$$

Thus, we obtain the following matrix with recursion coefficients as functions of the wave number ω:

$$\begin{bmatrix} \alpha_1 & \overline{\beta_1} & 0 \\ \beta_1 & \alpha_2 & \overline{\beta_2} \\ 0 & \beta_2 & \alpha_3 \end{bmatrix} \approx \begin{bmatrix} p\omega^2 & \|\tilde{\mathbf{q}}\|_2 & 0 \\ \|\tilde{\mathbf{q}}\|_2 & p\omega^2 & 2p|\omega|\|\mathbf{q}_x\|_2/\|\tilde{\mathbf{q}}\|_2 \\ 0 & 2p|\omega|\|\mathbf{q}_x\|_2/\|\tilde{\mathbf{q}}\|_2 & p\omega^2 \end{bmatrix}.$$

It follows that the frequency-dependent nodes can easily be estimated as

$$\lambda_{1,\omega} = p\omega^2, \quad \lambda_{i,\omega} = p\omega^2 \pm \sqrt{\beta_1^2 + \beta_2^2}, \quad i = 2,3. \tag{3.40}$$

We now consider the case of homogeneous Neumann boundary conditions, with the same operator. Let $\mathbf{x}$ be a vector of uniformly spaced grid points on $[0, 2\pi)$. Given the initial values $\beta_0 = 0$, $\mathbf{l} = 0$, $\mathbf{u} = \cos(\omega x)$, and $Lu = pu_{xx} + q(x)u$, we first find $\mathbf{m}$

$$\mathbf{m} = \frac{\mathbf{u}}{\|\mathbf{u}\|_2} = \frac{\cos(\omega x)}{\|\cos(\omega x)\|_2} = \frac{\cos(\omega x)}{\sqrt{\pi}}. \tag{3.41}$$

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

$$\mathbf{v} = L\mathbf{m} = \frac{1}{\sqrt{\pi}}[-p\omega^2\cos(\omega x) + q\cos(\omega x)]. \tag{3.42}$$

$$\alpha_1 \approx \mathbf{m}^T\mathbf{v} = \frac{1}{\pi}\int_0^{2\pi}(-\omega^2 p\cos^2(\omega x) + q\cos^2(\omega x))dx \tag{3.43}$$

Since $\int_0^{2\pi}\cos(\omega x)^2 dx = \pi$, then (3.43) simplifies to

$$\alpha_1 \approx -\omega^2 p + \bar{q}. \tag{3.44}$$

The next step in the Lanczos algorithm is to update $\mathbf{v}$

$$\begin{aligned} \mathbf{v} &= \mathbf{v} - \beta_0\mathbf{l} - \alpha_1\mathbf{m} \\ &= \frac{1}{\sqrt{\pi}}[-p\omega^2\cos(\omega x) + q\cos(\omega x)] + (\omega^2 p + \bar{q})\frac{\cos(\omega x)}{\sqrt{\pi}} \\ &= (q - \bar{q})\frac{\cos(\omega x)}{\sqrt{\pi}} = \frac{\tilde{q}\cos(\omega x)}{\sqrt{\pi}}. \end{aligned} \tag{3.45}$$

From (3.45), we obtain the new β

$$\beta_1 = \left\| \frac{\tilde{q}\cos(\omega x)}{\sqrt{\pi}} \right\|_2 = \frac{\|\tilde{q}\cos(\omega x)\|_2}{\sqrt{\pi}}. \tag{3.46}$$

To conclude the first iteration of the Lanczos algorithm, we update the values of $\mathbf{l}$ and $\mathbf{m}$

$$\mathbf{l} = \mathbf{m} = \frac{\cos(\omega x)}{\sqrt{\pi}} \tag{3.47}$$

$$\mathbf{m} = \frac{\mathbf{v}}{\beta_1} = \frac{\tilde{q}\cos(\omega x)}{\|\tilde{q}\cos(\omega x)\|_2}. \tag{3.48}$$

We continue with the second iteration of the Lanczos algorithm and obtain the next $\mathbf{v}$

$$\begin{aligned} \mathbf{v} &= \frac{1}{\|\tilde{q}\cos(\omega x)\|_2}[p(-\omega^2\tilde{q}\cos(\omega x) - 2\omega q_x \sin(\omega x) + q_{xx}\cos(\omega x)) \\ &\quad + q\tilde{q}\cos(\omega x)]. \end{aligned} \tag{3.49}$$

We proceed with obtaining the next α value

$$\begin{aligned} \alpha_2 &\approx \frac{1}{\pi\|\tilde{q}\cos(\omega x)\|_2^2}\int_0^{2\pi} \tilde{q}\cos(\omega x)[p(-\omega^2\tilde{q}\cos(\omega x) - 2\omega q_x \sin(\omega x) \\ &\quad + q_{xx}\cos(\omega x)) + q\tilde{q}\cos(\omega x)]dx \end{aligned} \tag{3.50}$$

If we drop the lower order ω-terms, then α_2 simplifies to

$$\alpha_2 \approx -p\omega^2. \tag{3.51}$$

The next step in the Lanczos algorith is to update $\mathbf{v}$

$$\begin{aligned} \mathbf{v} &= \frac{1}{\|\tilde{q}\cos(\omega x)\|_2}[p(-\omega^2\tilde{q}\cos(\omega x) - 2\omega q_x \sin(\omega x) + q_{xx}\cos(\omega x)) \\ &\quad + q\tilde{q}\cos(\omega x)] - \frac{\|\tilde{q}\cos(\omega x)\|_2\cos(\omega x)}{\pi} + p\omega^2\frac{\tilde{q}\cos(\omega x)}{\|\tilde{q}\cos(\omega x)\|_2} \end{aligned} \tag{3.52}$$

Dropping the lower order ω-terms is (3.52) simplifies to

$$\mathbf{v} = \frac{-2p\omega q_x \sin(\omega x)}{\|\tilde{q}\cos(\omega x)\|_2}. \tag{3.53}$$

As a result, the new β is given by

$$\beta_2 = \frac{2p\omega\|q_x \sin(\omega x)\|_2}{\|\tilde{q}\cos(\omega x)\|_2}. \tag{3.54}$$

Finally, we update $\mathbf{l}$ and $\mathbf{m}$

$$\mathbf{l} = \frac{\tilde{q}\cos(\omega x)}{\|\tilde{q}\cos(\omega x)\|_2} \tag{3.55}$$

$$\mathbf{m} = -\frac{q_x \sin(\omega x)}{\|q_x \sin(\omega x)\|_2}. \tag{3.56}$$

The third step of the Lanczos algorith proceeds as follows

$$\begin{aligned} \mathbf{v} &= -\frac{1}{\|q_x \sin(\omega x)\|_2}[p(-\omega^2 q_x \sin(\omega x) + 2\omega q_{xx}\cos(\omega x) \\ &\quad + q_{xxx}\sin(\omega x)) + q q_x \sin(\omega x)]. \end{aligned} \tag{3.57}$$

The next α is given by

$$\begin{aligned} \alpha_3 &\approx \frac{1}{\|q_x \sin(\omega x)\|_2^2}\int_0^{2\pi} q_x \sin(\omega_x)[p(-\omega^2 q_x \sin(\omega x) + 2\omega q_{xx}\cos(\omega x) \\ &\quad + q_{xxx}\sin(\omega x)) + q q_x \sin(\omega x)]dx. \end{aligned} \tag{3.58}$$

Dropping the lower order ω-terms in (3.58) yields

$$\alpha_3 \approx -p\omega^2. \tag{3.59}$$

After neglecting lower-order terms, we have the following matrix of recursion coefficients as functions of the wave number ω:

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

Then, we obtain the approximate quadrature nodes using (3.40).

3.3 The 2D Case

We now generalize to a two-dimensional domain $[0,2\pi]^2$, with periodic boundary conditions in both directions. Given the initial values $\beta_0 = 0$, $\mathbf{m} = \mathbf{0}$, $\mathbf{u} = e^{i\omega\cdot\mathbf{x}}$, and the differential operator $L = -p\Delta + q(x,y)$, we first find

$$\mathbf{m}_1 = \frac{\frac{1}{2\pi}e^{i\omega\cdot\mathbf{x}}}{\frac{1}{2\pi}\|e^{i\omega\cdot\mathbf{x}}\|_2} = e^{i\omega\cdot\mathbf{x}}. \tag{3.60}$$

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

$$\mathbf{v}_1 = L\mathbf{m}_1 = -p\|\omega\|_2^2 e^{i\omega\cdot\mathbf{x}} + q e^{i\omega\cdot\mathbf{x}} \tag{3.61}$$

$$\alpha_1 \approx \mathbf{m}_1^H \mathbf{v}_1 = \int_0^{2\pi}\int_0^{2\pi} [-p\|\omega\|_2^2 + q]\, dy\, dx = -p\|\omega\|_2^2 + \bar{q}, \tag{3.62}$$

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

$$\begin{aligned}\mathbf{v}_1 &= \mathbf{v}_1 - \beta_0\mathbf{l}_1 - \alpha_1\mathbf{m}_1 \\ &\approx -\|\omega\|_2^2 p e^{i\omega\cdot\mathbf{x}} + qe^{i\omega\cdot\mathbf{x}} - (-p\|\omega\|_2^2 + \bar{q})e^{i\omega\cdot\mathbf{x}} \\ &\approx (q-\bar{q})e^{i\omega\cdot\mathbf{x}} = \tilde{q}e^{i\omega\cdot\mathbf{x}}, \end{aligned} \tag{3.63}$$

where $\mathbf{l}_1 = \mathbf{m}_0$. From (3.63) we obtain β_1

$$\beta_1 = \|\mathbf{v}_1\|_2 \approx \|\tilde{q}\|_2. \tag{3.64}$$

Then, we obtain $\mathbf{l}_2$ and $\mathbf{m}_2$ to finish the first iteration of the Lanczos algorithm

$$\mathbf{l}_2 = \mathbf{m}_1 = e^{i\omega\cdot\mathbf{x}} \tag{3.65}$$

$$\mathbf{m}_2 \approx \frac{\mathbf{v}}{\beta_1} = \frac{\tilde{q}e^{i\omega\cdot\mathbf{x}}}{\|\tilde{q}\|_2}. \tag{3.66}$$

The second iteration of the Lanczos algorithm yields

$$\mathbf{v}_2 \approx \frac{1}{\|\tilde{q}\|_2}[p(-\|\omega\|_2^2\tilde{q} + 2i\omega\cdot\nabla q + \Delta q)e^{i\omega\cdot\mathbf{x}} + q\tilde{q}e^{i\omega\cdot\mathbf{x}}]. \tag{3.67}$$

Then, we find the next value of α

$$\begin{aligned}\alpha_2 &\approx \frac{1}{\|\tilde{q}\|_2^2}\int_0^{2\pi}\tilde{q}[p(-\|\omega\|_2^2\tilde{q} + 2i\omega\cdot\nabla q + \Delta q) + q\tilde{q}]ds \\ &\approx -p\|\omega\|_2^2 + \bar{q}_{\tilde{q}^2} + \frac{1}{\|\tilde{q}\|_2^2}\int_0^{2\pi}\tilde{q}[2ip\omega\cdot\nabla q + \Delta q]ds, \end{aligned} \tag{3.68}$$

where $\bar{q}_{\tilde{q}^2}$ is the average value of the function $q(x,y)$ on the rectangle $[0,2\pi]^2$ with respect to $\tilde{q}^2$. If we drop the lower order ω-terms, then we can rewrite (3.68) as

$$\alpha_2 \approx -p\|\omega\|_2^2 \tag{3.69}$$

We update the value of $\mathbf{v}_2$ to obtain

$$\begin{aligned}\mathbf{v}_2 &\approx \frac{1}{\|\tilde{q}\|_2}[p(-\|\omega\|_2^2\tilde{q} + 2i\omega\cdot\nabla q + \Delta q)e^{i\omega\cdot\mathbf{x}} + q\tilde{q}e^{i\omega\cdot\mathbf{x}}] \\ &\quad + p\|\omega\|_2^2\frac{\tilde{q}e^{i\omega\cdot\mathbf{x}}}{\|\tilde{q}\|_2} - \|\tilde{q}\|_2 e^{i\omega\cdot\mathbf{x}}. \end{aligned} \tag{3.70}$$

Dropping the lower-order terms in (3.70) yields

$$\mathbf{v}_2 \approx \frac{1}{\|\tilde{q}\|_2}2ip\omega\cdot\nabla q e^{\omega\cdot\vec{x}}. \tag{3.71}$$

From (3.71), we obtain the next value of β

$$\beta_2 \approx \frac{2p\|\omega\cdot\nabla q\|_2}{\|\tilde{q}\|_2}. \tag{3.72}$$

We then obtain $\mathbf{l}_3$ and $\mathbf{m}_3$ to finish the second iteration of the Lanczos algorithm

$$\mathbf{l}_3 \approx \frac{\tilde{q}e^{i\omega\cdot\mathbf{x}}}{\|\tilde{q}\|_2} \tag{3.73}$$

$$\mathbf{m}_3 \approx \frac{ie^{i\omega\cdot\mathbf{x}}\omega\cdot\nabla q}{\|\omega\cdot\nabla q\|_2}. \tag{3.74}$$

The third iteration of the Lanczos algorithm yields

$$\begin{aligned}\mathbf{v}_3 \approx\ & \frac{1}{\|\omega\cdot\nabla q\|_2}[p(-\|\omega\|_2^2 i\omega\cdot\nabla q - 2\omega\cdot\nabla(\omega\cdot\nabla q) + \\ & +\Delta(i\omega\cdot\nabla q))e^{i\omega\cdot\mathbf{x}} + iq\omega\cdot\nabla q e^{i\omega\cdot\mathbf{x}}].\end{aligned} \tag{3.75}$$

The next value of α is

$$\begin{aligned}\alpha_3 \approx\ & \frac{1}{\|\omega\cdot\nabla q\|_2^2}\int_0^{2\pi}\omega\cdot\nabla q[p(-\|\omega\|_2^2\omega\cdot\nabla q - 2\omega\cdot\nabla(i\omega\cdot\nabla q) - \\ & -i\Delta(i\omega\cdot\nabla q))e^{i\omega\cdot\mathbf{x}} + q\omega\cdot\nabla q e^{i\omega\cdot\mathbf{x}}]ds.\end{aligned} \tag{3.76}$$

Dropping the lower-order ω-terms in (3.76) yields

$$\alpha_3 \approx -p\|\omega\|_2^2. \tag{3.77}$$

The matrix of recursion coefficients as functions of wave number ω is given by

$$\begin{bmatrix} \alpha_1 & \overline{\beta_1} & 0 \\ \beta_1 & \alpha_2 & \overline{\beta_2} \\ 0 & \beta_2 & \alpha_3 \end{bmatrix} \approx \begin{bmatrix} -p\|\omega\|_2^2 & \|\tilde{q}\|_2 & 0 \\ \|\tilde{q}\|_2 & -p\|\omega\|_2^2 & \frac{2p\|\omega\cdot\nabla q\|_2}{\|\tilde{q}\|_2} \\ 0 & \frac{2p\|\omega\cdot\nabla q\|_2}{\|\tilde{q}\|_2} & -p\|\omega\|_2^2 \end{bmatrix}.$$

Then, as in (3.40), we obtain the estimated quadrature nodes

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

When the differential operator includes advection terms, the same approach can be used for node estimation, except that it is best to use Arnoldi iteration instead of unsymmetric Lanczos, and, of course, the resulting nodes can have imaginary parts.

3.3.1 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 $\varphi(\tau A)\mathbf{b}$ instead of unsymmetric Lanczos iteration, which can suffer from "serious breakdown" [6]. The Arnoldi algorithm proceeds as follows, with a given matrix A and initial vector $\mathbf{z}_0$:

$\mathbf{v}_1 = \mathbf{z}_0/\|\mathbf{z}_0\|_2$
for $j = 1,2,\ldots$
    $\mathbf{z}_j = A\mathbf{v}_j$
    for $k = 1,2,\ldots,j$
        $h_{kj} = \mathbf{v}_k^H \mathbf{z}_j$
        $\mathbf{z}_j = \mathbf{z}_j - h_{kj}\mathbf{v}_k$
    end
    $h_{j+1,j} = \|\mathbf{z}_j\|_2$
    $\mathbf{v}_{j+1} = \mathbf{z}_j/h_{j+1,j}$
end

This iteration produces an upper Hessenberg matrix H_m and matrix V_m with orthonormal columns such that

$$AV_m = V_m H_m + h_{m+1,m}\mathbf{z}_{m+1}\mathbf{e}_m^H.$$

By analogy with (2.11), to approximate $\mathbf{u} = \varphi(\tau A)\mathbf{b}$, we could compute each discrete Fourier component $[\hat{\mathbf{u}}]_\omega$ of $\mathbf{u}$, corresponding to wave number ω, by applying *block* Arnoldi iteration [26] to A, with initial block $R_0(\omega)$ as defined in (2.8), and after m iterations that yield a block upper Hessenberg matrix $H_m(\omega)$, we obtain the approximation

$$[\hat{\mathbf{u}}]_\omega = \left[B_0^H E_{12}^H \varphi(\tau H_m(\omega)) E_{12} B_0\right]_{12}, \tag{3.78}$$

where B_0 and E_{12} are as defined in Chapter 2. However, 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 $\mathbf{b}$. 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 $\hat{\mathbf{e}}_\omega$, 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.

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 5.2.5). We will apply Arnoldi iteration to a matrix A that is a spatial discretization of the differential operator

$$Lu = p\Delta u + q(u_{x_1} + u_{x_2}) + \phi(x_1,x_2)u,$$

where p and q are constants. We use the initial vector $\mathbf{z}_0 = e^{i\omega\cdot\mathbf{x}}$, where ω contains the wave numbers and $\mathbf{x}$ is a vector of equally-spaced grid points. We begin the first iteration of the Arnoldi algorithm by finding $\mathbf{v}_1$:

$$\mathbf{v}_1 = \frac{e^{i\omega\cdot\mathbf{x}}}{\|e^{i\omega\cdot\mathbf{x}}\|_2} = \frac{1}{2\pi}e^{i\omega\cdot\mathbf{x}}. \tag{3.79}$$

We then have

$$\mathbf{z}_1 = L\mathbf{v}_1 \approx \frac{1}{2\pi}(-\|\omega\|_2^2 p e^{i\omega\cdot\mathbf{x}} + qie^{i\omega\cdot\mathbf{x}}(\omega_1+\omega_2) + \phi e^{i\omega\cdot\mathbf{x}}). \tag{3.80}$$

In the next step, we compute

$$\begin{aligned} h_{11} &= \mathbf{v}_1^H\mathbf{z}_1 \\ &\approx \frac{1}{4\pi^2}\left(\int_0^{2\pi}\int_0^{2\pi} e^{-i\omega\cdot\mathbf{x}}e^{i\omega\cdot\mathbf{x}}(-\|\omega\|_2^2 p + qi(\omega_1+\omega_2)+\phi)\,dx_1\,dx_2\right) \\ &\approx -\|\omega\|_2^2 p + qi(\omega_1+\omega_2) + \bar{\phi}, \end{aligned} \tag{3.81}$$

where $\bar{\phi}$ is the average of the function $\phi(x_1,x_2)$ over the rectangle $[0,2\pi]^2$. We then update $\mathbf{z}_1$ as follows:

$$\mathbf{z}_1 = \mathbf{z}_1 - h_{11}\mathbf{v}_1 \approx \frac{1}{2\pi}e^{i\omega\cdot\mathbf{x}}(\phi-\bar{\phi}) \approx \frac{1}{2\pi}e^{i\omega\cdot\mathbf{x}}\tilde{\phi}, \quad \tilde{\phi} \equiv \phi - \bar{\phi}. \tag{3.82}$$

Next, we compute

$$h_{21} = \|\mathbf{z}_1\|_2 \approx \frac{1}{2\pi}\|\tilde{\phi}\|_2. \tag{3.83}$$

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

$$\mathbf{v}_2 = \frac{\mathbf{z}_1}{h_{21}} = \frac{e^{i\omega\cdot\mathbf{x}}\tilde{\phi}}{\|\tilde{\phi}\|_2}. \tag{3.84}$$

The second iteration of the Arnoldi algorithm starts by computing $\mathbf{z_2}$

$$\begin{aligned} \mathbf{z_2} &= L\mathbf{v_2} \\ &= \frac{1}{\|\tilde{\phi}\|_2}[pe^{i\omega\cdot\mathbf{x}}(-\|\omega\|_2^2\tilde{\phi} + 2i\omega\cdot\nabla\phi + \Delta\phi) + qie^{i\omega\cdot\mathbf{x}}\tilde{\phi}(\omega_1+\omega_2) \\ &+ qe^{i\omega\cdot\mathbf{x}}(\phi_{x_1}+\phi_{x_2}) + e^{i\omega\cdot\mathbf{x}}\phi\tilde{\phi}]. \end{aligned} \tag{3.85}$$

We continue by finding h_{12} and h_{22}

$$\begin{aligned} h_{12} &= \mathbf{v}_1^H \mathbf{z}_2 \\ &= \langle \mathbf{u_1}, \mathbf{z_2} \rangle \\ &= \frac{1}{2\pi \|\tilde{\phi}\|_2} \int_0^{2\pi} \int_0^{2\pi} [p(-\|\omega\|_2^2 \tilde{\phi} + 2i\omega \cdot \nabla\phi + \Delta\phi) + qi\tilde{\phi}(\omega_1 + \omega_2) \\ &+ q(\phi_{x_1} + \phi_{x_2}) + \phi\tilde{\phi}] dx_1 dx_2. \end{aligned} \tag{3.86}$$

All the terms in (3.86) will be equal to zero except for $\int_0^{2\pi} \int_0^{2\pi} \phi\tilde{\phi} dx_1 dx_2 = \int_0^{2\pi} \int_0^{2\pi} (\tilde{\phi}^2 + \tilde{\phi}\bar{\phi}) dx_1 dx_2 = \int_0^{2\pi} \int_0^{2\pi} \tilde{\phi}^2 dx_1 dx_2 = \|\tilde{\phi}\|_2^2$. Therefore, equation (3.86) becomes

$$h_{12} \approx \frac{\|\tilde{\phi}\|_2}{2\pi}. \tag{3.87}$$

$$\begin{aligned} h_{22} &= \mathbf{v}_2^H \mathbf{z}_2 \\ &= \frac{1}{\|\tilde{\phi}\|_2^2} \int_0^{2\pi} \int_0^{2\pi} [p(-\|\omega\|_2^2 \tilde{\phi}^2 + 2i\tilde{\phi}\omega \cdot \nabla\phi + \tilde{\phi}\Delta\phi) + qi\tilde{\phi}^2(\omega_1 + \omega_2) \\ &\quad + q\tilde{\phi}(\phi_{x_1} + \phi_{x_2}) + \phi\tilde{\phi}^2] dx_1 dx_2 \\ &\approx -p\|\omega\|_2^2 - \frac{p(\|\phi_{x_1}\|_2^2 + \|\phi_{x_2}\|_2^2)}{\|\tilde{\phi}\|_2^2} + qi(\omega_1 + \omega_2) + \bar{\phi}_{\tilde{\phi}^2}, \end{aligned} \tag{3.88}$$

where $\bar{\phi}_{\tilde{\phi}^2}$ is the average of the function ϕ, with weight function $\tilde{\phi}^2$, on the rectangle $[0, 2\pi]^2$. In the next step we update $\mathbf{z}_2$ as follows:

$$\begin{aligned} \mathbf{z}_2 &= L\mathbf{v}_2 \\ &\approx \frac{1}{\|\tilde{\phi}\|_2} [pe^{i\omega\cdot\mathbf{x}}(2i\omega \cdot \nabla\phi + \Delta\phi) + qe^{i\omega\cdot\mathbf{x}}(\phi_{x_1} + \phi_{x_2}) + e^{i\omega\cdot\mathbf{x}}\phi\tilde{\phi}] \\ &\quad + \frac{\|\tilde{\phi}\|_2 e^{i\omega\cdot\mathbf{x}}}{4\pi^2} - \frac{e^{i\omega\cdot\mathbf{x}}\tilde{\phi}}{\|\tilde{\phi}\|_2} \left(-\frac{p(\|\phi_{x_1}\|_2^2 + \|\phi_{x_2}\|_2^2)}{\|\tilde{\phi}\|_2^2} + \bar{\phi}_{\tilde{\phi}^2} \right). \end{aligned} \tag{3.89}$$

Dropping the lower order ω-terms in (3.89) yields

$$\mathbf{z}_2 \approx \frac{2ip\omega \cdot \nabla\phi e^{i\omega\cdot\mathbf{x}}}{\|\tilde{\phi}\|_2}. \tag{3.90}$$

Continuing, we then compute

$$\begin{aligned}
h_{32} &= \|\mathbf{z}_2\|_2 \approx \frac{2p\|\omega\cdot\nabla\phi\|_2}{\|\tilde{\phi}\|_2},\\
h_{13} &= \mathbf{v}_1^H\mathbf{z}_3\\
&\approx \frac{i}{2\pi\|\omega\cdot\nabla\phi\|_2}\int_0^{2\pi}\int_0^{2\pi}[p(-\|\omega\|_2^2(\omega\cdot\nabla\phi)+2i\omega\cdot\nabla(\omega\cdot\nabla\phi)\\
&\quad +\Delta(\omega\cdot\nabla\phi))+qi((\omega\cdot\nabla\phi)_{x_1}+(\omega\cdot\nabla\phi)_{x_2})\\
&\quad +qi(\omega\cdot\nabla\phi)(\omega_1+\omega_2)+\phi(\omega\cdot\nabla\phi)]\,dx_1\,dx_2\\
&\approx 0, \qquad (3.91)\\
h_{23} &= \mathbf{v}_2^H\mathbf{z}_3\\
&\approx \frac{2p\|\omega\cdot\nabla\phi\|_2}{\|\tilde{\phi}\|_2}, \qquad (3.92)\\
h_{33} &= \mathbf{v}_3^H\mathbf{z}_3\\
&\approx -p\|\omega\|_2^2-\frac{p(\|(\omega\cdot\nabla\phi)_{x_1}\|_2^2+\|(\omega\cdot\nabla\phi)_{x_2}\|_2^2)}{\|\omega\cdot\nabla\phi\|_2^2}+qi(\omega_1+\omega_2)+\bar{\phi}_{(\omega\cdot\nabla\phi)^2}.
\end{aligned}$$

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_2, for a 3rd-order method, by

$$\lambda_{1,2} = h_{11} \pm h_{12}, \tag{3.93}$$

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

$$\lambda_1 = h_{11}, \quad \lambda_{2,3} = h_{11} \pm \sqrt{h_{12}^2+h_{23}^2}. \tag{3.94}$$

Next, we consider a system of coupled PDE, based on a linearization of the 2-D Brusselator (see Section 5.2.6). The system is

$$\begin{aligned}
u_t &= \alpha\Delta u + pu + \phi v,\\
v_t &= \alpha\Delta v + qv + \psi u,
\end{aligned}$$

with appropriate initial conditions and periodic boundary conditions. It is assumed that α is a constant, and all other coefficients are variable. Following an approach described in [20] for constructing basis functions for a coupled system of PDE, it can be shown that at high frequencies, these basis functions can be approximated by $(e^{i\omega\cdot\mathbf{x}},0)$ and $(0,e^{i\omega\cdot\mathbf{x}})$, for each wave number ω.

Therefore, given the operator L and the initial vector $\mathbf{z}_0$ defined by

$$L = \begin{bmatrix}\alpha\Delta+p & \phi\\ \psi & \alpha\Delta+q,\end{bmatrix}, \qquad \mathbf{z}_0 = \begin{bmatrix}e^{i\omega\cdot\mathbf{x}}\\ 0\end{bmatrix}, \begin{bmatrix}0\\ e^{i\omega\cdot\mathbf{x}}\end{bmatrix}, \tag{3.95}$$

with a matrix A representing a spatial discretization of L, we apply the Arnoldi algorithm to A with the initial vector $\mathbf{z}_0$ being the first vector given in (3.95); the iteration with the second vector is similar. The first step of the first outer iteration of the Arnoldi algorithm is to find $\mathbf{v}_1$:

$$\mathbf{v}_1 = \frac{\mathbf{z}_0}{\|\mathbf{z}_0\|_2} = \begin{bmatrix} \dfrac{e^{i\omega\cdot\mathbf{x}}}{2\pi} \\ 0 \end{bmatrix}. \tag{3.96}$$

We then find $\mathbf{z}_1$ by applying the matrix A to $\mathbf{v}_1$:

$$\mathbf{z}_1 = L\mathbf{v}_1 \approx \begin{bmatrix} \dfrac{(-\alpha\|\omega\|_2^2 + p)e^{i\omega\cdot\mathbf{x}}}{2\pi} \\ \dfrac{\psi e^{i\omega\cdot\mathbf{x}}}{2\pi} \end{bmatrix}. \tag{3.97}$$

The next steps are to compute h_{11} and update $\mathbf{z}_1$:

$$h_{11} = \mathbf{v}_1^H\mathbf{z}_1 \approx \frac{1}{4\pi}\int_0^{2\pi}\int_0^{2\pi}(-\alpha\|\omega\|_2^2 + p)\,dx_1\,dx_2 \approx -\alpha\|\omega\|_2^2 + \bar{p}. \tag{3.98}$$

$$\mathbf{z}_1 = \mathbf{z}_1 - h_{11}\mathbf{v}_1 \approx \begin{bmatrix} \dfrac{\tilde{p}e^{i\omega\cdot\mathbf{x}}}{2\pi} \\ \dfrac{\psi e^{i\omega\cdot\mathbf{x}}}{2\pi} \end{bmatrix}. \tag{3.99}$$

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

$$h_{21} = \|\mathbf{z}_1\|_2 \approx \frac{\sqrt{\|\tilde{p}\|_2^2 + \|\psi\|_2^2}}{2\pi}. \tag{3.100}$$

and normalizing our updated $\mathbf{z}_1$ to obtain $\mathbf{v}_2$.

$$\mathbf{v}_2 = \frac{\mathbf{z}_1}{h_{21}} \approx \begin{bmatrix} \dfrac{\tilde{p}e^{i\omega\cdot\mathbf{x}}}{\sqrt{\|\tilde{p}\|_2^2 + \|\psi\|_2^2}} \\ \dfrac{\psi e^{i\omega\cdot\mathbf{x}}}{\sqrt{\|\tilde{p}\|_2^2 + \|\psi\|_2^2}} \end{bmatrix}. \tag{3.101}$$

Continuing with the second and third outer iterations, we obtain

$$
\begin{aligned}
h_{12} &= \mathbf{v}_1^H \mathbf{z}_2 \approx \frac{\|\tilde{p}\|_2^2 + \overline{\phi\psi}}{2\pi\sqrt{\|\tilde{p}\|_2^2 + \|\psi\|_2^2}}, \\
h_{22} &= \mathbf{v}_2^H \mathbf{z}_2 \approx -\alpha\|\omega\|_2^2 + \frac{-\alpha(\|p_{x_1}\|_2^2 + \|p_{x_2}\|_2^2 + \|\psi_{x_1}\|_2^2 + \|\psi_{x_2}\|_2^2) + \bar{p}_{\tilde{p}^2} + \bar{q}_{\psi^2} + \overline{\tilde{p}\phi\psi} + \bar{\tilde{p}}_{\psi^2}}{\|\tilde{p}\|_2^2 + \|\psi\|_2^2}, \\
h_{32} &= \|\mathbf{z}_2\|_2 \approx \frac{2\alpha\sqrt{\|\omega\cdot\nabla p\|_2^2 + \|\omega\cdot\nabla\psi\|_2^2}}{\sqrt{\|\tilde{p}\|_2^2 + \|\psi\|_2^2}}, \\
h_{13} &\approx \frac{i\overline{\phi(\omega\cdot\nabla\psi)}}{2\pi\sqrt{\|\omega\cdot\nabla p\|_2^2 + \|\omega\cdot\nabla\psi\|_2^2}}, \\
h_{23} &\approx \frac{2\alpha(\|\omega\cdot\nabla p\|_2^2 + \|\omega\cdot\nabla\psi\|_2^2) + i\overline{\tilde{p}\phi(\omega\cdot\nabla\psi)} + i\overline{q\psi(\omega\cdot\nabla\psi)} + i\overline{\psi^2(\omega\cdot\nabla p)}}{\sqrt{\|\tilde{p}\|_2^2 + \|\psi\|_2^2}\sqrt{\|\omega\cdot\nabla p\|_2^2 + \|\omega\cdot\nabla\psi\|_2^2}}, \\
h_{33} &\approx -\alpha\|\omega\|_2^2 - \frac{\alpha[\|(\omega\cdot\nabla p)_{x_1}\|_2^2 + \|(\omega\cdot\nabla p)_{x_2}\|_2^2 + \|(\omega\cdot\nabla\psi)_{x_1}\|_2^2 + \|(\omega\cdot\nabla\psi)_{x_2}\|_2^2]}{\|\omega\cdot\nabla p\|_2^2 + \|\omega\cdot\nabla\psi\|_2^2} \\
&\quad + \frac{\bar{p}_{(\omega\cdot\nabla p)^2} + \overline{\phi(\omega\cdot\nabla p)(\omega\cdot\nabla\psi)} + \bar{q}_{(\omega\cdot\nabla\psi)^2} + \overline{\psi(\omega\cdot\nabla p)(\omega\cdot\nabla\psi)}}{\|\omega\cdot\nabla p\|_2^2 + \|\omega\cdot\nabla\psi\|_2^2}
\end{aligned}
$$

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 $\|\omega\|_2$, h_{32} and h_{23} are equal, as are all of the diagonal entries. By neglecting the lowest-order entries, which are h_{12}, h_{21} and h_{13}, we obtain the estimated nodes

$$\lambda_1 = h_{11}, \quad \lambda_{2,3} = h_{11} \pm \sqrt{h_{23}h_{32}} \tag{3.102}$$

for a 5th-order method, whereas for a 3rd-order method, our estimated nodes are

$$\lambda_{1,2} = h_{11} \pm \sqrt{h_{12}h_{21}}. \tag{3.103}$$

Chapter 4

KSS-EPI Methods

4.1 EPI Methods

We now give a brief description of exponential propagation iterative (EPI) methods, introduced by Tokman [16]. Suppose that we have a nonlinear autonomous system of ODE of the form (1.1). Then we use the Taylor expansion of $F(\mathbf{y}(t))$ around $\mathbf{y}(t_n)$ to obtain

$$\frac{d\mathbf{y}}{dt} = F(\mathbf{y}(t_n)) + A_n(\mathbf{y}(t) - \mathbf{y}(t_n)) + R(\mathbf{y}(t)), \tag{4.1}$$

where $A_n = \dfrac{dF(\mathbf{y}(t_n))}{d\mathbf{y}}$ is the Jacobian of $F(\mathbf{y}(t))$ and $R(\mathbf{y}(t))$ is the nonlinear remainder function. Using an integrating factor $e^{-A_n t}$ and integrating (4.1) over the time interval $[t_n, t_{n+1}]$ gives us the integral form of (1.1)

$$\mathbf{y}(t_{n+1}) = \mathbf{y}(t_n) + [e^{A_n \Delta t} - I]A_n^{-1} F(\mathbf{y}(t_n)) + \int_{t_n}^{t_{n+1}} e^{A_n(t_{n+1}-\tau)} R(\mathbf{y}(\tau))\, d\tau. \tag{4.2}$$

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

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

$$\varphi(A\tau)\mathbf{b} \approx \|\mathbf{b}\|_2 V_m \varphi(H_m \tau)\mathbf{e}_1, \tag{4.3}$$

where H_m is an upper Hessenberg matrix which is given by $H_m = V_m^T A V_m$, 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 $K_m(A, \mathbf{b})$, which can be obtained using the Arnoldi algorithm [6]. The accuracy of the approximation in (4.3) depends on the number of the Krylov vectors constructed, the eigenvalues of A, the magnitude of τ, and φ.

In this dissertation we will look at three EPI methods. The first is a 3rd-order, 2-stage EPI method [16]

$$\begin{aligned} Y_1 &= \mathbf{y}_n + \frac{1}{3} h a_{11} \varphi_1\left(\frac{1}{3}hA\right) F(\mathbf{y}_n), \\ \mathbf{y}_{n+1} &= \mathbf{y}_n + h\varphi_1(hA)F(\mathbf{y}_n) + 3hb_1\varphi_2(hA)[F(Y_1) - F(\mathbf{y}_n) - A(Y_1 - \mathbf{y}_n)], \end{aligned} \tag{4.4}$$

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

$$R(Y_1) = F(Y_1) - F(\mathbf{y}_n) - A(Y_1 - \mathbf{y}_n).$$

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 4th-order, 3-stage EPI method [16]

$$\begin{aligned} Y_1 &= \mathbf{y}_n + \frac{1}{3}ha_{11}\varphi_1\left(\frac{1}{3}hA\right)F(\mathbf{y}_n), \\ Y_2 &= \mathbf{y}_n + \frac{2}{3}ha_{21}\varphi_1\left(\frac{2}{3}hA\right)F(\mathbf{y}_n), \\ \mathbf{y}_{n+1} &= \mathbf{y}_n + h\varphi_1(hA)F(\mathbf{y}_n) + 3hb_1\varphi_2(hA)R(Y_1) + \\ &\quad \frac{3}{2}hb_2\varphi_3(hA)[-2R(Y_1) + R(Y_2)], \end{aligned} \tag{4.5}$$

where

$$a_{11} = \frac{27c_3}{4}, \quad a_{21} = \frac{9s}{8c_3}, \quad b_1 = \frac{32(54-s^2)c_4}{729}, \quad b_2 = \frac{128c_4}{81},$$

with $s = \sqrt{30}$ and

$$c_1 = 54 - 3s^2 + 2s^3, \quad c_2 = s^2 + 18, \quad c_3 = \frac{c_1}{c_2}, \quad c_4 = \frac{c_3^2}{c_2}.$$

The third is a 5th-order, 3 stage EPI method [17]

$$\begin{aligned} Y_1 &= \mathbf{y}_n + ha_{11}\psi_1(g_{11}hA)F(\mathbf{y}_n), \\ Y_2 &= \mathbf{y}_n + ha_{21}\psi_1(g_{21}hA)F(\mathbf{y}_n) + ha_{22}\psi_2(g_{22}hA)R(Y_1), \\ \mathbf{y}_{n+1} &= \mathbf{y}_n + hb_1\psi_1(g_{31}hA)F(\mathbf{y}_n) + hb_2\psi_2(g_{32}hA)R(Y_1) + \\ &\quad hb_3\psi_3(g_{33}hA)[-2R(Y_1) + R(Y_2)], \end{aligned} \tag{4.6}$$

where

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

and the coefficients are

$$a_{11} = g_{11} = 0.41657015580651858694, \quad a_{21} = g_{21} = 0.86246743701274574979,$$

$$a_{22} = 1.32931146991722972036, \quad g_{22} = 0.5, \quad g_{31} = 1.0, \quad g_{32} = 0.730416157608327661916,$$

$$g_{33} = 0.325076967060782773227, \quad b_1 = 1.0, \quad b_2 = 1.15468303405015770322,$$

$$b_3 = 0.30931492086655796815, \quad p_{11} = p_{33} = 1, \quad p_{21} = p_{22} = p_{31} = \frac{2}{3}, \quad p_{32} = \frac{1}{2}.$$

For larger matrices A and larger time steps τ, 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 $\mathbf{b}$ represents a very smooth function. This is illustrated in Figure 4.1. For this reason, in the numerical

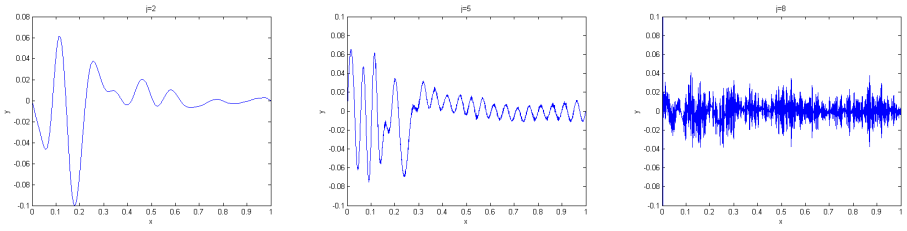


Figure 4.1: Columns of V_m from (4.3) generated by Arnoldi iteration applied to the matrix from Burgers' equation (see Section 5.2.3)

experiments performed in Chapter 5, a simple denoising process is applied after each matrix-vector multiplication, in which Fourier components whose magnitudes are below a certain threshold are zeroed. The effect of this denoising can be seen in Figure 4.2. As can be seen in the results presented in Chapter 5, 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 linear dependence of the Krylov subspace basis vectors. Future work will include adaptive selection of this threshold.

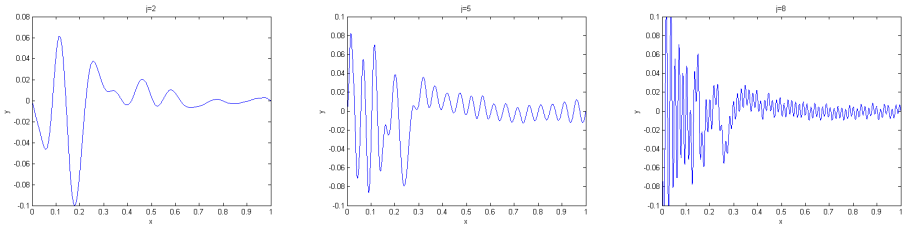


Figure 4.2: Columns of V_m from (4.3) generated by Arnoldi iteration, with denoising, applied to the matrix from Burgers' equation (see Section 5.2.3)

4.2 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 $\varphi(A\tau)\mathbf{b}$, for some function φ, matrix

A, scaling parameter τ based on the time step, and vector $\mathbf{b}$, the following procedure is carried out in place of standard Krylov projection as in (4.3):

1. Use an FFT to decompose $\mathbf{b} = \mathbf{b}_L + \mathbf{b}_H$, where $\mathbf{b}_L$ consists of low-frequency components and $\mathbf{b}_H$ contains all other (high-frequency) components. This is accomplished by zeroing all Fourier coefficients of $\mathbf{b}_L$ for which the absolute value of any wave number exceeds a selected threshold.

2. Use standard Krylov projection as in (4.3) to compute $\varphi(A\tau)\mathbf{b}_L$.

3. Use KSS, with nodes prescribed as in Chapter 3, to compute $\varphi(A\tau)\mathbf{b}_H$.

4. Add the results of steps 2 and 3 to obtain $\varphi(A\tau)\mathbf{b}$.

The decomposition of $\mathbf{b}$ 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 $\mathbf{b}_L$, as the nodes prescribed in Section 3 are based on a high-frequency analysis and are therefore not effective choices at low frequencies [15].

We will denote by N_c the cutoff point for the low-frequency components. Specifically, a Fourier coefficient $\hat{b}(\vec{\omega})$ of $\mathbf{b}$ will be zeroed in $\mathbf{b}_L$ if $\|\vec{\omega}\|_\infty \geq N_c$. 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 Chapter 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 PDE 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 ω.

As discussed in Chapter 3, from the decoupling of the block tridiagonal matrix $\mathcal{T}_K$ 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 ω and half do not. For each ω, 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 $\mathbf{b}_H$, as described in Chapter 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 Chapter 3.

The Fourier component of $\varphi(A\tau)\mathbf{b}_H$ corresponding to the wave number ω is obtained by computing the same Fourier component of $p_{2K-1}(A\tau)\mathbf{b}_H$, where p_{2K-1} is the polynomial interpolant of $\varphi(\lambda)$ with interpolation points $\{\lambda_i, \lambda_{i,\omega}\}_{i=1}^{K}$. Expressing this interpolant in Newton form, we have

$$\begin{aligned} p_{2K-1}(\lambda) &= \varphi[\lambda_1] + \varphi[\lambda_1,\lambda_2](\lambda-\lambda_1) + \cdots \\ &\quad + \varphi[\lambda_1,\lambda_2,\ldots,\lambda_K](\lambda-\lambda_1)\cdots(\lambda-\lambda_{K-1}) \\ &\quad + \varphi[\lambda_1,\ldots,\lambda_K,\lambda_{1,\omega}](\lambda-\lambda_1)\cdots(\lambda-\lambda_K) \\ &\quad + \varphi[\lambda_1,\ldots,\lambda_{2,\omega}](\lambda-\lambda_{1,\omega})(\lambda-\lambda_1)\cdots(\lambda-\lambda_K) + \cdots \\ &\quad + \varphi[\lambda_1,\ldots,\lambda_{K,\omega}](\lambda-\lambda_{1,\omega})\cdots(\lambda-\lambda_{K-1,\omega})(\lambda-\lambda_1)\cdots(\lambda-\lambda_K), \end{aligned} \tag{4.7}$$

where $\varphi[\lambda_1,\ldots,\lambda_i] = \dfrac{\varphi[\lambda_2,\ldots,\lambda_i] - \varphi[\lambda_1,\ldots,\lambda_{i-1}]}{(\lambda_i - \lambda_1)}$ is the i-th divided difference. Arranging the interpolation points in the order indicated above allows us to reduce the number of FFTs needed. Using the relation from Lanczos iteration,

$$AX_K = X_K T_K + \mathbf{r}_K \mathbf{e}_K^T, \tag{4.8}$$

we define

$$\begin{aligned} \mathbf{v} &= p_{K-1}(A)\mathbf{b}_H = \{\varphi[\lambda_1] + \varphi[\lambda_1,\lambda_2](A-\lambda_1 I) + \cdots \\ &\quad + \varphi[\lambda_1,\lambda_2,\ldots,\lambda_K](A-\lambda_1 I)\cdots(A-\lambda_{K-1}I)\}\,\mathbf{b}_H \\ &= \|\mathbf{b}_H\|_2 X_K p_{K-1}(A)\mathbf{e}_1, \\ \mathbf{w} &= q_K(A)\mathbf{b}_H = (A-\lambda_1 I)\cdots(A-\lambda_K I)\mathbf{b}_H \\ &= \beta_1\beta_2\ldots\beta_{K-1}\mathbf{r}_K \\ &= \beta_1\ldots\beta_K X_{K+1}\mathbf{e}_{K+1}, \end{aligned}$$

and

$$\begin{aligned} \tilde{p}_{K-1}(\lambda) &= \varphi[\lambda_1,\ldots,\lambda_K,\lambda_{1,\omega}] + \varphi[\lambda_1,\ldots,\lambda_{2,\omega}](\lambda-\lambda_{1,\omega}) + \cdots \\ &\quad + \varphi[\lambda_1,\ldots,\lambda_{K,\omega}](\lambda-\lambda_{1,\omega})\cdots(\lambda-\lambda_{K-1,\omega}) \\ &= C_{K-1}^{\omega}\lambda^{K-1} + \cdots + C_1^{\omega}\lambda + C_0^{\omega}. \end{aligned}$$

Then, using $\mathscr{F}$ to denote the discrete Fourier transform, we have

$$\varphi(A\tau)\mathbf{b}_H \approx p_{2m-1}(A\tau)\mathbf{b}_H = \mathbf{v} + \tilde{p}_{K-1}(A)\mathbf{w} = \mathbf{v} + \mathscr{F}^{-1}\sum_{j=0}^{K-1}[C_j^{\omega}]\mathscr{F}A^j\mathbf{w}, \tag{4.9}$$

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

Chapter 5

Numerical-Results

5.1 Linear Problems

In this section, we demonstrate the effectiveness of our approach to selecting component-dependent interpolants of the solution operator for PDE. The following three approaches are compared:

- Block KSS, as described in [12, 18],
- Block KSS with rapid node estimation, and
- Krylov projections as seen in (1.6).

Krylov subspaces of the same dimension are used in all three methods to show the benefit of using component-wise polynomial approximations of the solution operator rather than a polynomial approximation.

A comparison of the performance, in terms of both accuracy and efficiency, of block KSS with rapid node estimation against (1.6) is given at the end of this section. In all experiments, as the exact solution to these variable-coefficient problems is not known, error is estimated by computing the solution at various time steps and comparing all solutions against the one computed with the smallest time step.

5.1.1 1-D Parabolic Problems

We start by showing the accuracy of KSS methods combined with rapid node estimation on a parabolic equation in one space dimension,

$$u_t - (p(x)u_x)_x + q(x)u = 0, \quad 0 < x < 2\pi, \quad t > 0, \tag{5.1}$$

where the coefficients $p(x)$ and $q(x)$, given by

$$p(x) = 1, \quad q(x) = \frac{4}{3} + \frac{1}{4}\cos x, \tag{5.2}$$

are chosen to be smooth functions. The initial condition is

$$u(x,0) = 1 + \frac{3}{10}\cos x - \frac{1}{20}\sin 2x, \quad 0 < x < 2\pi, \tag{5.3}$$

and periodic boundary conditions are imposed.

First, we use a Krylov subspace of dimension 4, in order to get 3rd-order accuracy in time. The results are shown in Figure 5.1 and Table 5.1. For both KSS methods, we observe slightly greater than 3rd-order convergence in time, and the error estimates are basicly the same. However, block KSS with rapid node estimation is approximately 10 times faster than standard block KSS for $N = 128$, and 20 times faster when $N = 256$. While (1.6) can achieve 3rd-order accuracy as we decrease the time step, this is not the case for Δt used in Table 5.1, and a slight degradation of performance is observed as N increases. At the same time, the accuracy of the two KSS methods with the two grid sizes is virtually identical.

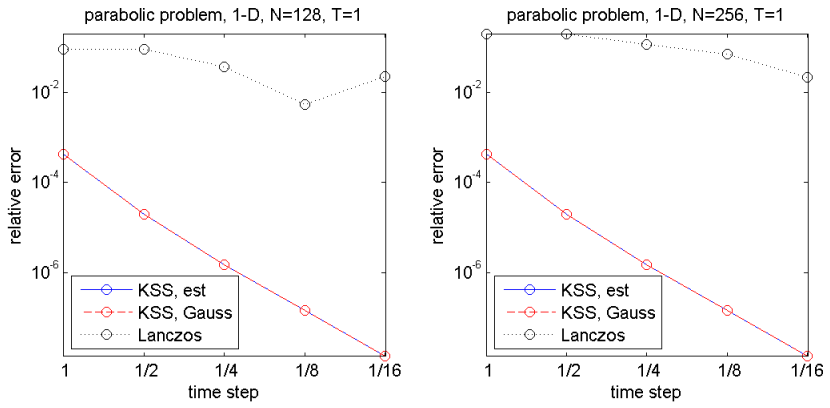


Figure 5.1: Estimates of relative error at $t = 1$ in the solution of (5.1), (5.2), (5.3), with periodic boundary conditions, computed by a 4-node KSS method with rapidly estimated nodes (solid curve), a 4-node block KSS method with Gauss nodes (dashed curve), and Lanczos iteration as described in (1.6) (dotted curve) on an N-point grid and various time steps. All methods are 3rd-order accurate in time.

Increasing the Krylov subspace dimension to 6 should produce fifth-order accurate methods. However, as can be seen from the results shown in Figure 5.2 and Table 5.2, both KSS methods are only fourth-order accurate in time. A decrease in the time step would produce fifth-order convergence. While the accuracy in both KSS methods is independent of the number of grid points and the node selection scheme, KSS with rapid node estimation

N	Δt	KSS-est	KSS-Gauss	Lanczos
128	1	4.268e-004	4.268e-004	8.997e-002
	1/2	1.979e-005	1.979e-005	8.973e-002
	1/4	1.495e-006	1.495e-006	3.620e-002
	1/8	1.435e-007	1.435e-007	5.311e-003
	1/16	1.415e-008	1.415e-008	2.258e-002
256	1	4.268e-004	4.268e-004	1.944e-001
	1/2	1.979e-005	1.979e-005	1.937e-001
	1/4	1.495e-006	1.495e-006	1.163e-001
	1/8	1.435e-007	1.435e-007	6.864e-002
	1/16	1.415e-008	1.415e-008	2.149e-002

Table 5.1: Estimates of relative error at $t = 1$ in the solution of (5.1), (5.2), (5.3), with periodic boundary conditions, computed by a 4-node KSS method with rapidly estimated nodes (KSS-est), a 4-node block KSS method with Gauss nodes (KSS-Gauss), and Lanczos iteration as described in (1.6) (Lanczos) on an N-point grid and various time steps. All methods are 3rd-order accurate in time.

is approximately 10 times faster for $N = 128$ and 20 times faster for $N = 256$. Using Krylov projections with the same Krylov subspace dimension does not yield a similar accuracy. To achieve a similar accuracy, a larger dimension is required.

N	Δt	KSS-est	KSS-Gauss	Lanczos
128	1	1.518e-006	1.520e-006	3.407e-004
	1/2	5.579e-008	5.579e-008	3.189e-003
	1/4	3.913e-009	3.913e-009	1.425e-002
	1/8	8.989e-011	8.989e-011	3.707e-002
	1/16	2.285e-012	2.286e-012	7.244e-002
256	1	1.518e-006	1.520e-006	1.567e-001
	1/2	5.579e-008	5.579e-008	1.565e-001
	1/4	3.913e-009	3.913e-009	9.261e-002
	1/8	8.989e-011	8.989e-011	3.391e-002
	1/16	2.288e-012	2.285e-012	1.987e-002

Table 5.2: Estimates of relative error at $t = 1$ in the solution of (5.1), (5.2), (5.3), with periodic boundary conditions, computed by a 6-node KSS method with rapidly estimated nodes (KSS-est), a 6-node block KSS method with Gauss nodes (KSS-Gauss), and Lanczos iteration as described in (1.6) (Lanczos) on an N-point grid and various time steps. All methods are 5th-order accurate in time.

The nodes used by both KSS methods for Krylov subspace dimensions 4 and 6 are plotted in Figure 5.3.

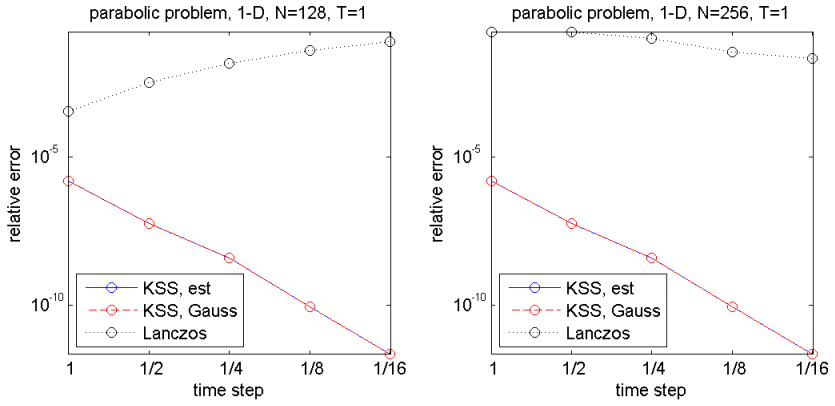


Figure 5.2: Estimates of relative error at $t = 1$ in the solution of (5.1), (5.2), (5.3), with periodic boundary conditions, computed by a 6-node KSS method with rapidly estimated nodes (solid curve), a 6-node block KSS method with Gauss nodes (dashed curve), and Lanczos iteration as described in (1.6) (dotted curve) on an N-point grid and various time steps. All methods are 5th-order accurate in time.

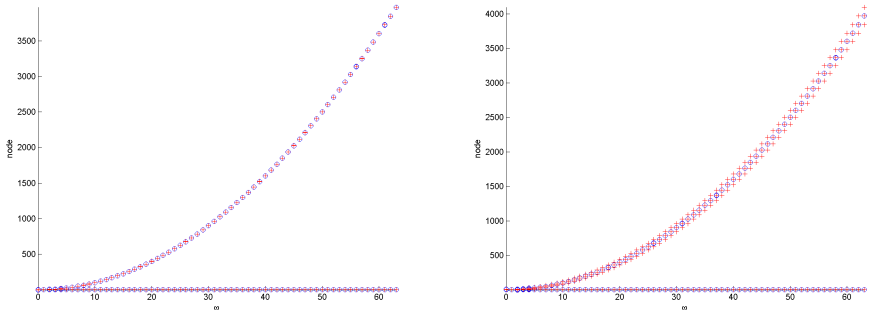


Figure 5.3: Quadrature nodes used by block KSS with Gaussian nodes (red crosses) and estimated Gaussian nodes (blue circles) with 2 (left plot) and 3 (right plot) block Lanczos iterations applied to the operator $Lu = -(pu_x)_x + qu$, with p and q defined in (5.2), for a total of 4 or 6 scalar nodes per frequency component (indicated by ω) in the left and right plots, respectively.

We now change equation (5.1) by adding more oscillatory coefficients

$$\begin{aligned} p(x) &= 1+\frac{1}{4}\cos x-\frac{1}{4}\sin 2x+\frac{1}{8}\cos 3x, \\ q(x) &= 1+\frac{1}{4}\sin x-\frac{1}{4}\cos 2x+\frac{1}{8}\sin 3x-\frac{1}{8}\cos 4x \end{aligned} \tag{5.4}$$

and initial data

$$u(x,0) = 1+\frac{3}{10}\cos x-\frac{3}{20}\sin 2x+\frac{3}{40}\cos 3x, \quad 0<x<2\pi. \tag{5.5}$$

For third-order methods, the results are shown in Figure 5.4 and Table 5.4. As before, both KSS methods perform identically in terms of accuracy independently of the grid size, but are only second-order accurate in time; third-order accuracy can be observed at much smaller time steps. Again, the Krylov projections approach of (1.6) is not competitive with KSS at this Krylov subspace dimension or choice of time step. Figure 5.5 plots the nodes used by both methods.

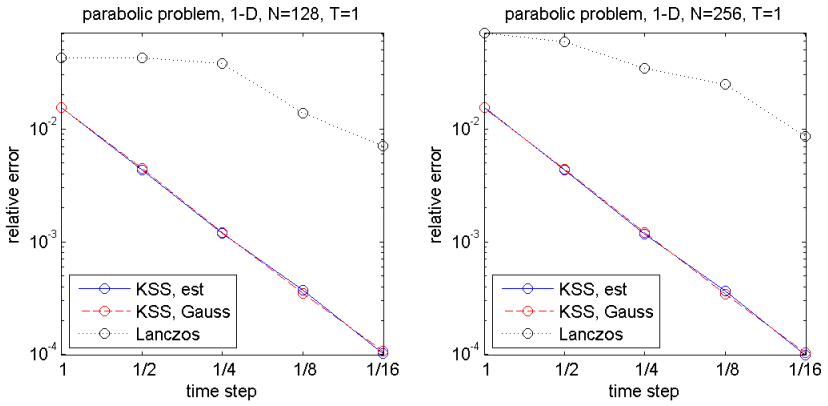


Figure 5.4: Estimates of relative error at $t = 1$ in the solution of (5.1), (5.4), (5.5), with periodic boundary conditions, computed by a 4-node KSS method with rapidly estimated nodes (solid curve), a 4-node block KSS method with Gauss nodes (dashed curve), and Lanczos iteration as described in (1.6) (dotted curve) on an N-point grid and various time steps. All methods are 3rd-order accurate in time.

N	Δt	KSS-est	KSS-Gauss	Lanczos
128	1	1.546e-002	1.541e-002	4.287e-002
	1/2	4.323e-003	4.434e-003	4.262e-002
	1/4	1.180e-003	1.213e-003	3.804e-002
	1/8	3.722e-004	3.466e-004	1.372e-002
	1/16	1.021e-004	1.068e-004	7.108e-003
256	1	1.545e-002	1.540e-002	7.075e-002
	1/2	4.320e-003	4.431e-003	5.914e-002
	1/4	1.177e-003	1.210e-003	3.463e-002
	1/8	3.685e-004	3.435e-004	2.480e-002
	1/16	9.918e-005	1.036e-004	8.636e-003

Table 5.3: Estimates of relative error at $t = 1$ in the solution of (5.1), (5.4), (5.5), with periodic boundary conditions, computed by a 4-node KSS method with rapidly estimated nodes (KSS-est), a 4-node block KSS method with Gauss nodes (KSS-Gauss), and Lanczos iteration as described in (1.6) (Lanczos) on an N-point grid and various time steps. All methods are 3rd-order accurate in time.

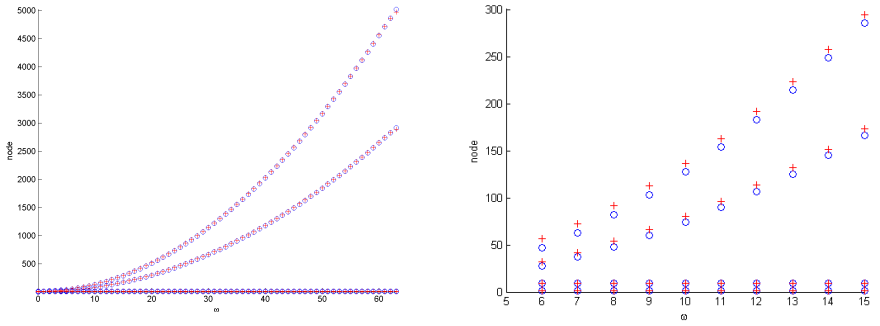


Figure 5.5: Quadrature nodes used by block KSS with Gaussian nodes (red crosses) and estimated Gaussian nodes (blue circles) with 2 block Lanczos iterations applied to the operator $Lu = -(pu_x)_x + qu$, with p and q defined in (5.4), for a total of 4 scalar nodes per frequency component (indicated by ω). The left plot shows frequencies $0 \leq \omega \leq 64$, while the right plot zooms in on frequencies $6 \leq \omega \leq 15$.

We now increase the Krylov subspace dimension to 6 for all three methods. For the first time, we observe a disparity in the performance of the original block KSS method and block KSS with rapid node estimation, which is not as accurate in this case, although both KSS methods again yield results that are, for the most part, independent of the grid size. The reason for this disparity lies in the quadrature nodes corresponding to *low*-frequency components. While the estimation methods deliver sufficiently accurate approximations of the block Gaussian nodes for nearly all frequencies in the case where the leading coefficient

$p(x)$ of the operator L is constant, this is not so when $p(x)$ varies. This is illustrated in Figure 5.7. While overall there is good agreement, as shown in the left plot of the figure, this is not the case for low frequencies, as shown in the right plot, compared to the 4-node case presented earlier.

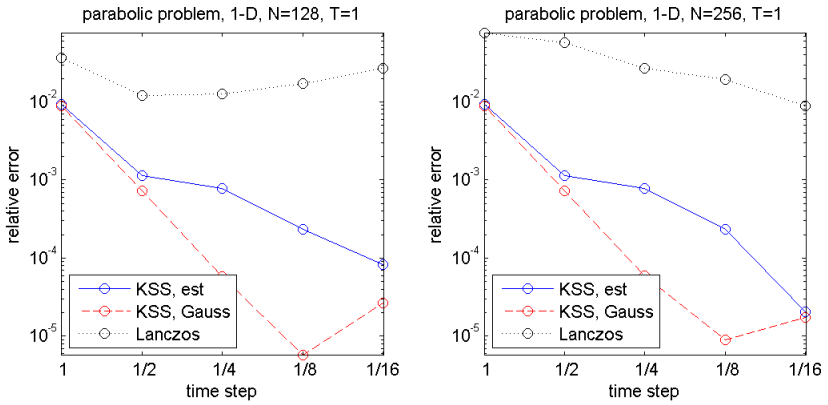


Figure 5.6: Estimates of relative error at $t = 1$ in the solution of (5.1), (5.4), (5.5), with periodic boundary conditions, computed by a 6-node KSS method with rapidly estimated nodes (solid curve), a 6-node block KSS method with Gauss nodes (dashed curve), and Lanczos iteration as described in (1.6) (dotted curve) on an N-point grid and various time steps. All methods are 5th-order accurate in time.

5.1.2 1-D Hyperbolic Problems

We now apply all three methods to the second-order wave equation

$$u_{tt} = (p(x)u_x)_x - q(x)u, \quad 0 < x < 2\pi, \quad t > 0, \tag{5.6}$$

with smooth coefficients $p(x)$ and $q(x)$ defined in (5.2). The initial data is

$$u(x,0) = 1 + \frac{3}{10}\cos x - \frac{1}{20}\sin 2x, \quad u_t(x,0) = \frac{1}{2}\sin x + \frac{2}{25}\cos 2x, \quad 0 < x < 2\pi, \tag{5.7}$$

and periodic boundary conditions are imposed. KSS methods are applied to the wave equation by reducing it to a first-order system and then computing both the solution and its time derivative.

N	Δt	KSS-est	KSS-Gauss	Lanczos
128	1	9.370e-003	8.944e-003	3.705e-002
	1/2	1.130e-003	7.235e-004	1.207e-002
	1/4	7.759e-004	5.832e-005	1.281e-002
	1/8	2.337e-004	5.727e-006	1.722e-002
	1/16	8.212e-005	2.645e-005	2.687e-002
256	1	9.367e-003	8.941e-003	7.634e-002
	1/2	1.128e-003	7.213e-004	5.699e-002
	1/4	7.754e-004	5.931e-005	2.695e-002
	1/8	2.337e-004	8.889e-006	1.960e-002
	1/16	2.069e-005	1.737e-005	8.812e-003

Table 5.4: Estimates of relative error at $t = 1$ in the solution of (5.1), (5.4), (5.5), with periodic boundary conditions, computed by a 6-node KSS method with rapidly estimated nodes (KSS-est), a 6-node block KSS method with Gauss nodes (KSS-Gauss), and Lanczos iteration as described in (1.6) (Lanczos) on an N-point grid and various time steps. All methods are 5th-order accurate in time.

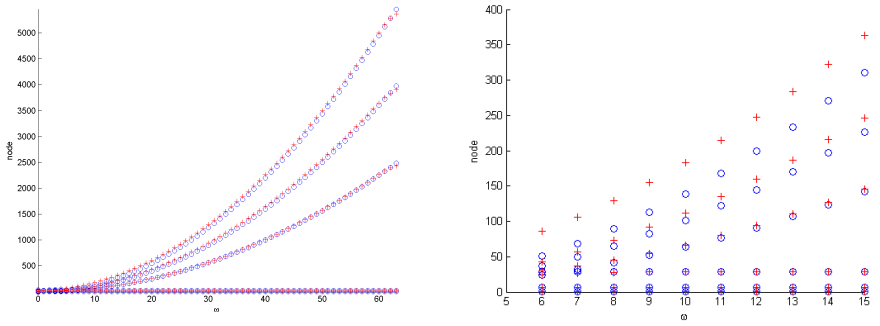


Figure 5.7: Quadrature nodes used by block KSS with Gaussian nodes (red crosses) and estimated Gaussian nodes (blue circles) with 3 block Lanczos iterations applied to the operator $Lu = -(pu_x)_x + qu$, with p and q defined in (5.4), for a total of 6 scalar nodes per frequency component (indicated by ω). The left plot shows frequencies $0 \leq \omega \leq 64$, while the right plot zooms in on frequencies $6 \leq \omega \leq 15$.

The results for 4-dimensional Krylov subspaces are shown in Figure 5.8 and Table 5.5. In this case, KSS methods are generally 6th-order accurate in time. As before, both KSS methods yield similar results on the two different grids, whereas (1.6), while more competitive with KSS than in the parabolic case, exhibits a substantial degradation in accuracy as the number of grid points increases.

We now change the problem in (5.4) by adding more oscillatory coefficients and the

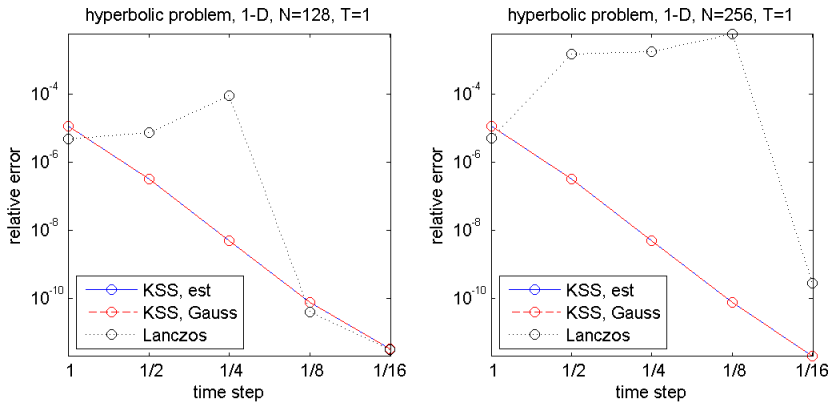


Figure 5.8: Estimates of relative error at $t = 1$ in the solution of (5.6), (5.2), (5.7), with periodic boundary conditions, computed by a 4-node KSS method with rapidly estimated nodes (solid curve), a 4-node block KSS method with Gauss nodes (dashed curve), and Lanczos iteration as described in (1.6) (dotted curve) on an N-point grid and various time steps. All methods are 6th-order accurate in time.

following initial data

$$\begin{aligned} u(x,0) &= 1+\frac{3}{10}\cos x-\frac{3}{20}\sin 2x+\frac{3}{40}\sin 3x, \\ u_t(x,0) &= \frac{1}{2}\sin x+\frac{1}{4}\cos 2x-\frac{1}{8}\sin 3x, \quad 0<x<2\pi, \end{aligned} \tag{5.8}$$

with periodic boundary conditions.

The results for 4-dimensional Krylov subspaces are shown in Figure 5.9 and Table 5.6. While both KSS methods have very similar results for both grid sizes, KSS with rapid estimation is slightly more accurate. At the same time, (1.6), while more competitive with KSS than in the parabolic case, again shows a decrease in accuracy as the number of grid points increases.

N	Δt	KSS-est	KSS-Gauss	Lanczos
128	1	1.167e-005	1.167e-005	4.751e-006
	1/2	3.127e-007	3.127e-007	7.488e-006
	1/4	4.917e-009	4.917e-009	8.900e-005
	1/8	7.360e-011	7.360e-011	3.961e-011
	1/16	3.181e-012	3.181e-012	3.072e-012
256	1	1.167e-005	1.167e-005	4.915e-006
	1/2	3.127e-007	3.127e-007	1.513e-003
	1/4	4.917e-009	4.917e-009	1.814e-003
	1/8	7.346e-011	7.345e-011	5.828e-003
	1/16	1.971e-012	1.971e-012	2.768e-010

Table 5.5: Estimates of relative error at $t = 1$ in the solution of (5.6), (5.2), (5.7), with periodic boundary conditions, computed by a 4-node KSS method with rapidly estimated nodes (KSS-est), a 4-node block KSS method with Gauss nodes (KSS-Gauss), and Lanczos iteration as described in (1.6) (Lanczos) on an N-point grid and various time steps. All methods are 6th-order accurate in time.

N	Δt	KSS-est	KSS-Gauss	Lanczos
128	1	6.543e-003	7.501e-003	2.730e-002
	1/2	3.265e-004	4.084e-004	1.909e-003
	1/4	4.390e-006	5.508e-006	1.168e-003
	1/8	8.150e-008	1.066e-007	5.349e-007
	1/16	1.344e-009	1.768e-009	5.164e-009
256	1	6.543e-003	7.501e-003	2.730e-002
	1/2	3.265e-004	4.084e-004	1.428e-002
	1/4	4.390e-006	5.508e-006	2.315e-002
	1/8	8.150e-008	1.066e-007	3.325e-002
	1/16	1.344e-009	1.768e-009	2.466e-002

Table 5.6: Estimates of relative error at $t = 1$ in the solution of (5.6), (5.4), (5.8), with periodic boundary conditions, computed by a 4-node KSS method with rapidly estimated nodes (KSS-est), a 4-node block KSS method with Gauss nodes (KSS-Gauss), and Lanczos iteration as described in (1.6) (Lanczos) on an N-point grid and various time steps. All methods are 6th-order accurate in time.

5.1.3 2-D Parabolic Problems

The last linear problem that we consider is the 2-D parabolic equation

$$u_t - \nabla \cdot (p(x,y)\nabla u) + q(x,y)u = 0, \quad 0 < x,y < 2\pi, \quad t > 0, \tag{5.9}$$

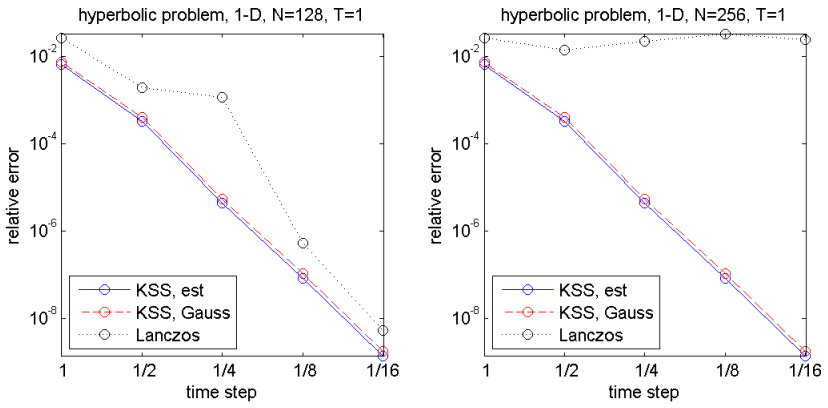


Figure 5.9: Estimates of relative error at $t = 1$ in the solution of (5.6), (5.4), (5.8), with periodic boundary conditions, computed by a 4-node KSS method with rapidly estimated nodes (solid curve), a 4-node block KSS method with Gauss nodes (dashed curve), and Lanczos iteration as described in (1.6) (dotted curve) on an N-point grid and various time steps. All methods are 6th-order accurate in time.

where the smooth coefficients are given by

$$p(x,y) = 1, \quad q(x,y) = \frac{4}{3} + \frac{1}{4}\cos x - \frac{1}{4}\sin y. \tag{5.10}$$

The initial condition is

$$u(x,y,0) = 1 + \frac{3}{10}\cos x - \frac{1}{20}\sin 2y, \quad 0 < x,y < 2\pi, \tag{5.11}$$

and periodic boundary conditions are imposed for both dimensions.

The results are shown in Figure 5.10 and Table 5.7. Both KSS methods produce approximately 3rd-order accuracy in time with similar error estimates. At the same time, block KSS with rapid node estimation is approximately 150 times faster than standard block KSS for $N = 16$, and 600 times faster when $N = 32$. While Krylov projections method does not show a similar accuracy in time and a slight degradation of performance is observed as N increases.

Next, we increase the Krylov subspace dimension to 6, so that all methods should be fifth-order accurate. The results are shown in Figure 5.11 and Table 5.8. Again, in terms

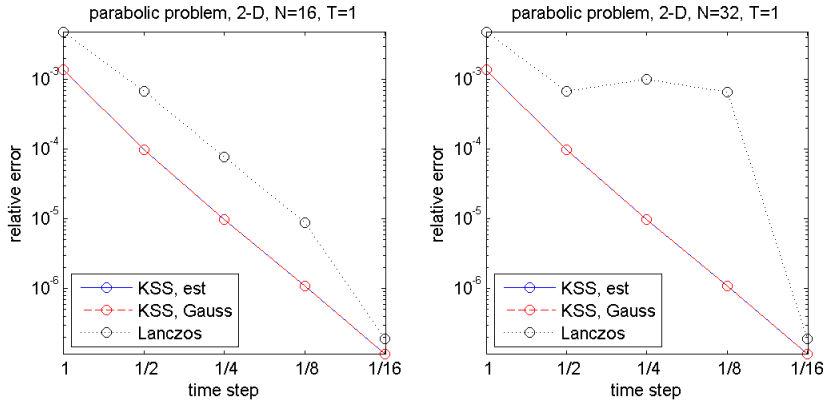


Figure 5.10: Estimates of relative error at $t = 1$ in the solution of (5.9), (5.10), (5.11), with periodic boundary conditions, computed by a 4-node KSS method with rapidly estimated nodes (solid curve), a 4-node block KSS method with Gauss nodes (dashed curve), and Lanczos iteration as described in (1.6) (dotted curve) on an N-point grid (per dimension) and various time steps. All methods are 3rd-order accurate in time.

of accuracy, the performance of both KSS methods is independent of the number of grid points and the node selection scheme; however, as before, KSS with rapid node estimation is approximately 150 times faster for $N = 16$ and 600 times faster for $N = 32$. Using (1.6) with the same Krylov subspace dimension is not competitive in terms of accuracy. To achieve similar accuracy a larger Krylov dimension is needed. Unlike the parabolic 1-D case, both KSS methods actually do achieve the expected fifth-order accuracy in time. As N increases, the degradation in performance of (1.6) is far more substantial than in the 4-dimensional case.

We repeat the experiment with more oscillatory coefficients

$$\begin{aligned} p(x,y) &= 1+\frac{1}{2}\cos(x+y)-\frac{1}{4}\sin(2(x-y))+\frac{1}{8}\cos(3(x+y)), \\ q(x,y) &= 1+\frac{1}{4}\sin x-\frac{1}{4}\cos 2y+\frac{1}{8}\sin 3x-\frac{1}{8}\cos 4y. \end{aligned} \tag{5.12}$$

N	Δt	KSS-est	KSS-Gauss	Lanczos
16	1	1.385e-003	1.385e-003	4.823e-003
	1/2	9.948e-005	9.948e-005	6.809e-004
	1/4	9.785e-006	9.785e-006	7.708e-005
	1/8	1.083e-006	1.083e-006	8.911e-006
	1/16	1.146e-007	1.146e-007	1.884e-007
32	1	1.385e-003	1.385e-003	4.823e-003
	1/2	9.948e-005	9.948e-005	6.811e-004
	1/4	9.785e-006	9.785e-006	1.021e-003
	1/8	1.083e-006	1.083e-006	6.588e-004
	1/16	1.146e-007	1.146e-007	1.884e-007

Table 5.7: Estimates of relative error at $t = 1$ in the solution of (5.9), (5.10), (5.11), with periodic boundary conditions, computed by a 4-node KSS method with rapidly estimated nodes (KSS-est), a 4-node block KSS method with Gauss nodes (KSS-Gauss), and Lanczos iteration as described in (1.6) (Lanczos) on an N-point grid (per dimension) and various time steps. All methods are 3rd-order accurate in time.

N	Δt	KSS-est	KSS-Gauss	Lanczos
16	1	2.272e-005	2.272e-005	3.114e-004
	1/2	5.715e-007	5.715e-007	1.427e-005
	1/4	1.593e-008	1.593e-008	2.204e-006
	1/8	4.591e-010	4.591e-010	2.194e-009
	1/16	1.335e-011	1.335e-011	2.984e-011
32	1	2.272e-005	2.272e-005	3.114e-004
	1/2	5.715e-007	5.715e-007	4.058e-004
	1/4	1.593e-008	1.593e-008	8.116e-004
	1/8	4.591e-010	4.591e-010	4.578e-004
	1/16	1.335e-011	1.335e-011	4.187e-005

Table 5.8: Estimates of relative error at $t = 1$ in the solution of (5.9), (5.10), (5.11), with periodic boundary conditions, computed by a 6-node KSS method with rapidly estimated nodes (KSS-est), a 6-node block KSS method with Gauss nodes (KSS-Gauss), and Lanczos iteration as described in (1.6) (Lanczos) on an N-point grid (per dimension) and various time steps. All methods are 5th-order accurate in time.

The initial data

$$u(x,y,0) = 1 + \frac{3}{10}\cos x - \frac{3}{20}\sin(2(x+y)) + \frac{3}{40}\cos 3x, \quad 0 < x,y < 2\pi, \tag{5.13}$$

is more oscillatory as well.

For third-order methods, the results are shown in Figure 5.12 and Table 5.12. Both KSS methods are only slightly greater than second-order accurate in time while third-order

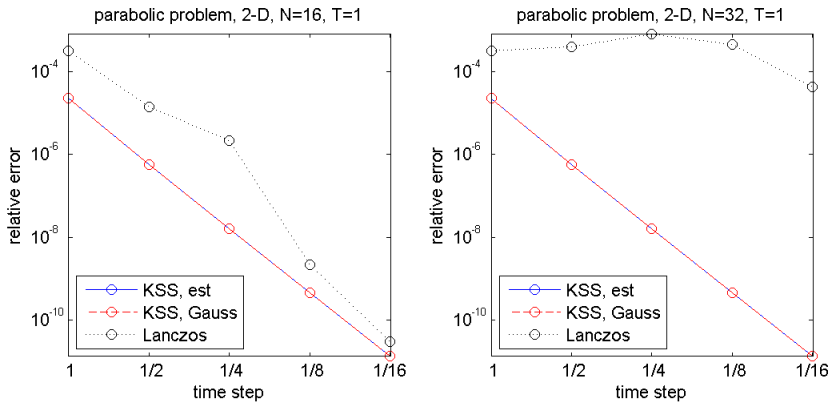


Figure 5.11: Estimates of relative error at $t = 1$ in the solution of (5.9), (5.10), (5.11), with periodic boundary conditions, computed by a 6-node KSS method with rapidly estimated nodes (solid curve), a 6-node block KSS method with Gauss nodes (dashed curve), and Lanczos iteration as described in (1.6) (dotted curve) on an N-point grid (per dimension) and various time steps. All methods are 5th-order accurate in time.

accuracy can be observed at much smaller time steps. As before, the approach of (1.6) is not competitive with KSS at this Krylov subspace dimension or choice of time step, and once again, there is significant degradation in accuracy as N increases, as this increase requires a corresponding increase in the Krylov subspace dimension.

We now increase the Krylov subspace dimension to 6 for all three methods. As in the 1-D case, we observe a disparity in the performance of the original block KSS method and block KSS with rapid node estimation, which is not as accurate in this case. Furthermore, we also observe a disparity in the results for the two grid sizes, except that accuracy and order of convergence *improves* as N increases, while (1.6), once again, shows the opposite trend. Because the time step is too large, the KSS methods only exhibit roughly second-order accuracy again, on average.

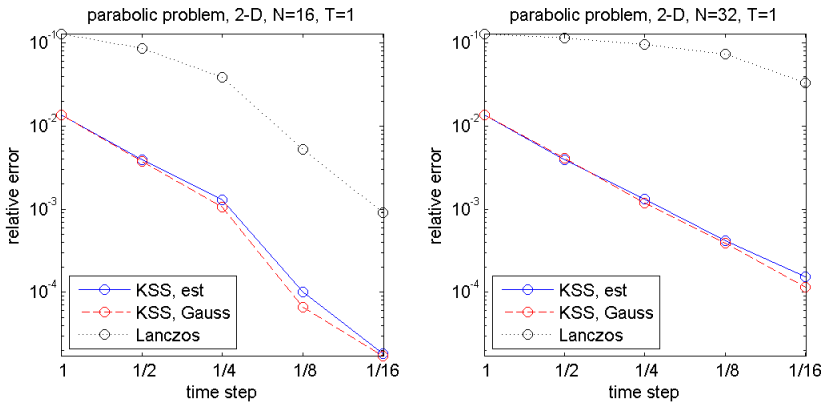


Figure 5.12: Estimates of relative error at $t = 1$ in the solution of (5.9), (5.12), (5.13), with periodic boundary conditions, computed by a 4-node KSS method with rapidly estimated nodes (solid curve), a 4-node block KSS method with Gauss nodes (dashed curve), and Lanczos iteration as described in (1.6) (dotted curve) on an N-point grid (per dimension) and various time steps. All methods are 3rd-order accurate in time.

5.1.4 Performance

Now, we use (1.6) in a different way, that is consistent with its use in time-stepping methods such as those described in, among other sources, [10, 14]. That is, for each product of the form $f(A)\mathbf{b}$ that needs to be computed, Lanczos iteration continues until convergence is achieved to within a specified tolerance, rather than being restricted to a Krylov subspace dimension that is determined by the desired temporal order of accuracy, as in KSS methods. We solve the 1-D parabolic problem (5.1), (5.2), (5.3), with smooth coefficients and initial data, with grids of dimension $N = 128, 256, 512$.

The results are shown in Figure 5.14. It can be seen that as N increases, the amount of time needed to achieve a given level of accuracy by (1.6), allowed to run until convergence is achieved to a relative error tolerance of 10^{-7}, is far greater than that required by a 4-node block KSS method with rapid node estimation. For the KSS method, the Krylov subspace dimension is always 4, whereas for (1.6), the maximum number of iterations needed in a

N	Δt	KSS-est	KSS-Gauss	Lanczos
16	1	1.357e-002	1.370e-002	1.284e-001
	1/2	3.911e-003	3.766e-003	8.602e-002
	1/4	1.299e-003	1.051e-003	3.845e-002
	1/8	1.011e-004	6.626e-005	5.205e-003
	1/16	1.830e-005	1.703e-005	9.002e-004
32	1	1.357e-002	1.375e-002	1.273e-001
	1/2	3.923e-003	4.065e-003	1.142e-001
	1/4	1.310e-003	1.172e-003	9.730e-002
	1/8	4.192e-004	3.860e-004	7.405e-002
	1/16	1.529e-004	1.146e-004	3.295e-002

Table 5.9: Estimates of relative error at $t = 1$ in the solution of (5.9), (5.12), (5.13), with periodic boundary conditions, computed by a 4-node KSS method with rapidly estimated nodes (KSS-est), a 4-node block KSS method with Gauss nodes (KSS-Gauss), and Lanczos iteration as described in (1.6) (Lanczos) on an N-point grid (per dimension) and various time steps. All methods are 3rd-order accurate in time.

N	Δt	KSS-est	KSS-Gauss	Lanczos
16	1	8.998e-003	8.423e-003	9.219e-002
	1/2	2.971e-003	8.671e-004	2.787e-002
	1/4	9.370e-004	6.374e-005	4.485e-003
	1/8	3.549e-005	2.653e-005	4.631e-004
	1/16	1.714e-005	1.682e-005	3.737e-005
32	1	8.803e-003	8.488e-003	1.099e-001
	1/2	2.678e-003	1.457e-003	8.929e-002
	1/4	1.626e-003	3.309e-004	5.766e-002
	1/8	1.078e-004	7.003e-006	1.185e-002
	1/16	5.230e-007	4.257e-007	1.199e-003

Table 5.10: Estimates of relative error at $t = 1$ in the solution of (5.9), (5.12), (5.13), with periodic boundary conditions, computed by a 6-node KSS method with rapidly estimated nodes (KSS-est), a 6-node block KSS method with Gauss nodes (KSS-Gauss), and Lanczos iteration as described in (1.6) (Lanczos) on an N-point grid (per dimension) and various time steps. All methods are 5th-order accurate in time.

time step for $N = 128, 256, 512$ was 21,35 and 60, respectively. On the other hand, the time required by the KSS method scales approximately linearly with N.

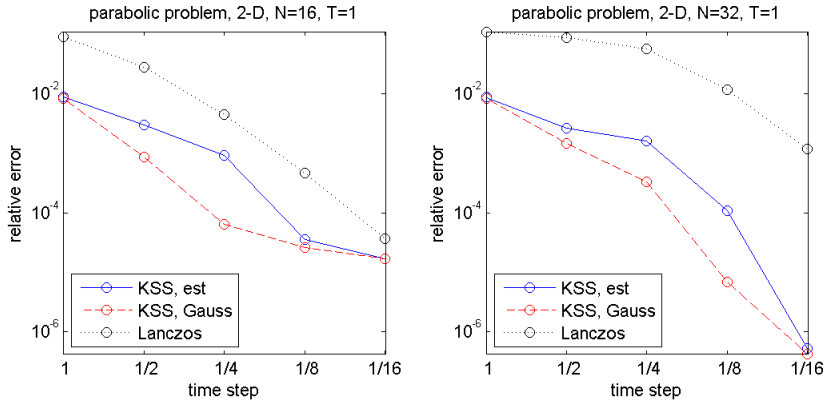


Figure 5.13: Estimates of relative error at $t = 1$ in the solution of (5.9), (5.12), (5.13), with periodic boundary conditions, computed by a 6-node KSS method with rapidly estimated nodes (solid curve), a 6-node block KSS method with Gauss nodes (dashed curve), and Lanczos iteration as described in (1.6) (dotted curve) on an N-point grid (per dimension) and various time steps. All methods are 5th-order accurate in time.

5.2 Non-Linear Problems

In this section we compare several versions of EPI methods, as applied to three test problems. The versions differ in the way in which they compute matrix function-vector products of the form $\varphi(A\tau)\mathbf{b}$:

- Standard Krylov projection, as in (4.3), hereafter referred to as "Krylov-EPI", either with or without denoising as in Chapter 4,
- Using the KSS approach, as described in Section 4.2, hereafter referred to as "KSS-EPI",
- Newton interpolation using Leja points [2], hereafter referred to as "LEJA", and
- Adaptive Krylov projection [23], hereafter referred to as "AKP".

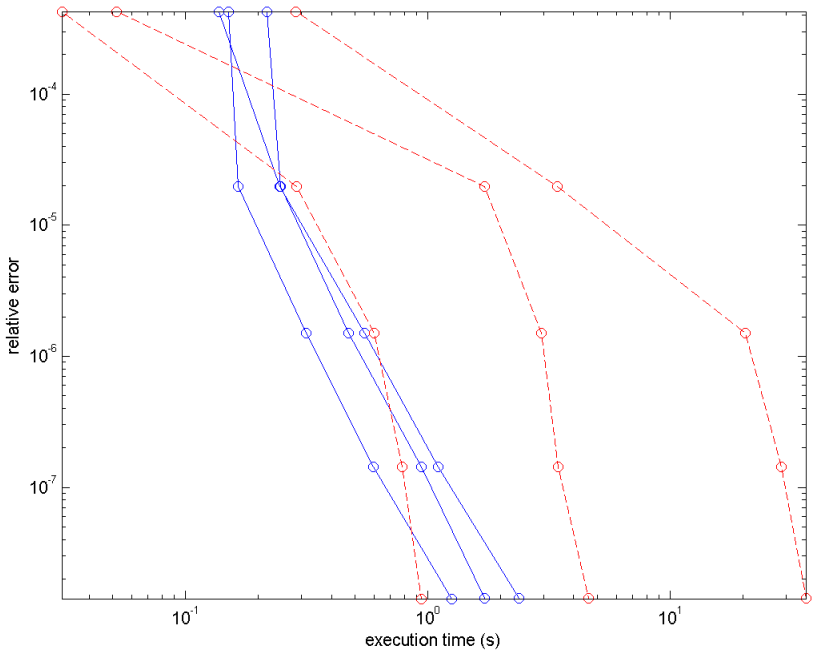


Figure 5.14: Estimates of relative error at $t = 1$ in the solution of (5.1), (5.2), (5.3), with periodic boundary conditions, computed by a 4-node block KSS method with rapid node estimation (solid blue curves), and Lanczos iteration as described in (1.6) (dashed red curves) with various time steps. For both methods, the curves, as displayed from left to right, correspond to solutions computed on N-point grids for $N = 128, 256, 512$.

All of these approaches are used in the context of three EPI methods described in Chapter 4. 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.

5.2.1 LEJA and AKP

We start this section by giving a brief description of the Newton interpolation using Leja points as described in [2] and [3]. Given a compact set $K \subset \mathbb{C}$, the sequence of Leja points $\{z_j\}_{j=0}^{\infty}$ are defined recursively by starting with a fixed point $z_0 \in K$, where $|z_0| = \max_{z \in K} |z|$, and then the z_j are chosen such that

$$\prod_{k=0}^{j-1} |z_j - z_k| = \max_{z \in K} \prod_{k=0}^{j-1} |z_j - z|, \quad j = 1, 2 \ldots . \tag{5.14}$$

The LEJA method interpolates $\varphi(h\lambda)$ using the Leja points for a suitable timestep $h \leq \Delta t$ on the interval $[c-2\gamma, c+2\gamma]$, where γ is the analytic capacity of K . The matrix Newton polynomials p_m of degree m that interpolate $\varphi(hA)$ using Leja points are defined as

$$p_m(A) = \sum_{i=0}^{m} d_i \Omega_i \approx \varphi(hA), \quad \Omega_i = \prod_{j=0}^{i-1}((A-cI)/\gamma - z_j I), \tag{5.15}$$

where $\{d_i\}$ is the corresponding divided difference for the function $\varphi(hA)$. The sequence of polynomials in (5.15) converges maximally to φ on K, i.e.

$$\limsup \|\varphi - p_m\|_K^{1/m} = \limsup \|\varphi - p_m^*\|_K^{1/m}, \tag{5.16}$$

where $\|\cdot\|_K$ is the maximum norm of K, and $\{p^*\}$ is the sequence of best uniform approximation polynomials of φ on K.

When the expected degree of convergence is too large, LEJA with the original time step Δt becomes unfeasible. To deal with this, the LEJA algorithm divides the original time step into smaller time steps $h = h_k$. Then, it produces the solution vector $\varphi(hA)\mathbf{v}$ using the following time steping scheme:

$$\mathbf{y}_{k+1} = \mathbf{y}_k + h_k \varphi(h_k A)(A\mathbf{y}_k + v), \quad k = 0, 1, \ldots, \tag{5.17}$$

where $\sum h_k = \Delta t$ and $\Delta t \varphi(\Delta t A)\mathbf{v}$ is the solution at time $t = \Delta t$ of the differential system $\mathbf{y}'(t) = A(\mathbf{y})(t) + \mathbf{v}, \mathbf{y}(0) = 0$.

We now give a brief description of the AKP method as seen in [23]. Given the ODE system

$$u'(t) = Au(t) + b_1 + tb_2 + \cdots + \frac{t^{p-1}}{(p-1)!}b_p, \quad u(0) = b_0, \tag{5.18}$$

with the exact solution

$$u(t) = \varphi_0(tA)b_0 + t\varphi_1(tA)b_1 + t^2\varphi_2(tA)b_2 + \cdots + t^p\varphi_p(tA)b_p, \tag{5.19}$$

the idea behind the AKP method is to evaluate a linear combination of type

$$\varphi_0(A)b_0 + \varphi_1(A)b_1 + \varphi_2(A)b_2 + \cdots + \varphi_p(A)b_p,$$

which is just the expression (5.19) evaluated at time $t = 1$. By splitting the interval $[0,1]$ into subintervals $0 = t_0 < t_1 < \cdots < t_k < t_{k+1} = t_k + \tau_k < \cdots < t_K = 1$, the solution at time t_{k+1} can be expressed exactly in terms of the solution at the previous time step as follows

$$u(t_{k+1}) = \varphi_0(\tau_k A)u(t_k) + \sum_{i=1}^{p} \tau_k^i \varphi_i(\tau_k A) \sum_{j=0}^{p-1} \frac{t_k^j}{j!} b_{i+j}, \quad \tau_k = t_{k+1} - t_k. \tag{5.20}$$

At each step τ_k, only one evaluation of the φ-function product is needed. Since the matrix A is scaled by τ_k, where $0 < \tau_k < 1$, the computation of $\varphi_p(\tau_k A)w_p$ needs less Krylov vectors than the computation of $\varphi_p(A)w_p$. This adaptive substepping approach is more efficient if the total comptational cost of evaluating the small Krylov subspaces for all K substeps is smaller than computing one large Krylov subspace for the large initial time step. Since the computational cost of evaluating one Krylov projection scales quadratically with the number of Krylov vectors needed, it is possible that evaluating a few smaller Krylov bases is cheaper than computing a large Krylov subspace.

5.2.2 Diffusive Problem

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

$$u_t = \alpha\nabla^2 u + u - u^3, \quad x,y \in [0,1], t \in [0,0.2] \tag{5.21}$$

with $\alpha = 0.1$, using homogeneous Neumann boundary conditions and initial conditions given by

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

The ∇^2 term is discretized using a centered finite difference. For KSS-EPI, the low-frequency portion $\mathbf{b}_L$ 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, as defined in Section 4.2, for this problem is 7. The formula for the frquency-dependent nodes, as defined in (3.40), of the 3-rd order EPI scheme is given by

$$nf = \left[0.1\|\vec{\omega}\|^2 + \bar{q} - \|\tilde{q}\| \quad 0.1\|\vec{\omega}\|^2 + \bar{q} + \|\tilde{q}\|\right], \tag{5.22}$$

where $q = 1 - 3u^2$ is obtained from the Jacobian of (5.21), $\bar{q}$ is the average value of q, and $\tilde{q} = q - \bar{q}$. Similarly, the 4-th and 5-th order frequency-dependent nodes are computed by the formula

$$nf = \left[0.1\|\vec{\omega}\|^2 + \bar{q} - \sqrt{2}\|\tilde{q}\| \quad 0.1\|\vec{\omega}\|^2 + \bar{q} \quad 0.1\|\vec{\omega}\|^2 + \bar{q} + \sqrt{2}\|\tilde{q}\|\right]. \tag{5.23}$$

Figures 5.15, 5.16 and 5.17 show the error vs. time performance for the two approaches to matrix-function-vector multiplication, used within the 3rd- and 5th-order EPI methods, respectively, as well as the Leja point approximation and adaptive Krylov (AKP) methods, while Tables 5.11, 5.14, and 5.17 show the errors for all four methods relative to the time step. We can see from the tables that the errors for both Krylov-EPI and KSS-EPI methods are essentially the same, but as can be seen in the figures, the computational time is different. For both orders, only for the grid size $N = 25$ grid points per dimension is the Krylov-EPI

method slightly faster than the KSS-EPI method, 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 Tables 5.13, 5.16 and 5.19. 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.

The denoising effect is clearly shown in this problem. From Tables 5.13, 5.16, and 5.19 we can see that denoising does not reduce the number of iterations considerably for $N = 25$ and $N = 50$, resulting in a slight increase in computational time. However, for $N = 150$ and and $N = 300$, the number of iterations is reduced considerably, as can be seen in Tables 5.12, 5.15, and 5.18. Although AKP is slightly more accurate than KSS-EPI for 5th order, this is more than offset by the far superior efficiency of KSS-EPI with denoising.

Grid Size	Krylov-EPI	LEJA	AKP	KSS-EPI	KSS-EPI (denoised)
25	0.0020 3.9197e-04 5.4020e-05 6.3915e-06 7.4846e-07	0.0020 3.9197e-04 5.4020e-05 6.3915e-06 7.4846e-07	0.0020 3.9197e-04 5.4020e-05 6.3915e-06 7.4846e-07	0.0020 3.9197e-04 5.4020e-05 6.3915e-06 7.4846-07	0.0020 3.9197e-04 5.4021e-05 6.3921e-06 7.4856e-07
50	0.0021 4.0639e-04 5.6536e-05 6.7028e-06 7.8453e-07	0.0011 4.0639e-04 5.6536e-05 6.7028e-06 7.8453e-07	0.0021 4.0639e-04 5.6536e-05 6.7028e-06 7.8453e-07	0.0021 4.0639e-04 5.6536e-05 6.7028e-06 7.8453e-07	0.0021 4.0639e-04 5.6537e-05 6.7034e-06 7.8464e-07
150	0.0021 4.1072e-04 5.7300e-05 6.7977e-06 7.9553e-07	3.3942e-04 1.3867e-05 1.9368e-05 3.9875e-06 7.9555e-07	0.0021 4.1072e-04 5.7300e-05 6.7977e-06 7.9553e-07	0.0021 4.1072e-04 5.7301e-05 6.7977e-06 7.9553e-07	0.0021 4.1072e-04 5.7302e-05 6.8033e-06 7.9655e-07
300	0.0021 4.1112e-04 5.7372e-05 6.8066e-06 7.9656e-07	2.7226e-04 5.4533e-05 3.8506e-05 1.2076e-05 2.8286e-06	0.0021 4.1112e-04 5.7372e-05 6.8067e-06 7.9657e-07	0.0021 4.1112e-04 5.7373e-05 6.8067e-06 7.9659e-07	0.0021 4.1126e-04 5.7412e-05 6.8123e-06 7.9755e-07

Table 5.11: Error for Allen-Cahn equation, 3rd order

5.2.3 Advective Problem

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

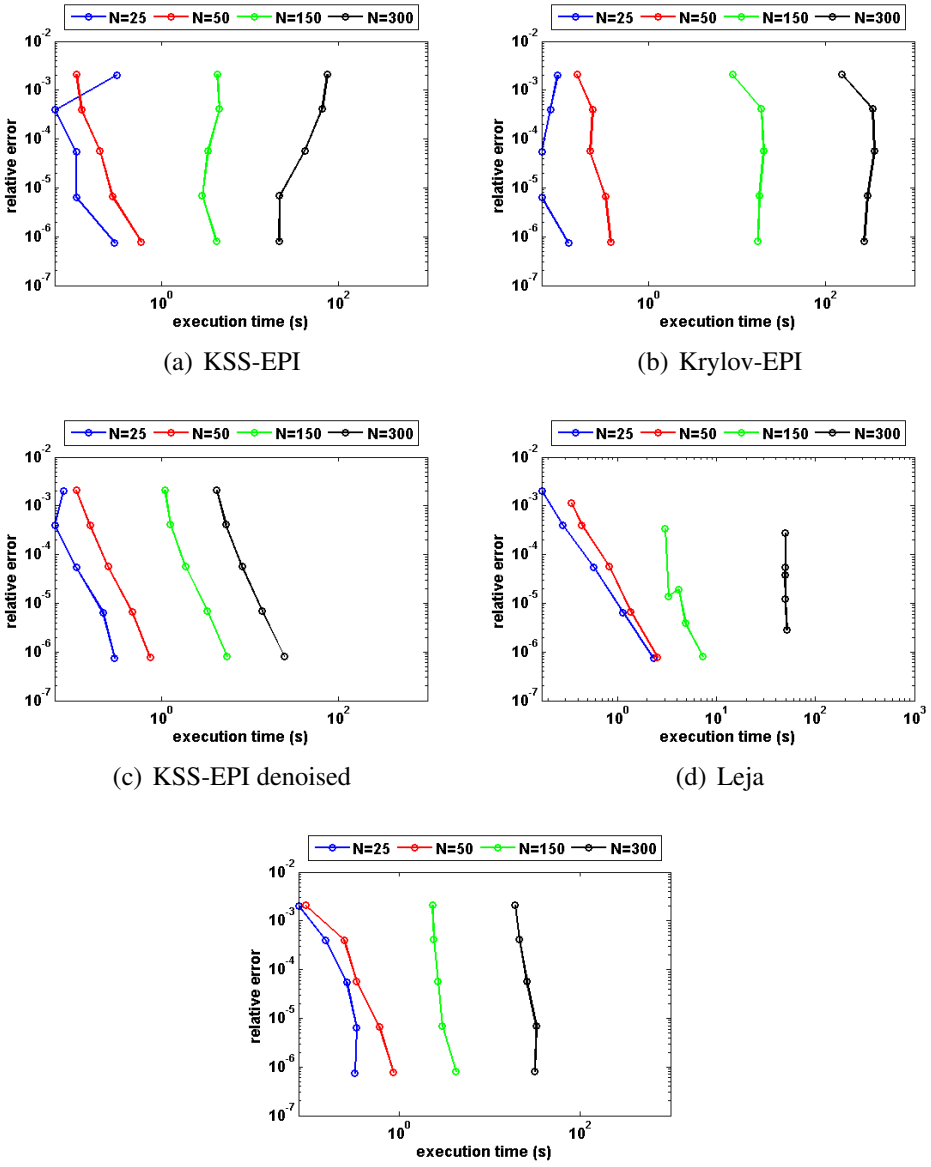
$$u_t + uu_x = \nu u_{xx}, \quad x \in [0,1], t \in [0,1] \tag{5.24}$$

Grid Size	Krylov-EPI	LEJA	AKP	KSS-EPI	KSS-EPI (denoised)
25	0.0936	0.1716	0.0780	0.3120	0.0780
	0.0780	0.2808	0.1560	0.0624	0.0624
	0.0624	0.5772	0.2652	0.1092	0.1092
	0.0624	1.1388	0.3432	0.1092	0.2184
	0.1248	2.3088	0.3276	0.2964	0.2964
50	0.1560	0.3432	0.0936	0.1092	0.1092
	0.2340	0.4368	0.2496	0.1248	0.1560
	0.2184	0.8268	0.3432	0.2028	0.2469
	0.3276	1.3572	0.6084	0.2808	0.4680
	0.3744	2.5428	0.8736	0.5928	0.7488
150	8.8765	3.0264	2.3556	4.2588	1.0920
	18.7045	3.2916	2.4180	4.4928	1.2636
	19.7341	4.1808	2.7144	3.3228	1.8720
	17.6437	4.8516	3.0108	2.8860	3.3072
	17.1289	7.3008	4.2900	4.1496	5.4912
300	152.1478	49.7643	19.15695	73.9133	4.1964
	337.9606	49.8111	21.4345	64.4908	5.3352
	354.3407	48.9999	25.8182	41.3871	8.1277
	293.5939	49.4835	32.6822	21.4501	13.7125
	269.8817	51.3555	31.7618	21.2629	24.3674

Table 5.12: Computational time for Allen-Cahn equation, 3rd order

Grid Size	Krylov-EPI	LEJA	AKP	KSS-EPI	KSS-EPI (denoised)
25	14.5000	42.5000	18.0000	13.5000	13.5000
	12.5000	30.0000	14.5000	10.5000	10.0000
	10.3750	21.3750	10.2500	8.1250	8.0000
	8.0000	15.3750	7.3125	6.6875	6.6875
	6.5000	11.5938	5.3125	5.7188	5.6875
50	26.0000	142.0000	47.0000	19.0000	16.0000
	23.2500	54.2500	39.5000	14.5000	10.2500
	17.5000	38.1250	20.0000	9.8750	8.3750
	12.7500	26.1250	12.7500	7.1250	6.8125
	9.2188	19.5625	7.8750	5.8438	5.7813
150	63.5000	1.0745e+03	375.5000	46.0000	16.5000
	62.2500	509.7500	235.2500	32.0000	10.5000
	47.0000	252.3750	104.1250	16.7500	8.0000
	32.8750	125.9375	46.5000	9.0625	6.7500
	22.9375	47.7500	26.3750	6.4375	5.7813
300	122.0000	4286.0000	656.0000	89.0000	16.5000
	119.7500	2.0628e+03	547.5000	58.2500	10.7500
	91.7500	979.6250	318.5000	28.2500	8.1250
	63.1250	471.4375	193.5625	12.9375	6.6250
	43.6250	225.4375	82.7188	7.3438	5.6875

Table 5.13: Number of iterations for Allen-Cahn equation, 3rd order



(a) KSS-EPI

(b) Krylov-EPI

(c) KSS-EPI denoised

(d) Leja

(e) AKP

Figure 5.15: Allen-Cahn equation, 3rd order

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

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

For KSS-EPI, the low-frequency portion $\mathbf{b}_L$ consists of all components with wave numbers $\omega \leq N_c$. A higher threshold $N_c = 40$ is used in this problem than for the Allen-Cahn equation, as the initial data is less smooth. The 3-rd order frequency-dependent nodes, as defined in

Grid Size	Krylov-EPI	LEJA	AKP	KSS-EPI	KSS-EPI (denoised)
25	4.5620e-04	9.0236e-04	8.4186e-04	4.5621e-04	4.8852e-04
	6.6711e-05	4.1242e-04	4.1596e-04	6.6712e-05	6.7060e-05
	5.5878e-06	1.3398e-04	1.4079e-04	5.5878e-06	5.7474e-06
	3.5303e-07	3.5032e-05	3.7431e-05	3.5303e-07	3.7913e-07
	2.0112e-08	8.6735e-06	9.3370e-06	2.0112e-08	2.3368e-08
50	4.6947e-04	6.6385e-04	8.1799e-04	4.6948e-04	5.0168e-04
	7.0286e-05	4.1041e-04	4.1292e-04	7.0287e-05	7.0580e-05
	5.9719e-06	1.3523e-04	1.1499e-04	5.9719e-06	6.1310e-06
	3.7955e-07	3.5499e-05	3.7928e-05	3.7955e-07	4.0625e-07
	2.1668e-08	8.7942e-06	9.4669e-06	2.1665e-08	2.5072e-08
150	4.7340e-04	4.6924e-04	8.1090e-04	4.7431e-04	5.0559e-04
	7.1370e-05	2.7112e-04	4.1196e-04	7.1371e-05	7.1647e-05
	6.0901e-06	1.0023e-04	1.4235e-04	6.0901e-06	6.2152e-06
	3.8777e-07	2.9602e-05	3.8076e-05	3.8776e-07	4.0961e-07
	2.2183e-08	8.8224e-06	9.5058e-06	2.2150e-08	2.4723e-08
300	4.7377e-04	4.4949e-04	8.1024e-04	4.7378e-04	5.0596e-04
	7.1471e-05	2.5622e-04	4.1187e-04	7.1473e-05	7.1748e-05
	6.1013e-06	9.1483e-05	1.4238e-04	6.1012e-06	6.2268e-06
	3.8860e-07	2.5132e-05	3.8090e-05	3.8852e-07	4.1040e-07
	2.2338e-08	6.6219e-06	9.5095e-06	2.2193e-08	2.4774e-08

Table 5.14: Error for Allen-Cahn, 4th order

Grid Size	Krylov-EPI	LEJA	AKP	KSS-EPI	KSS-EPI (denoised)
25	0.3744	0.3900	0.5148	0.4836	0.1404
	0.8424	0.9204	0.4056	0.7644	0.1716
	1.1856	1.4664	0.5772	1.1700	0.1872
	2.0748	2.9328	0.9516	1.5132	0.3432
	2.8392	5.4444	0.7644	1.4040	0.4680
50	0.7956	0.6084	0.3432	0.5772	0.1560
	1.2792	1.2948	0.6396	0.9204	0.2184
	1.9500	2.0124	0.9204	1.3728	0.4524
	3.3540	3.2916	1.4040	2.0436	0.7488
	5.5536	6.2556	2.1060	3.6036	1.4040
150	2.9640	5.4288	8.0700	2.3712	1.4040
	5.8032	5.8656	6.6612	2.9016	1.9500
	23.4158	6.3804	8.3461	4.1808	3.1668
	38.7038	8.6737	8.8297	7.0980	5.6784
	52.8687	13.9933	10.4677	12.7453	10.6237
300	10.3741	89.9802	54.1167	8.3305	6.2556
	111.4315	79.8101	57.7984	10.9669	9.1105
	323.5329	69.8260	71.3705	15.6469	13.2757
	455.5229	64.4752	79.6853	26.2238	23.9306
	554.8332	66.9712	84.4589	46.3167	46.4415

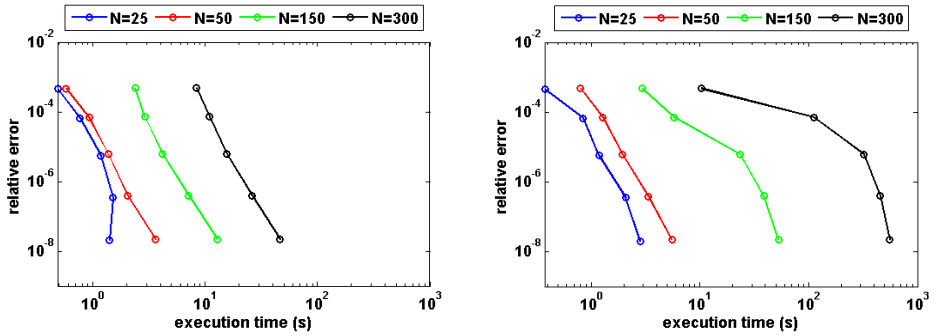
Table 5.15: Computation time for Allen-Cahn, 4th order

Grid Size	Krylov-EPI	LEJA	AKP	KSS-EPI	KSS-EPI (denoised)
25	14.3333	48.0000	17.0000	13.6667	12.6667
	12.6667	32.0000	12.0833	10.5000	9.3333
	9.6667	21.1667	8.4167	8.0833	7.0833
	8.1250	14.7500	6.1875	6.7083	6.0417
	6.8542	9.9583	4.3750	5.7083	5.4583
50	20.3333	157.3333	34.1667	16.3333	14.0000
	17.3333	53.5000	28.6667	11.6667	9.6667
	13.5000	34.4167	15.7083	8.4167	7.1667
	11.9167	22.5417	10.0000	6.9167	6.2083
	9.4167	14.3333	5.8542	5.8333	5.3958
150	21.6667	1.3013e+03	302.8333	17.6667	14.0000
	21.1667	519.5000	209.3333	12.0000	9.8333
	30.5833	190.5833	114.2083	8.5833	7.1667
	28.8333	70.2917	48.1250	6.9583	6.0833
	22.9583	25.1250	20.9063	5.8542	5.5000
300	23.0000	5.2227e+03	873.6667	17.6667	14.0000
	50.1667	2.1858e+03	506.4167	11.8333	10.0000
	61.2500	808.0000	295.0000	8.5833	6.9167
	54.5833	281.8750	155.9375	6.9167	6.0417
	42.8542	101.0208	73.2396	5.8542	5.8333

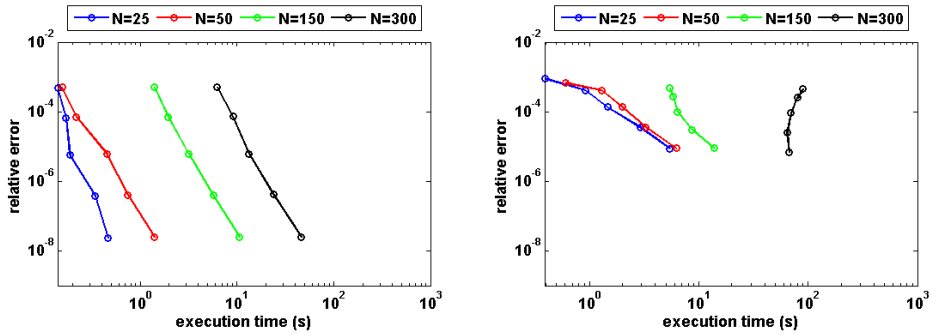
Table 5.16: Number of iterations for Allen-Cahn, 4th order

Grid Size	Krylov-EPI	LEJA	AKP	KSS-EPI	KSS-EPI (denoised)
25	2.6065e-05	9.4043e-04	2.6065e-05	2.6071e-05	2.6071e-05
	2.5909e-07	2.9471e-04	2.5880e-07	2.5869e-07	2.5876e-07
	4.4053e-08	8.4422e-05	4.4019e-08	4.4008e-08	4.4569e-08
	2.2979e-09	2.1781e-05	2.2612e-09	2.2617e-09	3.5787e-09
	1.0488e-10	5.4379e-06	8.3054e-11	8.4803e-11	8.3231e-10
50	2.6675e-05	0.012	2.6675-05	2.6681e-05	2.6681e-05
	2.6101e-07	2.9641e-04	2.5858e-07	2.5964e-07	2.5827e-07
	4.7887e-08	8.5382e-05	4.7241e-08	4.7274e-08	4.7739e-08
	2.8597e-09	2.2074e-05	2.4656e-09	2.4671e-09	4.3559e-09
	9.2058e-10	5.5124e-06	9.0992e-11	9.3881e-11	1.0330e-09
150	2.6872e-05	0.014	2.6872e-05	2.6875e-05	2.6878e-05
	2.9126e-07	4.0409e-04	2.5871e-07	2.7710e-07	2.5834e-07
	5.6581e-08	1.0736e-04	4.7228e-08	4.9135e-08	1.0347e-07
	2.8521e-08	2.5263e-05	2.5292e-09	5.0278e-09	9.0062e-09
	1.7910e-08	5.5347e-06	9.3473e-11	8.2219e-10	1.4776e-09
300	2.6891e-05	0.0014	2.6891e-05	2.6901e-05	2.6900e-05
	3.9812e-07	4.2024e-04	2.5873e-07	2.7520e-07	2.5847e-07
	8.5739e-08	1.1555e-04	4.8321e-08	1.0100e-07	1.0067e-07
	8.2542e-08	2.8554e-05	2.5353e-09	6.0632e-08	9.0657e-09
	3.0576e-08	6.7601e-06	9.3708e-11	8.5444e-09	1.5032e-09

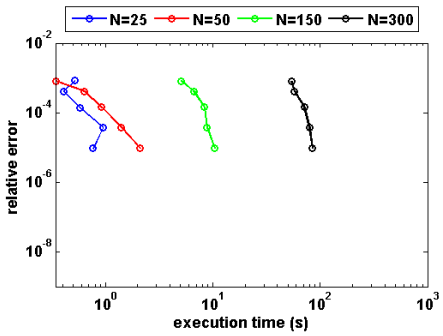
Table 5.17: Error for Allen-Cahn equation, 5th order



(a) KSS-EPI (b) Krylov-EPI



(c) KSS-EPI denoised (d) Leja



(e) AKP

Figure 5.16: Allen-Cahn equation, 4th order

(3.40), are given by

$$nf = \left[0.03\|\vec{\omega}\|^2 2 - \omega\|u-\bar{u}\| \quad 0.03\|\vec{\omega}\|^2 - \omega\|u-\bar{u}\|\right], \tag{5.25}$$

while the 4-th and 5-th order formula is

$$nf = \left[0.03\|\vec{\omega}\|^2 - \sqrt{2}\omega\|u-\bar{u}\| \quad 0.03\|\vec{\omega}\|^2 \quad 0.03\|\vec{\omega}\|^2 - \sqrt{2}\omega\|u-\bar{u}\|\right]. \tag{5.26}$$

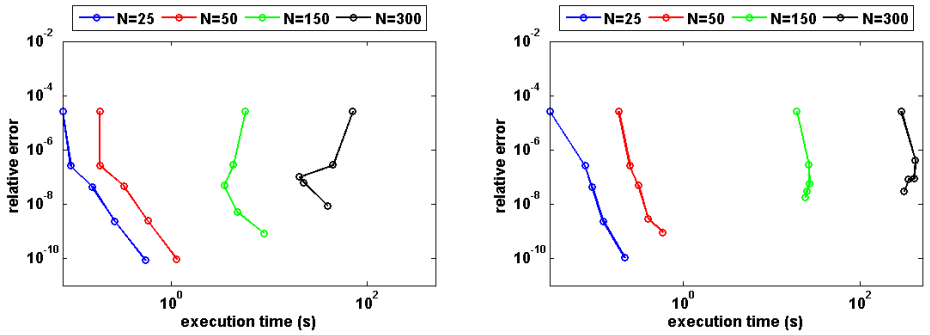
For this test problem, both the Krylov-EPI and KSS-EPI methods have similar error for the same time steps, with KSS-EPI being slightly more accurate for the smaller grid

Grid Size	Krylov-EPI	LEJA	AKP	KSS-EPI	KSS-EPI (denoised)
25	0.0312	0.4368	0.2496	0.0780	0.0936
	0.0780	0.4680	0.3432	0.0936	0.1560
	0.0936	0.8112	0.4992	0.1560	0.2340
	0.1248	1.5132	0.7956	0.2652	0.2028
	0.2184	3.1824	0.7020	0.5460	0.5616
50	0.1872	0.4212	0.2652	0.1872	0.1872
	0.2496	0.7488	0.5460	0.1872	0.2808
	0.3120	1.0608	0.7176	0.3276	0.4212
	0.4056	2.1216	1.2324	0.5772	0.7956
	0.5928	3.8688	1.9812	1.1388	1.5600
150	19.0477	5.3040	4.2276	5.7564	1.5756
	26.2550	5.6160	4.0560	4.3056	2.0592
	26.4110	7.3320	4.8828	3.4944	3.6660
	24.8510	8.2525	6.5520	4.7268	5.7096
	23.9462	10.6549	8.3773	8.8141	11.7157
300	288.3990	86.8302	39.9987	71.6825	6.8484
	416.7723	85.8630	38.9222	44.9439	8.7829
	402.1238	83.9285	44.1327	20.4049	14.0089
	346.5250	84.4172	50.5443	22.6045	25.2410
	310.2548	88.1094	58.0168	39.3435	50.0763

Table 5.18: Computational time for Allen-Cahn equation, 5th order

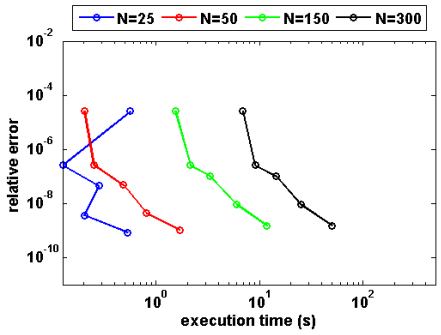
Grid Size	Krylov-EPI	LEJA	AKP	KSS-EPI	KSS-EPI (denoised)
25	13.3333	32.6667	14.3333	12.3333	12.0000
	10.6667	22.8333	10.1667	9.0000	8.6667
	8.6667	16.5833	7.1667	6.8333	6.8333
	7.0417	12.1250	5.2708	5.7083	5.7083
	6.4375	9.7919	3.8958	5.3750	5.4167
50	24.3333	97.6667	27.5000	16.0000	13.6667
	19.3333	41.0000	19.1667	10.5000	9.0000
	14.4167	29.0833	11.6667	7.0833	6.9167
	10.6250	20.5000	7.7083	5.7917	5.7917
	8.8750	14.7500	4.9271	5.5000	5.7083
150	64.6667	736.3333	231.1667	37.000	14.0000
	52.5000	364.6667	107.8333	20.8333	9.1667
	37.2500	186.6667	54.5833	9.8333	6.9167
	26.5417	84.1250	29.2083	6.2083	5.7083
	19.8542	34.3958	14.7083	5.7917	5.4167
300	121.0000	2.8133e+03	598.5000	63.6667	14.0000
	98.8333	1.3497e+03	327.6667	33.5000	9.1667
	71.5833	659.6667	173.8333	12.7500	6.7500
	49.0417	318.0833	90.8958	6.8333	5.6667
	34.2708	152.1250	46.2813	5.5417	5.4375

Table 5.19: Number of iterations for Allen-Cahn equation, 5th order

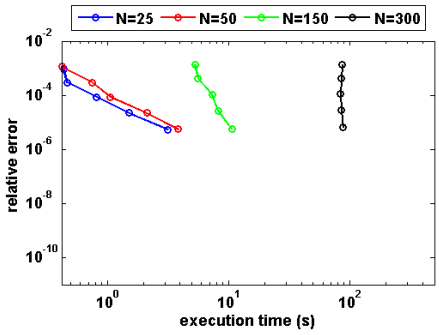


(a) KSS-EPI

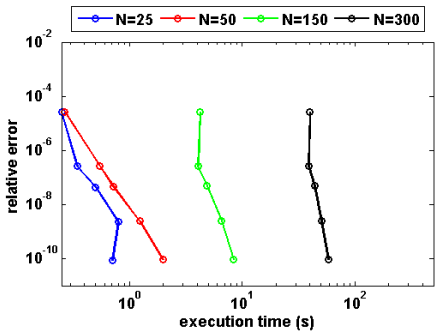
(b) Krylov-EPI



(c) KSS-EPI denoised



(d) Leja



(e) AKP

Figure 5.17: Allen-Cahn equation, 5th order

sizes $N = 250$ and $N = 500$, as can be seen in Tables 5.20, 5.44, and 5.26. The difference in computational time is almost insignificant for $N = 250$ in KSS-EPI and Krylov-EPI for both 3rd, 4th and 5th order, but as the grid size increases, we can see that KSS-EPI method is showing far superior efficiency compared to the Krylov-EPI method, as can be seen in Figures 5.18, 5.19 and 5.20. As shown in Tables 5.22, 5.25, and 5.28, this is again due to the increasing number of Krylov projection steps needed for Krylov-EPI.

The effect of denoising applied to KSS-EPI is even more pronounced than in the case

of the Allen-Cahn equation. For example, from Table 5.27, we can see that it takes only 4 seconds for KSS-EPI denoised to compute the solution for the first time step of the grid size $N = 3000$ for 5th order, while it takes Krylov-EPI 16,575 seconds to produce the same solution. A similar decrease is observed in the number of iterations, as seen in Tables 5.22, 5.25, and 5.28. It is worth mentioning that both Leja points 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 denoised, in terms of computational time and number of matrix-vector products.

Grid Size	Krylov-EPI	LEJA	AKP	KSS-EPI	KSS-EPI (denoised)
250	0.0161	1.7985e-05	0.0161	0.0161	0.0161
	0.0024	2.1581e-06	0.0024	0.0024	0.0024
	3.0029e-04	2.6364e-07	3.0027e-04	3.0029e-04	3.0028e-04
	3.5659e-05	3.2574e-08	3.5651e-05	3.5661e-05	3.5652e-05
	4.2649e-06	4.0665e-09	4.2603e-06	4.2619e-06	4.2616e-06
500	0.0161	4.3335e-05	0.0161	0.0161	0.0161
	0.0024	2.1643e-06	0.0024	0.0024	0.0024
	3.0117e-04	2.6444e-07	3.0113e-04	3.0117e-04	3.0113e-04
	3.5776e-05	3.2752e-08	3.5755e-05	3.5788e-05	3.5756e-05
	4.2861e-06	4.1306e-09	4.2726e-06	4.2890e-06	4.2738e-06
1500	0.0161	9.4851e-05	0.0161	0.0161	0.0161
	0.0024	2.2526e-05	0.0024	0.0024	0.0024
	3.0151e-04	4.9639e-06	3.0138e-04	3.0163e-04	3.0140e-04
	3.5875e-05	8.2588e-07	3.5786e-05	3.6182e-05	3.5796e-05
	4.3389e-06	5.3323e-09	4.2726e-06	4.6207e-06	4.2791e-06
3000	0.0161	1.0175e-04	0.0161	0.0161	0.0161
	0.0024	2.5492e-05	0.0024	0.0024	0.0024
	3.0169e-04	6.1770e-06	3.0141e-04	3.0209e-04	3.0147e-04
	3.6006e-05	1.4330e-06	3.5789e-05	3.8213e-05	3.5993e-05
	4.3883e-06	3.0784e-07	4.2766e-06	1.0413e-05	4.8464e-06

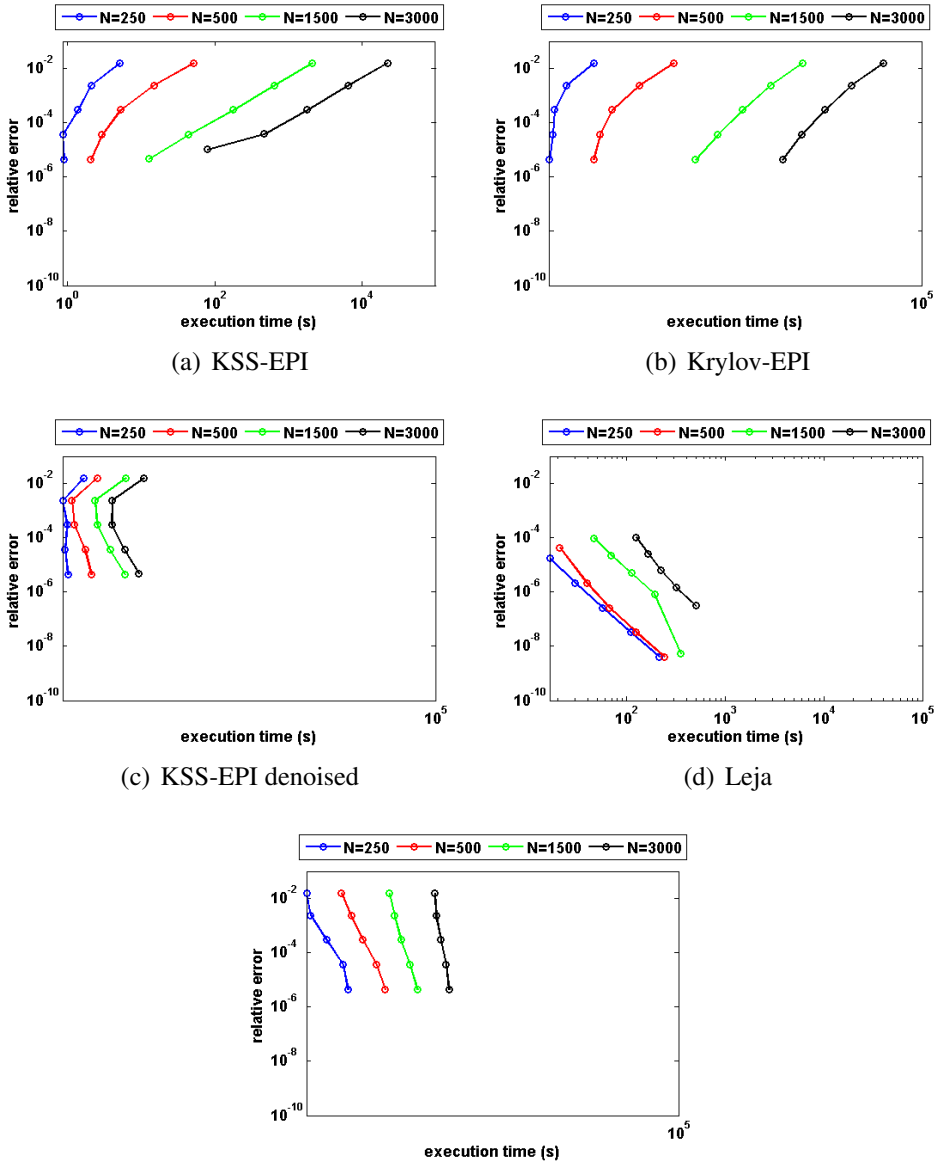
Table 5.20: Error for Burgers' equation, 3rd order

Grid Size	Krylov-EPI	LEJA	AKP	KSS-EPI	KSS-EPI (denoised)
250	6.7860	16.4113	1.1076	5.1480	2.1528
	3.0264	29.7494	1.2480	2.1060	1.1388
	2.1216	57.0964	2.0436	1.3884	1.3104
	2.0280	110.1055	3.4164	0.8580	1.2168
	1.8408	214.7510	3.9624	0.8892	1.3416
500	69.6388	20.8417	3.2136	52.0419	3.2760
	25.4282	40.2015	4.3836	14.9293	1.5132
	11.3881	67.4548	6.2556	5.2260	1.6380
	8.0029	125.2844	9.4849	2.9172	2.2620
	6.7080	241.9420	12.3085	2.0748	2.7144
1500	2.9888e+03	47.0187	14.1181	2.1350e+03	7.8781
	1.1803e+03	72.6809	16.4113	651.7722	3.0732
	523.0246	113.3347	20.2177	179.2763	3.2604
	249.3052	195.0481	26.4110	43.6023	4.8204
	130.1204	357.7415	33.0410	12.6829	7.6752
3000	3.2265e+04	125.0816	56.5348	2.2289e+04	13.6969
	1.2544e+04	165.0491	59.4052	6.5714e+03	5.1792
	5.8142e+03	229.5867	68.3440	1.8069e+03	5.1012
	2.9093e+03	325.1373	79.4825	473.0262	7.5348
	1.6709e+03	519.5301	87.3138	79.9973	11.7157

Table 5.21: Computational time for Burgers' equation,3rd order

Grid Size	Krylov-EPI	LEJA	AKP	KSS-EPI	KSS-EPI (denoised)
250	70.4500	31.6000	208.0000	64.0500	37.9000
	43.2250	23.2675	106.2750	34.9000	22.0500
	27.7625	16.5463	54.3375	18.1125	14.5125
	18.2938	12.5256	23.5563	9.4500	10.4438
	12.3281	9.4388	10.8094	6.4875	7.8250
500	135.1500	109.9000	543.2500	121.7000	43.4000
	82.5250	43.6200	282.8250	67.6000	22.8500
	52.6375	28.9275	150.3375	34.3375	14.6750
	34.6188	20.9913	74.2875	16.2250	10.5688
	22.7594	14.8294	34.9594	7.9750	7.9344
1500	381.7000	749.8550	2.8464e+03	338.0000	44.7000
	234.4500	364.9675	1.3123e+03	189.7500	23.1250
	149.9875	181.6525	701.3875	101.8375	14.8125
	96.9750	93.6288	385.8125	47.5125	10.7313
	64.4031	36.6297	182.2250	15.8844	8.0063
3000	734.4500	2.9607e+03	6.4046e+03	638.2000	44.7000
	452.8000	1.4244e+03	3.2878e+03	355.0750	23.1250
	291.6875	675.7688	1.8237e+03	192.9375	14.8875
	187.3000	314.5081	1.0106e+03	93.9813	10.6563
	125.6531	143.7350	499.4906	27.3531	8.0219

Table 5.22: Number of iterations for Burgers' equation,3rd order



(a) KSS-EPI

(b) Krylov-EPI

(c) KSS-EPI denoised

(d) Leja

(e) AKP

Figure 5.18: Burgers' equation, 3rd order

Grid Size	Krylov-EPI	LEJA	AKP	KSS-EPI	KSS-EPI (denoised)
250	0.0034	0.0104	0.0154	0.0034	0.0035
	3.0292e-04	0.0043	0.0055	3.1406e-04	3.3586e-04
	1.5942e-05	0.0014	0.0015	2.0542e-05	2.4377e-05
	9.2655e-07	3.7894e-04	3.7731e-04	8.1617e-07	1.5068e-06
	3.5075e-07	9.4171e-05	9.3398e-05	3.2615e-08	1.1855e-07
500	0.0033	0.0092	0.154	0.0033	0.0035
	2.6682e-04	0.0035	0.0055	2.9405e-04	3.3689e-04
	8.0746e-06	0.0010	0.0015	1.5345e-05	2.4217e-05
	3.3771e-06	3.0270e-04	3.7762e-04	5.8013e-07	1.5092e-06
	1.4000e-06	9.6171e-05	9.3475e-05	1.5045e-07	1.1901e-07
1500	0.0030	0.0088	0.0153	0.0031	0.0035
	1.9862e-04	0.0033	0.0055	2.4483e-04	3.3590e-04
	1.6140e-05	9.3069e-04	0.0015	8.2784e-06	2.4971e-05
	1.0921e-05	2.4351e-04	3.7771e-04	3.4392e-06	1.7710e-06
	5.5045e-06	6.8424e-05	9.3498e-05	1.5998e-06	1.3404e-07
3000	0.0028	0.0088	0.0153	0.0029	0.0035
	1.4683e-04	0.0033	0.0055	2.4860e-04	3.3705e-04
	3.5829e-05	9.1986e-04	0.0015	2.9881e-05	2.9361e-05
	2.3436e-05	2.3738e-04	3.7772e-04	1.5819e-05	8.4965e-06
	9.8058e-06	6.4545e-05	9.3500e-05	6.0488e-06	4.2095e-06

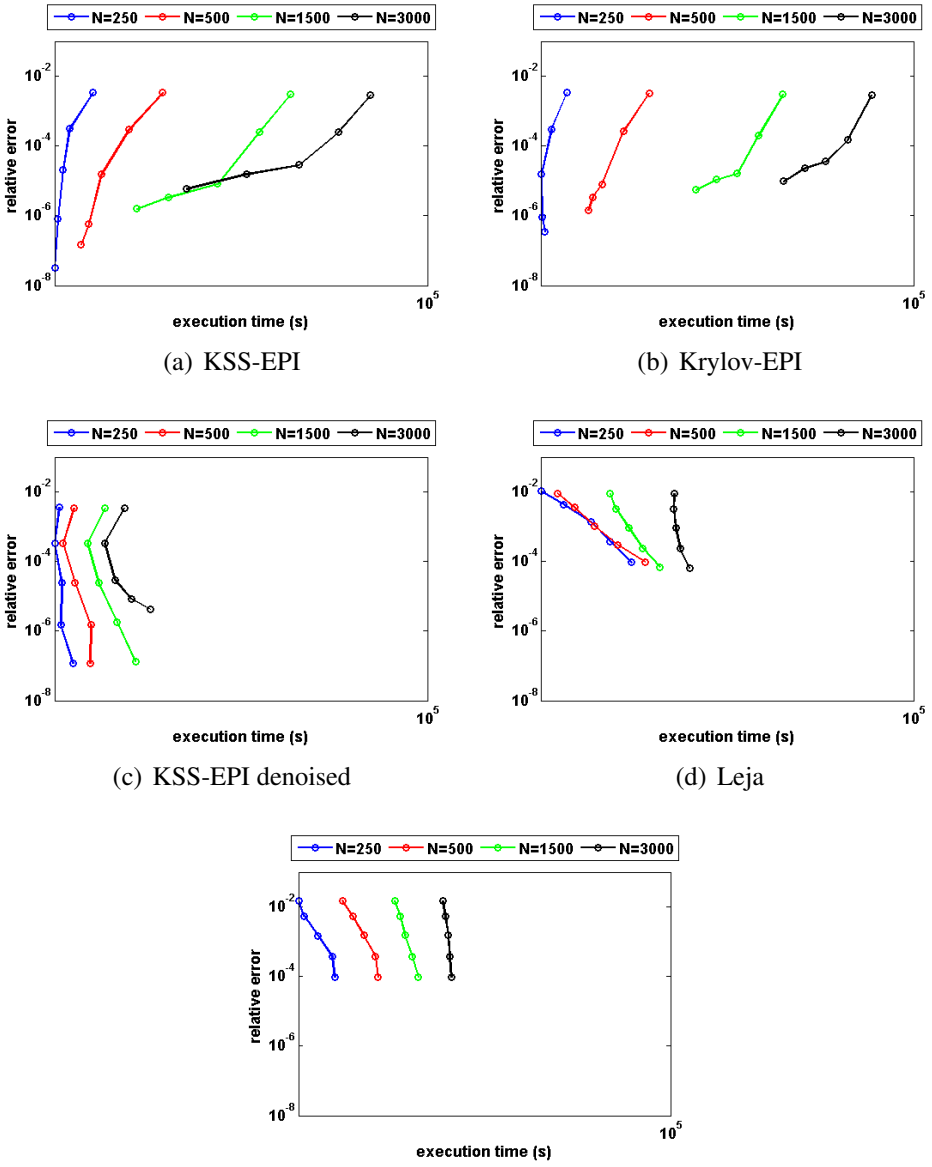
Table 5.23: Error for Burgers, 4th order

Grid Size	Krylov-EPI	LEJA	AKP	KSS-EPI	KSS-EPI (denoised)
250	6.9888	7.7376	3.1980	4.6800	1.3260
	4.5240	13.6033	3.7284	2.3244	1.1544
	3.4788	27.2222	5.5224	1.9344	1.4352
	3.5256	44.0703	8.1433	1.6536	1.3884
	3.8064	76.2533	8.7829	1.4976	2.0124
500	67.1428	11.4661	10.8109	37.3622	2.0592
	33.0566	18.0649	14.3833	13.6345	1.4976
	18.2209	29.4842	19.7653	6.0060	2.1528
	14.0713	53.0871	27.2690	4.1340	3.5100
	12.5425	106.8763	29.2814	3.2760	3.3852
1500	2.6684e+03	44.2419	46.4259	1.6902e+03	5.3508
	1.3589e+03	51.0123	54.0855	662.8170	3.1356
	759.8185	70.5749	61.9948	191.8032	4.4304
	432.0136	100.3398	75.5357	44.8659	7.7220
	240.8499	154.9402	90.1374	17.3629	13.6189
3000	3.0659e+04	223.0346	176.2499	1.8186e+04	9.6409
	1.5935e+04	221.8490	189.4476	6.9875e+03	5.2884
	8.7187e+03	233.0343	207.4033	2.1565e+03	7.2852
	4.8650e+03	260.6777	214.7978	458.6585	11.9185
	2.7317e+03	331.7673	224.6570	76.9709	21.2005

Table 5.24: Computation time for Burgers, 4th order

Grid Size	Krylov-EPI	LEJA	AKP	KSS-EPI	KSS-EPI (denoised)
250	65.3667	366.6333	186.4333	55.1333	27.0667
	43.2333	173.1833	96.5750	32.7833	16.0167
	28.4167	61.2833	46.8583	18.0000	11.0917
	19.2292	34.2458	18.0167	9.5542	8.3292
	12.9688	20.7146	7.1573	6.1000	6.6104
500	125.5333	1.3257e+03	498.5000	107.0000	29.3000
	82.2167	590.0667	263.0917	61.7833	16.4833
	54.5917	264.8083	136.2042	31.8583	11.2583
	36.9875	109.1325	66.4104	13.8208	8.4083
	25.0083	36.1979	23.8927	6.8313	6.6417
1500	353.9333	1.1760e+04	2.3576e+03	303.8000	30.0667
	233.2667	5.2436e+03	1.2660e+03	178.3500	16.6667
	156.4750	2.3059e+03	629.9333	93.5500	11.4250
	105.1083	991.7125	311.0938	38.3042	8.4875
	68.7813	391.4313	139.7115	13.3896	6.6813
3000	679.1333	4.8977e+04	6.0456e+03	579.8667	30.1000
	454.6500	2.2547e+04	3.1899e+03	348.7167	16.6333
	3047500.	1.0100e+04	1.7028e+03	186.4583	11.3417
	196.5667	4.3356e+03	811.3792	73.0500	8.5583
	125.7771	1.7993e+03	362.4531	21.0833	6.7500

Table 5.25: Number of iterations for Burgers, 4th order



(a) KSS-EPI

(b) Krylov-EPI

(c) KSS-EPI denoised

(d) Leja

(e) AKP

Figure 5.19: Burgers' equation, 4th order

Grid Size	Krylov-EPI	LEJA	AKP	KSS-EPI	KSS-EPI (denoised)
250	2.3480e-04	1.4120e-04	1.4018e-04	2.1824e-04	1.9903e-04
	2.1374e-05	3.4971e-05	4.9610e-06	1.8124e-05	1.0066e-05
	3.2881e-06	8.6913e-06	1.8102e-07	2.2339e-06	1.1042e-06
	9.7105e-07	2.1657e-06	6.5889e-09	3.9863e-07	1.7425e-07
	2.3316e-07	5.4051e-07	2.2893e-10	5.1162e-08	8.5376e-08
500	2.7689e-04	1.5929e-04	1.4070e-04	2.6430e-04	2.0282e-04
	3.4008e-05	3.4999e-05	4.9926e-06	2.6770e-05	1.1991e-05
	7.2781e-06	8.6982e-06	1.8248e-07	4.1766e-06	1.0005e-06
	2.0652e-06	2.1674e-06	6.6462e-09	1.1006e-06	1.8249e-07
	5.1971e-07	5.4089e-07	2.3109e-10	1.4746e-07	8.8101e-08
1500	3.8221e-04	1.8320e-04	1.4086e-04	3.6507e-04	2.1540e-04
	6.4847e-05	4.3995e-05	5.0020e-06	5.6508e-05	1.1803e-05
	1.6690e-05	1.0499e-05	1.8292e-07	1.1101e-05	4.2042e-06
	4.6827e-06	2.4241e-06	6.6635e-09	2.8963e-06	9.9159e-07
	1.9779e-06	5.4041e-07	2.3203e-10	1.1776e-06	1.0522e-07
3000	4.8117e-04	1.8672e-04	1.4087e-04	4.6786e-04	2.0923e-04
	9.2039e-05	4.5690e-05	5.0029e-06	8.9693e-05	1.4885e-05
	2.4561e-05	1.1160e-05	1.8296e-07	2.3928e-05	1.2566e-05
	7.7607e-06	2.6935e-06	6.6649e-09	1.0537e-05	8.3789e-06
	3.7903e-06	6.2979e-07	2.3174e-10	5.3535e-06	4.1938e-06

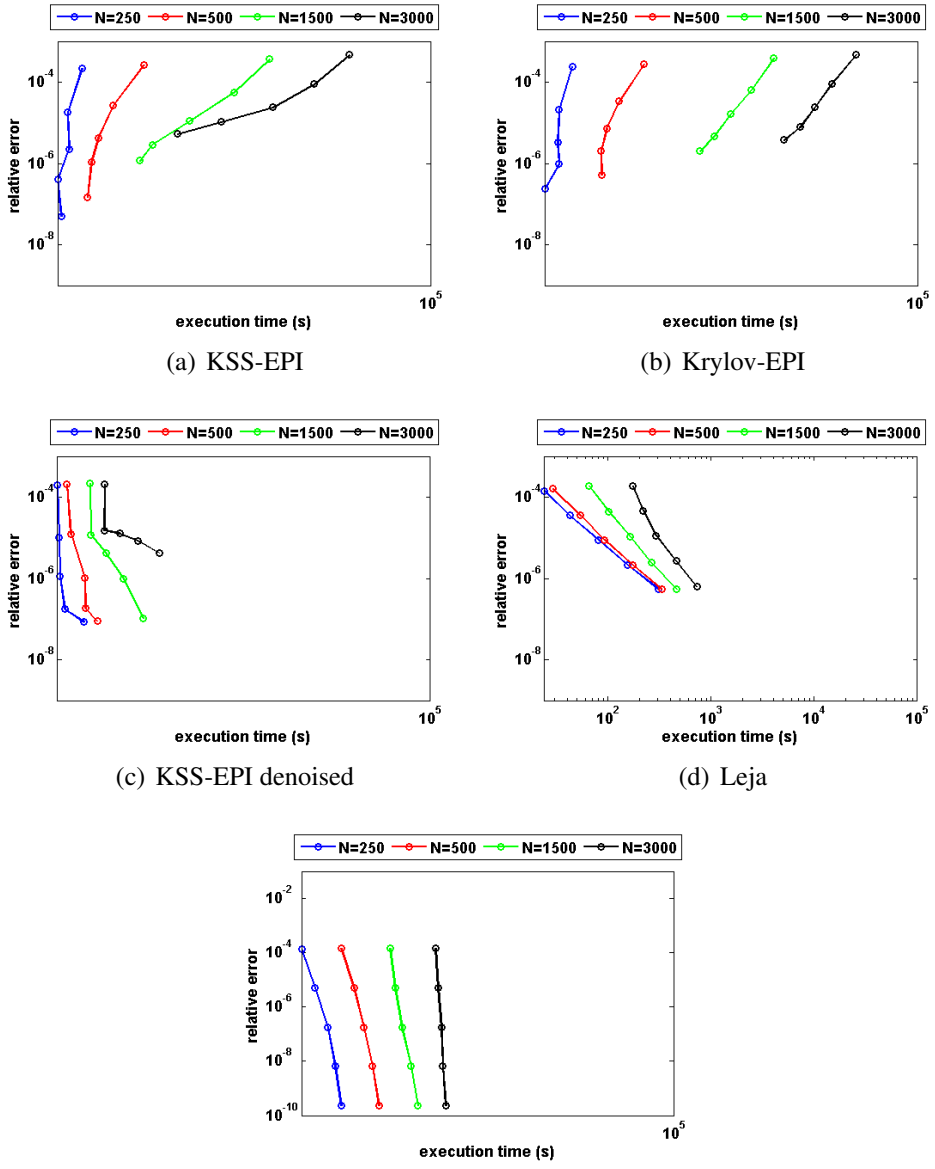
Table 5.26: Error for Burgers' equation, 5th order

Grid Size	Krylov-EPI	LEJA	AKP	KSS-EPI	KSS-EPI (denoised)
250	3.8220	24.2426	2.2308	2.6052	1.0140
	2.6052	42.6819	3.2916	1.7004	1.0608
	2.5116	80.4029	4.7424	1.7628	1.0620
	2.6052	154.2538	5.8500	1.2480	1.2636
	1.7004	308.5388	6.9888	1.3884	2.2776
500	31.7774	29.1878	6.9576	16.9261	1.3416
	15.3505	53.8515	10.0777	6.7080	1.5288
	10.6181	92.4930	13.3537	4.2744	2.3400
	8.9701	173.0675	17.2069	3.4476	2.3712
	9.1261	330.4569	20.7949	3.0732	3.4320
1500	1.4553e+03	65.0992	28.5794	763.6405	2.7456
	753.9060	102.0715	32.9942	262.1129	2.8392
	411.0626	163.4890	40.3419	67.2988	4.4772
	250.4752	262.3001	52.1355	21.9493	7.7844
	164.9867	464.5086	63.8512	14.9605	14.2429
3000	1.6575e+04	173.9879	105.4411	8.6653e+03	4.3368
	8.0333e+03	218.2142	113.0539	2.9622e+03	4.2744
	4.8663e+03	293.2195	124.8008	852.3895	6.9420
	3.1460e+03	457.7225	130.7600	179.3543	12.0121
	1.9679e+03	724.0006	141.6957	47.4867	23.2909

Table 5.27: Computational time for Burgers' equation, 5th order

Grid Size	Krylov-EPI	LEJA	AKP	KSS-EPI	KSS-EPI (denoised)
250	48.9000	23.9833	114.8667	39.2333	19.4333
	32.7833	17.4133	58.4083	22.8667	12.5000
	22.1833	12.3383	25.5625	13.0000	9.2417
	15.0708	9.5792	10.9208	7.2583	7.3375
	10.6354	7.9671	5.3167	5.5229	5.9875
500	93.7667	70.7767	314.6333	75.2000	19.9000
	62.7167	31.5333	167.1917	42.8667	12.6333
	41.6167	21.5267	78.0750	22.6417	9.3500
	28.3667	15.0625	36.2188	10.2083	7.3917
	19.8000	10.3608	14.4177	6.0438	5.9938
1500	269.0333	497.5800	1.5025e+03	218.2333	20.0000
	175.8833	250.2800	766.3833	125.3500	12.7000
	118.2500	119.7742	390.3625	62.2250	9.3750
	81.9958	58.9100	195.9667	26.6417	7.4000
	56.4000	24.9115	88.3052	10.4833	6.0375
3000	521.6333	1.9266e+03	3.8573e+03	422.9667	20.0000
	341.1667	917.4833	2.0312e+03	244.3833	12.6833
	232.9917	428.4083	1.0655e+03	124.6000	9.4167
	160.5458	202.3358	485.5333	49.5542	7.3833
	108.2063	90.3694	220.4406	16.0188	6.1271

Table 5.28: Number of iterations for Burgers' equation, 5th order



(a) KSS-EPI

(b) Krylov-EPI

(c) KSS-EPI denoised

(d) Leja

(e) AKP

Figure 5.20: Burgers' equation, 5th order

5.2.4 ADR Problem

The next test problem is the one-dimensional advection-diffusion-reaction equation

$$u_t = \varepsilon u_{xx} - \alpha u_x + \gamma u(u - \frac{1}{2})(1-u) \quad x \in [0,1], t \in [0, 0.2] \tag{5.27}$$

with $\varepsilon = 1/100$, $\alpha = -10$, and $\gamma = 1$. We used periodic boundary conditions with initial conditions given by

$$u_0(x) = 256(x(1-x))^2 + 0.3.$$

For KSS-EPI, the low-frequency portion $\mathbf{b}_L$ consists of all components with wave numbers $|\omega| \leq 75$. The N_c value is 20. The formula for the 3-rd order frequency-dependent nodes, as defined in (3.40) is

$$nf = \left[0.01\|\vec{\omega}\|^2 + \bar{q} + i(10\omega - i\|\tilde{q}\|) \quad 0.01\|\vec{\omega}\|^2 + \bar{q} - i(10\omega - i\|\tilde{q}\|)\right], \tag{5.28}$$

where $q = 3u - 0.5 - 3u^2$ is obtained from the Jacobian of (5.27). The corresponding 4-th and 5-th order frequency-dependent nodes are computed as follows

$$nf = \left[0.01\|\vec{\omega}\|^2 + \bar{q} + i(10\omega - i\|\tilde{q}\|) \quad 0.01\|\vec{\omega}\|^2 + \bar{q} \quad 0.01\|\vec{\omega}\|^2 + \bar{q} - i(10\omega - i\|\tilde{q}\|)\right] \tag{5.29}$$

For this test problem, just like with the Allen-Cahn equation, both Krylov-EPI and KSS-EPI yield approximately the same accuracy for 3rd-, 4th- and 5th-order accuracy, as can be seen from Tables 5.29, 5.32, and 5.35, for the most part (Krylov-EPI is more accurate than KSS-EPI for the same time steps with $N = 3000$ grid points with the 5th-order). From Tables 5.30, 5.33, and 5.36, we can see that while Krylov-EPI is faster at the smaller grid sizes, there is a much greater increase in needed Krylov projection steps for Krylov-EPI as N increases. However, for this problem, Tables 5.31, 5.34, and 5.37 show that Krylov-EPI does not need as many projection steps compared to the other test problems, so the advantage is much less pronounced. Also, since the number of iterations required to converge to the solution is small, the extra computational cost inquired by applying denoising to KSS-EPI is greater than the time saved from the reduction of the number of iterations, so KSS-EPI denoised was not added to the result tables. The AKP and Leja methods show similar errors to KSS-EPI and Krylov-EPI for both 3rd and 5th orders and sligtly worse error for 4th order, as can be seen from Tables 5.29, 5.32, and 5.35. However, the Leja method is slower than both Krylov-EPI and KSS-EPI for smaller grid sizes for 3rd and 5th orders, and only slightly faster than KSS-EPI for $N = 1500$ and $N = 3000$ for 3rd order, and $N = 3000$ for 5th order. On the other hand, AKP seems to be faster than both KSS-EPI and Krylov-EPI for all grid sizes for 3rd and 5th orders. Nevertheless, Table 5.33 shows that KSS-EPI is faster than both AKP and Leja for 4th order.

Grid Size	Krylov-EPI	LEJA	AKP	KSS-EPI
250	0.0019	0.0019	0.0019	0.0019
	2.1800e-04	2.1800e-04	2.1800e-04	2.1800e-04
	2.5204e-05	2.5204e-05	2.5204e-05	2.5204e-05
	2.9869e-06	2.9869e-06	2.6869e-06	2.9869e-06
	3.6230e-07	3.6085e-07	3.6230e-07	3.6230e-07
500	0.0019	0.0019	0.0019	0.0019
	2.1800e-04	2.1800e-04	2.1800e-04	2.1800e-04
	2.5204e-05	2.5204e-05	2.5204e-05	2.5216e-05
	2.9869e-06	2.9874e-06	2.9869e-06	2.9869e-06
	3.6229e-07	3.6254e-07	3.6229e-07	3.6229e-07
1500	0.0019	9.2211e-04	0.0037	0.0019
	2.1800e-04	2.1800e-04	2.1800e-04	2.1800e-04
	2.5204e-05	2.5203e-05	2.5204e-05	2.5205e-05
	2.9867e-06	2.9874e-06	2.9869e-06	2.9876e-06
	3.6226e-07	3.6232e-07	3.6229e-07	3.6231e-07
3000	0.0019	0.0025	0.0072	0.0019
	2.1800e-04	5.1477e-04	2.5725e-04	2.1800e-04
	2.5203e-05	9.5135e-05	2.5480e-05	2.5220e-05
	2.9867e-06	2.9854e-06	2.9869e-06	3.0204e-06
	3.6228e-07	3.6092e-07	3.6229e-07	3.9264e-07

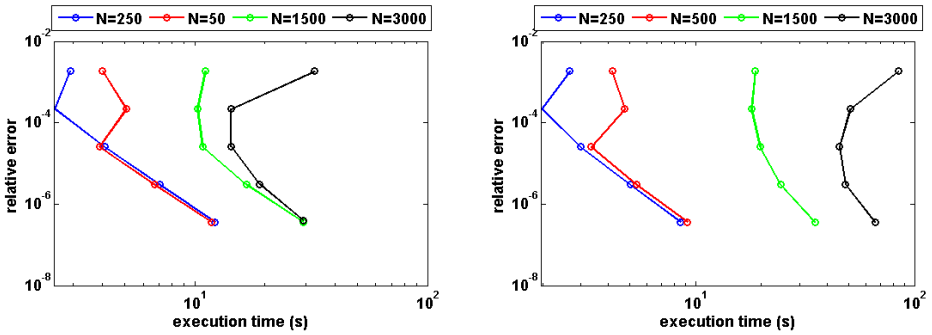
Table 5.29: Error for ADR, 3th order

Grid Size	Krylov-EPI	LEJA	AKP	KSS-EPI
250	2.6676	2.0904	0.5772	2.9172
	1.9812	4.0872	0.5772	2.4960
	2.9796	7.2852	0.9828	4.1028
	5.0388	14.4769	1.7472	7.0980
	8.5333	29.6090	3.1356	12.2149
500	4.1652	1.6380	0.9048	4.0092
	4.7580	3.2916	1.1544	5.0700
	3.3384	6.2400	1.0296	3.8844
	5.3820	11.7157	1.7472	6.7236
	9.2041	23.0101	3.2136	11.7781
1500	18.7513	2.3556	1.5132	11.0917
	17.9869	4.1184	2.9796	10.2493
	19.6717	7.4880	4.1964	10.7953
	24.4766	13.3069	5.9280	16.6297
	35.1002	25.3034	9.0949	29.2658
3000	84.5837	3.6036	3.8844	32.7914
	50.9655	5.9904	4.5552	14.3209
	45.3963	9.6409	8.2837	14.3209
	48.4383	16.7233	11.2321	18.9697
	66.0820	30.8414	15.4129	29.1566

Table 5.30: Computation time for ADR, 3th order

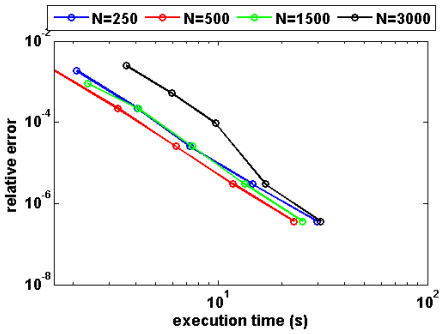
Grid Size	Krylov-EPI	LEJA	AKP	KSS-EPI
250	10.6500	18.0000	12.9500	10.6000
	7.9750	14.0000	9.1250	7.9500
	6.6625	12.0000	7.0750	6.6625
	5.8438	11.2500	5.5625	5.8438
	5.0000	10.0625	5.0313	5.0000
500	15.4500	22.5000	18.4500	13.6000
	9.4750	20.1250	10.9500	8.7250
	6.7250	13.3375	7.1750	6.7375
	5.8500	10.2375	5.6250	5.8438
	5.0000	10.3563	5.0375	5.0000
1500	39.6500	104.4500	32.4500	28.5000
	22.8500	39.2750	27.3250	13.7250
	13.8000	27.9125	16.3875	7.7750
	8.5938	19.3938	10.0188	5.8500
	6.2219	14.4750	6.5969	5.0000
3000	75.7000	333.2000	52.1000	49.3500
	41.7500	176.8000	35.520	18.7000
	25.7750	98.9250	37.0125	9.9625
	15.5125	37.8938	23.3625	6.0313
	11.0094	27.2250	11.9906	5.0000

Table 5.31: Number of iterations for ADR, 3th order

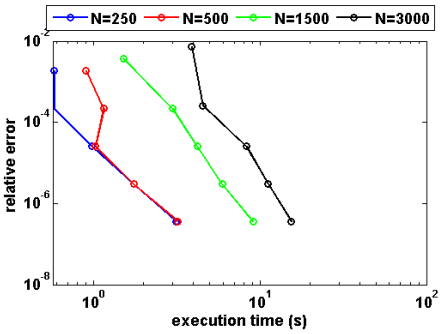


(a) KSS-EPI

(b) Krylov-EPI



(c) Leja



(d) AKP

Figure 5.21: ADR, 3rd order

Grid Size	Krylov-EPI	LEJA	AKP	KSS-EPI
250	0.0013	0.0052	0.0052	0.0013
	1.0548e-04	0.0016	0.0016	1.0548e-04
	9.3359e-06	4.1534e-04	4.1534e-04	9.3359e-06
	9.3797e-07	1.0412e-04	1.0412e-04	9.3797e-07
	1.0379e-07	2.5949e-05	2.5949e-05	1.0379e-07
500	0.0013	0.0052	0.0052	0.0013
	1.0548e-04	0.0016	0.0016	1.0548e-04
	9.3359e-06	4.1536e-04	4.1536e-04	9.3359e-06
	9.3798e-07	1.0412e-04	1.0412e-04	9.3798e-07
	1.0379e-07	2.5950e-05	2.5950e-05	1.0379e-07
1500	0.0013	0.0040	0.0060	0.0013
	1.0548e-04	0.0016	0.0016	1.0548e-04
	9.3360e-06	4.1536e-04	4.1536e-04	9.3371e-06
	9.3817e-07	1.0413e-04	1.0413e-04	9.3827e-07
	1.0381e-07	2.5950e-05	2.5950e-05	1.0379e-07
3000	0.0013	0.0033	0.0086	0.0013
	1.0548e-04	0.0012	0.0016	1.0548e-04
	9.3366e-06	3.4221e-04	4.1570e-04	9.3468e-06
	9.3810e-07	1.0413e-04	1.0413e-04	9.5171e-07
	1.0375e-07	2.5950e-05	2.5950e-05	1.0605e-07

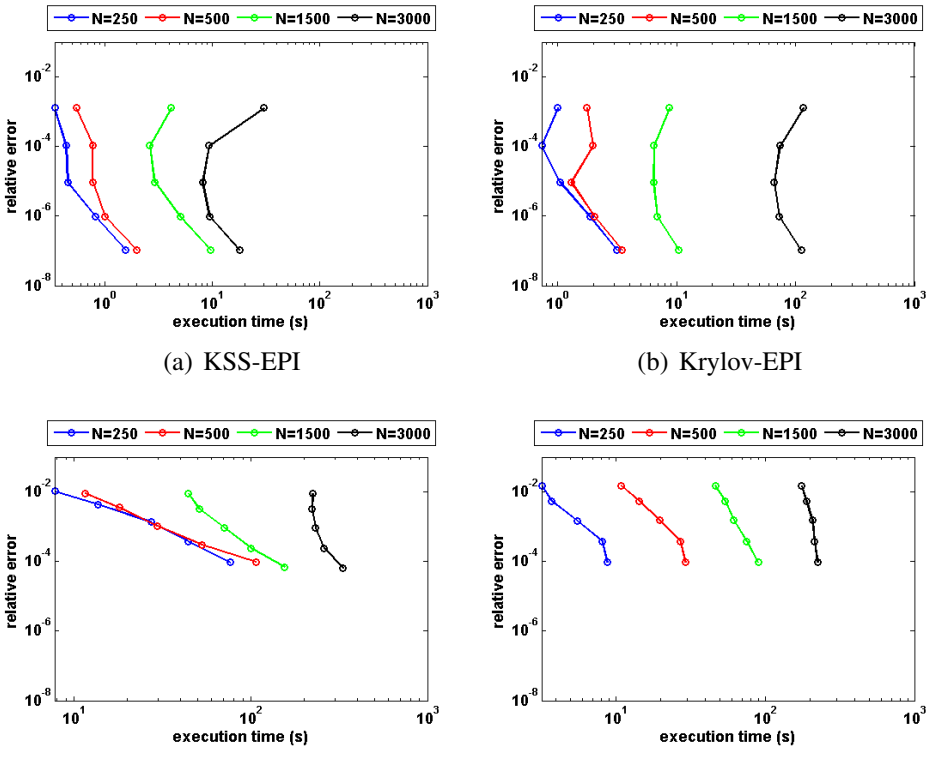
Table 5.32: Error for ADR, 4th order

Grid Size	Krylov-EPI	LEJA	AKP	KSS-EPI
250	0.9984	5.3196	1.3416	0.3432
	0.7332	9.5629	1.9032	0.4368
	1.0452	17.3629	2.3556	0.4524
	1.8720	35.4746	4.1652	0.8268
	3.1512	73.1117	6.8796	1.5600
500	1.7628	3.8532	2.3556	0.5460
	1.9812	7.9249	2.1216	0.7800
	1.2948	16.2553	2.5584	0.7800
	2.0436	29.7338	4.1808	1.0140
	3.4788	57.9232	7.3320	1.9968
1500	8.7361	5.3508	3.7128	4.1652
	6.4428	9.7969	6.7080	2.6364
	6.4116	17.5345	9.6097	2.9172
	6.8952	34.5542	14.1961	5.0544
	10.4209	63.4300	21.0601	9.7345
3000	116.5483	9.6877	7.3632	30.2486
	74.0537	14.0557	10.0777	9.3289
	66.4408	23.2753	17.6281	8.1745
	72.8993	41.0439	26.1926	9.4693
	111.4627	75.1301	37.5962	17.9401

Table 5.33: Computation time for ADR, 4th order

Grid Size	Krylov-EPI	LEJA	AKP	KSS-EPI
25	10.4000	31.0667	12.4800	10.4333
	7.8500	22.2667	8.4600	7.8333
	6.5500	18.3167	6.5150	6.5500
	5.7917	16.7167	4.9950	5.7917
	5.0000	15.5250	4.2963	5.0000
50	14.6000	37.9000	17.7600	12.9333
	8.9833	32.8667	10.0200	8.4167
	6.6000	21.1417	6.5700	5.5667
	5.8000	16.1833	5.1000	5.7958
	5.0000	16.4938	4.4475	5.0000
150	36.4333	162.7000	32.4800	27.7667
	20.9167	64.4333	25.3700	13.4000
	12.6500	43.5333	15.2000	7.1250
	8.2125	30.1667	9.4925	5.7958
	6.4217	21.4063	6.2550	5.0000
300	69.5333	526.9000	46.5200	46.4000
	37.7833	271.8167	32.8800	19.4500
	23.5333	152.6917	32.2000	10.3917
	14.5917	61.0583	23.1325	5.2625
	11.1979	41.0000	12.1238	5.0000

Table 5.34: Number of iterations for ADR, 4th order



(a) KSS-EPI (b) Krylov-EPI

(c) Leja (d) AKP

Figure 5.22: ADR, 4th order

Grid Size	Krylov-EPI	LEJA	AKP	KSS-EPI
250	1.2834e-04	1.2834e-04	1.2834e-04	1.2834e-04
	4.4759e-06	4.4759e-06	4.4759e-06	4.4759e-06
	1.3486e-07	1.3484e-07	1.3486e-07	1.3486e-07
	4.0011e-09	3.9782e-09	4.0014e-09	4.0011e-09
	1.2170e-10	1.0517e-10	1.2106e-10	1.2168e-10
500	1.2834e-04	1.2834e-04	1.2834e-04	1.2834e-04
	4.4758e-06	4.4759e-06	4.4759e-06	4.5187e-06
	1.3485e-07	1.3487e-07	1.3486e-07	1.3486e-07
	4.0012e-09	4.0038e-09	4.0013e-09	4.0010e-09
	1.2270e-10	1.2200e-10	1.2101e-10	1.2174e-10
1500	1.2834e-04	4.9928e-04	0.0033	1.2834e-04
	4.4758e-06	4.4758e-06	4.4758e-06	4.4842e-06
	1.3479e-07	1.3488e-07	1.3486e-07	2.0019e-07
	4.4881e-09	3.9953e-09	4.0013e-09	2.3433e-08
	4.3802e-10	1.3504e-10	1.2097e-10	3.2315e-10
3000	1.2834e-04	0.0012	0.0069	1.2834e-04
	4.4756e-06	1.0973e-04	1.3605e-04	4.5510e-06
	1.3442e-07	1.3480e-07	2.7834e-06	4.7402e-07
	5.0917e-09	3.9650e-09	4.0013e-09	1.5502e-07
	8.9753e-10	1.2865e-10	1.2129e-10	2.1472e-08

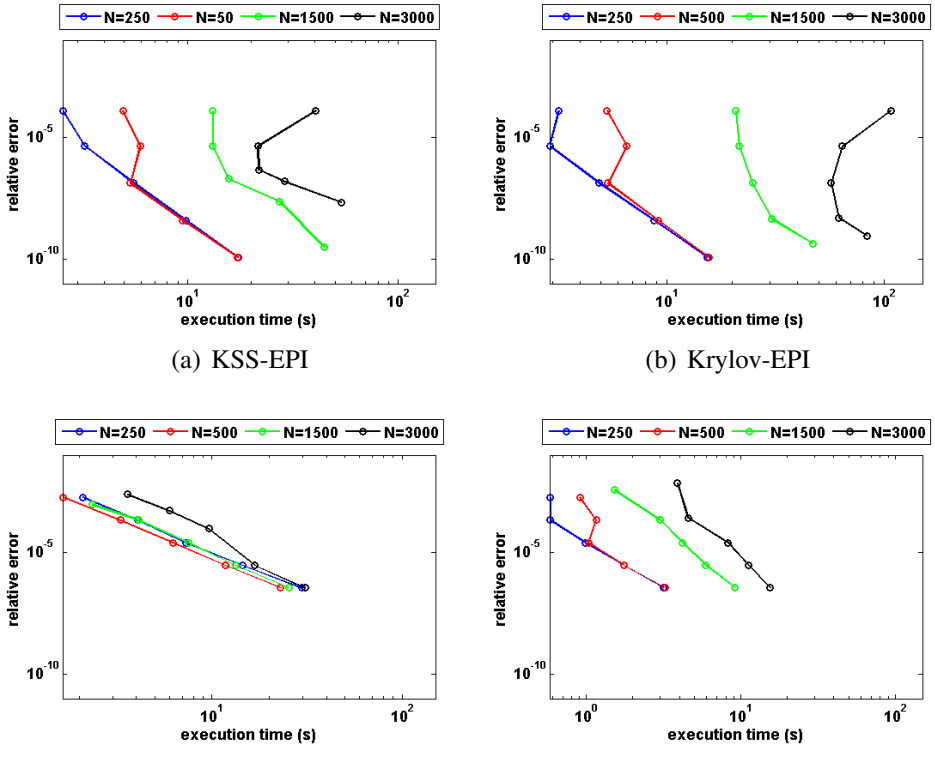
Table 5.35: Error for ADR, 5th order

Grid Size	Krylov-EPI	LEJA	AKP	KSS-EPI
250	3.1824	5.8188	1.0140	2.5428
	2.9016	11.3881	1.4508	3.2292
	4.8984	22.1521	2.5584	5.5380
	8.7673	43.8363	4.2744	9.8281
	15.3349	86.9394	7.4412	17.3005
500	5.3196	4.5396	2.0436	4.9140
	6.5520	9.2197	2.3712	5.9280
	5.3664	17.9401	2.6364	5.3196
	9.1261	35.0846	4.4460	9.4537
	15.6157	67.5172	7.6128	17.2537
1500	20.8261	6.3180	4.8672	13.1509
	21.5593	11.1853	5.9904	13.1353
	24.8666	20.4673	9.0325	15.6313
	30.4826	39.7179	13.9309	27.2222
	46.8783	75.5357	20.5921	44.3667
3000	107.2039	9.9373	10.0153	50.5603
	64.3972	16.2709	13.1821	21.5437
	57.1432	25.8650	21.9805	21.7465
	62.0884	45.1155	24.2582	28.9694
	83.0861	84.8489	34.8818	53.4459

Table 5.36: Computation time for ADR, 5th order

Grid Size	Krylov-EPI	LEJA	AKP	KSS-EPI
250	9.2000	29.8000	9.2167	9.1333
	7.2333	23.5333	7.0583	7.2167
	6.1417	20.7500	5.5750	6.1417
	5.5167	19.1167	4.3813	5.5167
	4.8458	17.5500	3.7885	4.8396
500	12.1333	36.9000	11.5500	10.9667
	8.1667	28.4333	7.5417	7.5333
	6.1833	21.9750	5.6667	6.1500
	5.5250	20.4542	4.5083	5.5208
	4.8396	19.2625	3.8240	4.8583
1500	29.5333	120.9333	33.1333	21.3000
	17.4333	57.5667	16.5083	11.2167
	11.0167	39.1583	10.2875	6.5917
	7.4250	28.2833	6.8271	5.5208
	5.6979	21.3375	4.5802	4.8417
3000	55.1000	401.3000	51.5500	34.9667
	31.6000	179.3000	34.4833	15.9500
	19.7333	89.0250	29.7042	8.5250
	12.8917	54.5083	13.8021	5.2583
	9.4979	37.2708	8.0125	4.8271

Table 5.37: Number of iterations for ADR, 5th order



(a) KSS-EPI (b) Krylov-EPI

(c) Leja (d) AKP

Figure 5.23: ADR, 5th order

5.2.5 2D ADR Problem

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

$$u_t = \varepsilon(u_{xx}+u_{yy}) - \alpha(u_x+u_y) + \gamma u(u-\frac{1}{2})(1-u) \quad x,y \in [0,1], t \in [0,0.1] \tag{5.30}$$

with $\varepsilon = 1$, $\alpha = -10$, and $\gamma = 1$. We used homogeneous Neumann 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 $|\omega| \leq N_c = 20$. The formula for the 3-rd frequency-dependent order nodes, as defined in (3.93), is given by

$$nf = \left[\|\vec{\omega}\|^2 + 10i(\omega_1+\omega_2) + \bar{\phi} + \mu \quad \|\vec{\omega}\|^2 + 10i(\omega_1+\omega_2) + \bar{\phi} - \mu\right], \tag{5.31}$$

where $\mu = \sqrt{\frac{\|\tilde{\phi}\|^2}{4\pi^2} + \frac{4\|\vec{\omega}\cdot\nabla\phi\|^2}{\|\tilde{\phi}\|^2}}$ and $\phi = u(u-\frac{1}{2})(1-u)$. The formula for the 4-th and 5-th order frequency-dependent nodes, as defined in (3.94), is

$$nf = \begin{bmatrix} \|\vec{\omega}\|^2 + 10i(\omega_1+\omega_2) + \bar{\phi} \\ \|\vec{\omega}\|^2 + 10i(\omega_1+\omega_2) + \bar{\phi} + \mu \\ \|\vec{\omega}\|^2 + 10i(\omega_1+\omega_2) + \bar{\phi} - \mu \end{bmatrix} \tag{5.32}$$

For this test problem, all four methods yield approximately the same accuracy, but both Leja interpolation and Adaptive Krylov are more efficient than Krylov-EPI and KSS-EPI, as we can see from Table 5.39. While Krylov-EPI is faster than KSS-EPI at the smaller grid sizes ($N = 25, 50$), we again observe the much greater increase in needed Krylov projection steps (Table 5.40) 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. Also, since KSS-EPI does not require many iterations to converge to the solution, denoising applied to KSS-EPI did not reduce the number of iterations enough, but added extra computational time. Therefore, KSS-EPI denoised was not added to Tables 5.38, 5.39, and 5.40.

Grid Size	Krylov-EPI	LEJA	AKP	KSS-EPI
25	0.0227	0.0227	0.0227	0.0227
	0.0027	0.0027	0.0027	0.0027
	3.2241e-04	3.2237e-04	3.2237e-04	3.2241e-04
	3.9533e-05	3.9524e-05	3.9526e-05	3.9533e-05
	4.8917e-06	4.8898e-06	4.8914e-06	4.8917e-06
50	0.0251	0.0251	.0251	0.0251
	0.0030	0.0030	0.0030	0.0030
	3.5600e-04	3.5595e-04	3.5595e-04	3.5598e-04
	4.3623e-05	4.3616e-05	4.3616e-05	4.3623e-05
	5.3965e-06	5.3957e-06	5.3958e-06	5.3965e-06
150	0.0259	0.0259	0.0259	0.0259
	0.0030	0.0030	0.0030	0.0030
	3.6695e-04	3.6657e-04	3.6655e-04	3.6660e-04
	4.4984e-05	4.4915e-05	4.4907e-05	4.4915e-05
	5.5678e-06	5.5538e-06	5.5549e-06	5.5558e-06
300	0.0259	0.0274	0.0259	0.0260
	0.0031	0.0047	0.0031	0.0031
	3.6847e-04	3.6760e-04	3.6756e-04	3.6804e-04
	4.5202e-05	4.5011e-05	4.5030e-05	4.5052e-05
	5.8755e-06	5.5877e-06	5.5701e-06	5.5710e-06

Table 5.38: Error for 2D-ADR, 3rd order

Grid Size	Krylov-EPI	LEJA	AKP	KSS-EPI
25	0.8112	13.2601	2.3556	1.1388
	1.2792	26.5826	3.2136	1.7940
	2.1372	51.4023	4.6336	2.6208
	5.6472	101.5411	6.3336	6.9108
	3.7596	214.5482	11.7313	7.3164
50	2.9172	15.8809	3.8376	2.5116
	4.1184	30.2330	5.9124	4.2276
	6.4584	63.1804	9.4849	8.1121
	10.2337	119.9180	15.8653	15.4285
	18.0961	226.6851	24.6014	25.0226
150	224.4854	40.1547	21.7933	49.7955
	224.8754	67.5796	26.2394	53.8359
	244.3444	117.2816	39.4995	77.7977
	281.9874	206.8573	64.1944	122.0240
	330.5661	388.4269	106.4239	205.2817
300	4.0516e+03	249.0244	147.4053	438.5968
	3.7590e+03	295.2163	184.3620	452.6993
	3.7677e+03	378.1932	204.9073	644.2217
	3.9029e+03	626.7340	309.4280	827.0237
	4.2586e+03	1.0882e+03	498.8444	1.3007e+03

Table 5.39: Computation time for 2D-ADR, 3rd order

Grid Size	Krylov-EPI	LEJA	AKP	KSS-EPI
25	6.8050	7.7300	4.4100	6.6850
	6.0525	7.0175	3.0400	5.9525
	5.6300	6.9913	2.4238	5.6563
	6.3750	6.9694	2.0775	6.1738
	4.6125	6.8825	1.8519	4.6941
50	10.4350	11.7000	6.3650	7.3400
	8.3525	9.0975	3.8900	6.4175
	7.1088	7.4125	2.7600	6.0000
	5.9500	6.7819	2.1844	5.6019
	5.4566	6.7953	1.8975	4.7466
150	28.3850	33.8100	26.8850	8.9500
	20.7850	21.9125	10.7325	6.5125
	15.5600	15.8500	6.2850	5.7825
	11.7663	11.2044	4.1919	5.1494
	8.8566	9.0122	2.9497	4.7509
300	55.4450	139.4600	58.5200	14.0850
	40.1475	67.6875	32.2525	8.6550
	29.4163	29.2638	13.4025	6.1488
	21.4300	20.7869	8.2669	5.1775
	15.7866	15.0753	5.4694	4.7609

Table 5.40: Number of iterations for 2D-ADR, 3rd order

5.2.6 System of Coupled PDE

For our final test problem, we consider the 2-D Brusselator problem [8, 22]

$$u_t = 1 + uv^2 - 4u + \alpha\nabla^2 u, \quad x,y \in [0,1], \quad t \in [0,0.1], \tag{5.33}$$

$$v_t = 3u - u^2 v + \alpha\nabla^2 u, \tag{5.34}$$

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

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

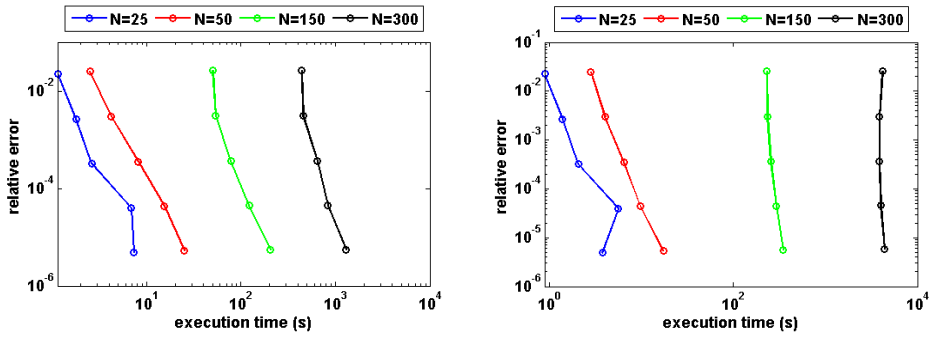
The N_c value for Brusselator equation is 30. The formula for 3-rd order frequency-dependent nodes, as defined in (3.103), is given by

$$nf = \begin{bmatrix} a - \sqrt{q_{11}q_{12}} & a + \sqrt{q_{11}q_{12}} \\ a - \sqrt{q_{21}q_{22}} & a + \sqrt{q_{21}q_{22}} \end{bmatrix}, \tag{5.35}$$

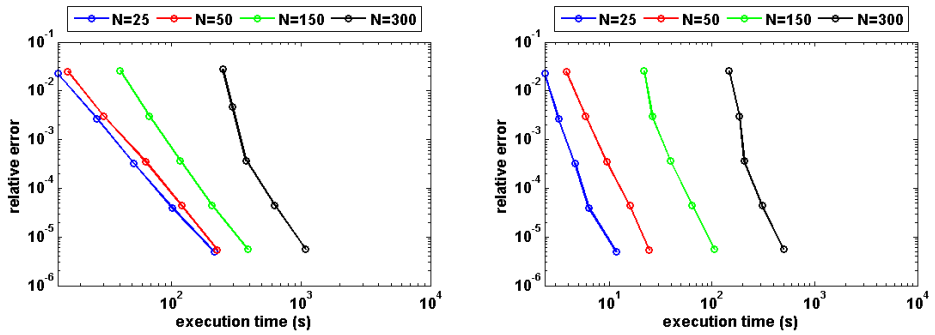
where

$$a = -0.2(N+1)^2 \left(4 - 2\cos\left(\frac{\omega_1}{N+1}\right) 2\cos\left(\frac{\omega_1}{N+1}\right)\right), \tag{5.36}$$

$$q_{11} = \frac{\|\tilde{p}\|_2^2 + \overline{\phi\psi}}{2\pi\sqrt{\|\tilde{p}\|_2^2 + \|\psi\|_2^2}}, \tag{5.37}$$



(a) KSS-EPI (b) Krylov-EPI



(c) Leja (d) AKP

Figure 5.24: 2D-ADR, 3rd order

$$q_{21} = \frac{\|\tilde{q}\|_2^2 + \overline{\phi\,\psi}}{2\pi\sqrt{\|\tilde{q}\|_2^2 + \|\phi\|_2^2}}, \tag{5.38}$$

$$q_{12} = \frac{\sqrt{\|\tilde{p}\|_2^2 + \|\psi\|_2^2}}{2\pi}, \tag{5.39}$$

$$q_{22} = \frac{\sqrt{\|\phi\|_2^2 + \|\tilde{q}\|_2^2}}{2\pi}, \tag{5.40}$$

and $p = v^2 - 4$, $\phi = 2uv$, $\psi = 3 - 2u$, and $q = -u^2$. The functions p, ϕ, ψ, and q are obtained from the Jacobian of (5.33). The 4-th and 5-th order frequency-dependent nodes, as defined in (3.102) are given by

$$nf = \begin{bmatrix} a - \sqrt{q_{11}q_{12} + b_{11}b_{12}} & a & a + \sqrt{q_{11}q_{12} + b_{11}b_{12}} \\ a - \sqrt{q_{21}q_{22} + b_{21}b_{22}} & a & a + \sqrt{q_{21}q_{22} + b_{21}b_{22}} \end{bmatrix}, \tag{5.41}$$

where

$$b_{11} = \frac{0.4(\|\vec{\omega}\cdot\nabla p\|_2^2 + \|\vec{\omega}\cdot\nabla\psi\|_2^2)}{\sqrt{\|\tilde{p}\|_2^2 + \|\psi\|_2^2}\sqrt{\|\vec{\omega}\cdot\nabla p\|_2^2 + \|\vec{\omega}\cdot\nabla\psi\|_2^2}}, \tag{5.42}$$

$$b_{21} = \frac{0.4(\|\vec{\omega}\cdot\nabla\phi\|_2^2 + \|\vec{\omega}\cdot\nabla q\|_2^2)}{\sqrt{\|\phi\|_2^2 + \|\tilde{q}\|_2^2}\sqrt{\|\vec{\omega}\cdot\nabla\phi\|_2^2 + \|\vec{\omega}\cdot\nabla q\|_2^2}}, \tag{5.43}$$

$$b_{12} = \frac{0.4\sqrt{\|\vec{\omega}\cdot\nabla p\|_2^2 + \|\vec{\omega}\cdot\nabla\psi\|_2^2}}{\sqrt{\|\tilde{p}\|_2^2 + \|\psi\|_2^2}}, \tag{5.44}$$

$$b_{22} = \frac{0.4\sqrt{\|\vec{\omega}\cdot\nabla\phi\|_2^2 + \|\vec{\omega}\cdot\nabla q\|_2^2}}{\sqrt{\|\phi\|_2^2 + \|\tilde{q}\|_2^2}}. \tag{5.45}$$

As we can see from Tables 5.41, 5.44, and 5.47, both Krylov-EPI and KSS-EPI yield similar error for the the smaller grid sizes $N = 25$ and $N = 50$ for 3rd and 5th orders, and $N = 25$ for 4th order, while Krylov-EPI is slightly more accurate KSS-EPI for the other grid sizes. At the same time, we again observe a decrease in the number of iterations used by KSS-EPI compared to Krylov-EPI as N increases, for both 3rd, 4th, and 5th orders, as shown in Tables 5.43, 5.46, and 5.49. However, since the number of iterations used for KSS-EPI is not much smaller than the number of iterations used for Krylov-EPI, the cost of Fourier transforms offset the advantage in the number of iterations, thus causing KSS-EPI to be only slightly more efficient than Krylov-EPI. This can also be observed in Tables 5.42, 5.45 and 5.48, and Figures 5.25, 5.26, and 5.27. Also, just like in the case of 2D ADR, denoising applied to KSS-EPI did not decrease the number of iterations significantly, adding extra computational time.

Grid Size	Krylov-EPI	LEJA	AKP	KSS-EPI
25	6.6193e-05	6.6193e-05	6.6193e-05	6.6193e-05
	7.3512e-06	7.3511e-06	7.3511e-06	7.3512e-06
	8.2593e-07	8.2586e-07	8.2587e-07	8.2593e-07
	9.6575e-08	9.6548e-08	9.6548e-08	9.6575e-08
	1.1345e-08	1.1633e-08	1.1632e-08	1.1645e-08
50	6.3453e-05	6.3453e-05	6.3453e-05	6.3450e-05
	7.0286e-06	7.0279e-06	7.0279e-06	7.0277e-06
	7.8699e-07	7.8660e-07	7.8660e-07	7.8643e-07
	9.1994e-08	9.1872e-08	9.1873e-08	9.1830e-08
	1.0188e-08	1.1065e-08	1.1066e-08	1.1069e-08
150	6.2598e-05	2.7120e-05	6.2594e-05	0.0015
	6.9258e-06	6.9225e-06	6.9226e-06	3.0811e-05
	7.7445e-07	7.2778e-07	7.2181e-07	7.7359e-07
	9.1561e-08	9.0128e-08	9.0146e-08	9.1252e-08
	1.1995e-08	1.0868e-08	1.0875e-08	1.2603e-08
300	6.2524e-05	4.5662e-05	6.2511e-05	0.1028
	6.2011e-06	9.8543e-06	6.9124e-06	0.0307
	7.7782e-07	1.5304e-06	7.7148e-07	0.0034
	9.6536e-08	8.9860e-08	8.9959e-08	4.6994e-05
	1.7033e-08	1.0839e-08	1.0845e-08	1.9347e-08

Table 5.41: Error for Brusselator equation, 3rd order

Grid Size	Krylov-EPI	LEJA	AKP	KSS-EPI
25	0.4992	1.9812	0.6708	0.5460
	0.8268	3.1200	0.9984	0.7488
	1.1856	6.1308	1.7316	1.2636
	2.0592	11.6377	2.9328	2.0280
	3.6816	22.3549	5.0700	3.8844
50	3.0732	1.9188	1.1076	2.8548
	3.9936	3.6348	1.5444	3.8532
	5.4288	6.8640	2.5116	6.0216
	7.7376	13.0573	4.0560	9.9373
	13.4161	26.3174	6.9264	19.0009
150	143.9733	10.9513	9.8593	70.9493
	135.9549	13.2133	11.2321	65.9260
	144.7221	23.6498	15.9901	75.1613
	170.8679	42.5571	23.0881	108.6547
	226.0454	73.7729	35.9894	187.1232
300	2.2553e+03	118.3580	107.4535	1.1094e+03
	1.8700e+03	129.7304	84.6305	869.3468
	1.7421e+03	139.7541	103.2571	669.3067
	1.8892e+03	153.5050	139.3089	662.8482
	2.1976e+03	251.0212	200.1961	923.4479

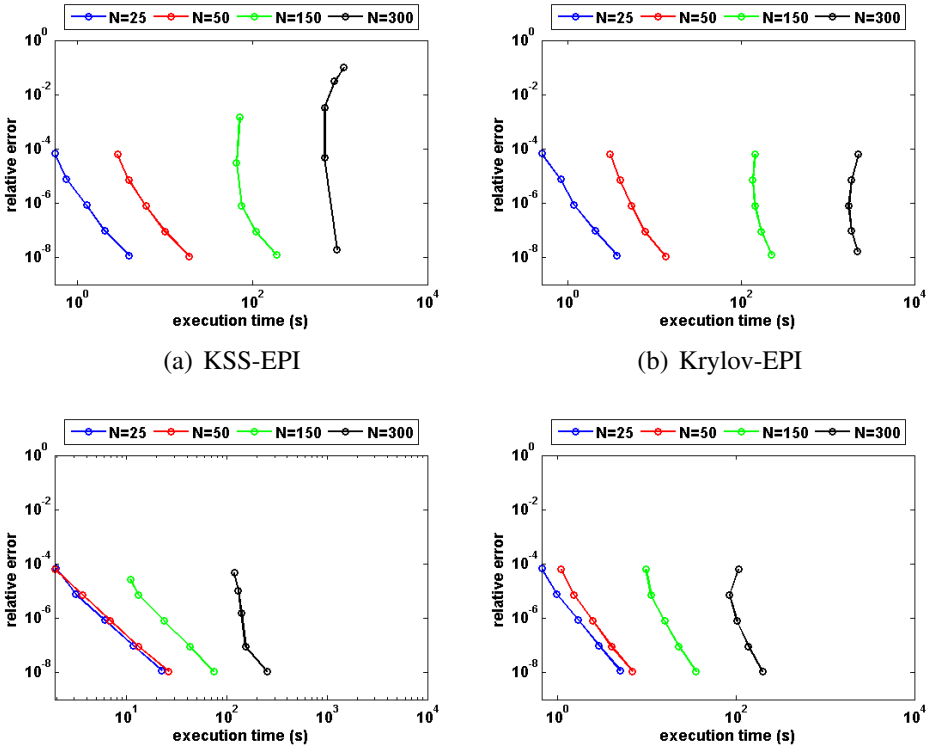
Table 5.42: Computation time for Brusselator equation, 3rd order

Grid Size	Krylov-EPI	LEJA	AKP	KSS-EPI
25	11.6000	16.1000	11.0000	11.6000
	8.8250	12.0750	7.5750	8.8250
	7.2500	10.7750	5.7375	7.2500
	6.0875	10.7250	4.6000	6.0875
	5.5438	10.6375	3.7125	5.5438
50	18.0500	26.6000	18.5500	15.5500
	12.4750	19.1500	11.6000	11.0250
	9.1250	13.7875	7.4875	8.2375
	7.0063	10.9688	5.2313	6.6688
	6.0656	10.1031	4.0625	5.8719
150	47.6000	120.7500	56.3000	34.6500
	31.1250	46.7750	33.7500	17.6000
	20.9500	33.6750	20.1125	10.7750
	14.4250	23.0500	11.9875	7.4625
	10.4938	16.9563	7.3875	6.1563
300	90.8000	423.1000	199.7500	64.9500
	58.9250	216.6000	76.3000	38.0250
	38.8125	101.5500	42.4625	18.8750
	26.5750	39.5563	24.8188	9.9813
	18.2188	28.3938	14.9938	6.9625

Table 5.43: Number of iterations for Brusselator equation, 3rd order

Grid Size	Krylov-EPI	LEJA	AKP	KSS-EPI
25	1.9390e-04	0.0013	0.0013	1.9390e-04
	1.5164e-05	3.2562e-04	3.2564e-04	1.5164e-05
	1.2886e-06	7.7112e-05	7.7108e-05	1.2886e-06
	1.2456e-07	1.8424e-05	1.8416e-05	1.2456e-07
	1.3354e-08	4.4892e-06	4.4785e-06	1.3354e-08
50	2.0925e-04	0.0013	0.0013	2.0930e-04
	1.6196e-05	3.3609e-04	3.3615e-04	1.6216e-05
	1.3675e-06	7.9614e-05	7.9584e-05	1.3777e-06
	1.3197e-07	1.9003e-05	1.8990e-05	1.3473e-07
	1.4121e-08	4.6213e-06	4.6150e-06	1.4606e-08
150	2.1422e-04	0.0010	0.0013	2.1885e-04
	1.6503e-05	3.3949e-04	3.3955e-04	1.6559e-05
	1.3887e-06	8.0520e-05	8.0372e-05	1.4702e-06
	1.3521e-07	1.9174e-05	1.9170e-05	2.6074e-07
	1.5218e-08	4.3757e-06	4.6581e-06	8.4155e-08
300	2.1471e-04	8.2644e-05	0.0034	0.1673
	1.6535e-05	2.4603e-04	3.6265e-04	0.0104
	1.3926e-06	6.6651e-05	8.0939e-05	6.9401e-05
	1.3719e-07	1.9246e-05	1.9188e-05	1.3757e-07
	1.7834e-08	4.7818e-06	4.6623e-06	1.1388e-07

Table 5.44: Error for Brusselator, 4th order



(a) KSS-EPI (b) Krylov-EPI

(c) Leja (d) AKP

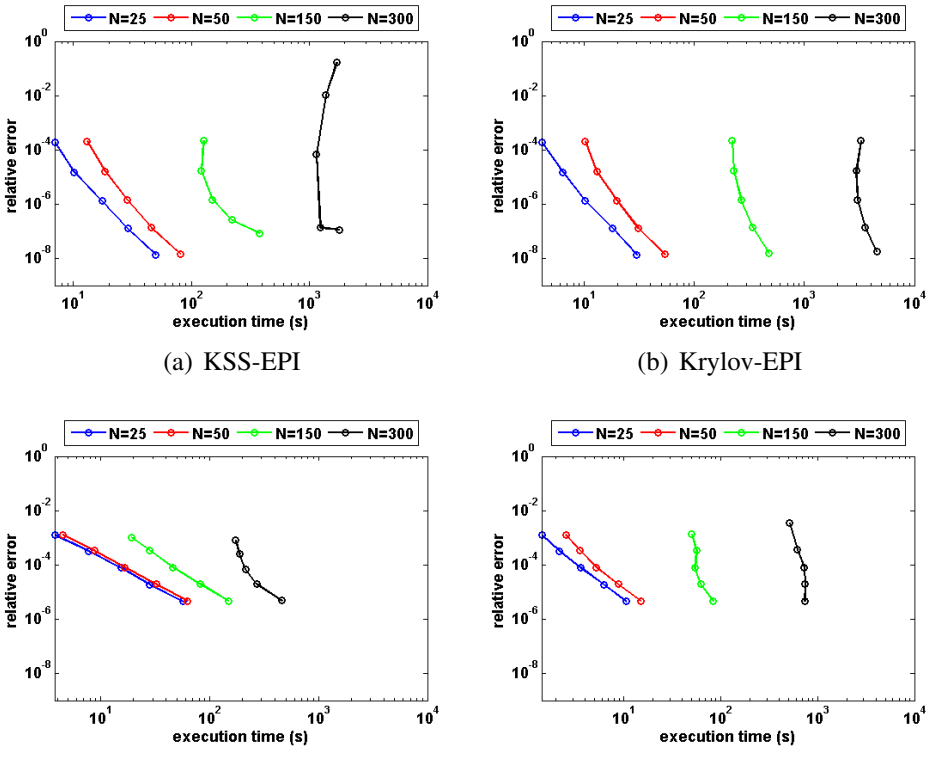
Figure 5.25: Brusselator equation, 3rd order

Grid Size	Krylov-EPI	LEJA	AKP	KSS-EPI
25	4.1028	3.7752	1.4352	6.9264
	6.3804	7.7532	2.1996	10.1557
	10.2181	15.3193	3.6348	17.4409
	18.0805	28.1894	6.2868	28.8134
	30.0926	56.6440	10.6393	49.7643
50	10.2181	4.5084	2.5428	12.9949
	12.9949	8.7985	3.5880	18.6109
	19.9369	16.6453	5.2884	28.6574
	30.8414	32.6510	8.8921	45.8799
	54.1011	62.6032	15.0853	80.6057
150	219.7742	19.1725	50.8251	126.1268
	228.5259	28.0490	56.8468	121.2752
	265.6073	45.9735	54.3351	150.3850
	3402538	82.1501	62.4160	219.4310
	477.7531	148.1541	83.8505	378.5832
300	3.2683e+03	173.1455	514.8813	1.6977e+03
	2.9652e+03	187.9032	624.0820	1.3777e+03
	3.0498e+03	213.2846	729.1643	1.1449e+03
	3.5814e+03	269.6789	746.2152	1.2348e+03
	4.5542e+03	463.1514	736.1375	1.8053e+03

Table 5.45: Computation time for Brusselator, 4th order

Grid Size	Krylov-EPI	LEJA	AKP	KSS-EPI
25	14.3333	48.0000	17.0000	13.6667
	12.6667	32.0000	12.0833	10.5000
	9.6667	21.1667	8.1467	8.0833
	8.1250	14.7500	6.1875	6.7083
	6.8542	9.9583	4.3750	5.7083
50	20.3333	157.3333	34.1667	16.3333
	17.3333	53.5000	28.6667	11.6667
	13.5000	34.4167	15.7083	8.4167
	11.9167	22.5417	10.0000	6.9167
	9.4167	14.3333	5.8542	5.8333
150	21.6667	1.3013e+03	302.8333	17.6667
	21.1667	519.5000	209.3333	12.0000
	30.5833	190.5833	114.2083	8.5833
	28.8333	70.2917	48.1250	6.9583
	22.9583	25.1250	20.9063	5.8542
300	23.0000	5.2227e+03	873.6667	17.6667
	50.1667	2.1858e+03	506.4167	11.8333
	61.2500	808.0000	295.0000	8.5833
	54.5833	281.8750	155.9375	6.9167
	42.8542	101.0208	73.2396	5.8542

Table 5.46: Number of iterations for Brusselator, 4th order



(a) KSS-EPI (b) Krylov-EPI

(c) Leja (d) AKP

Figure 5.26: Brusselator equation, 4th order

Grid Size	Krylov-EPI	LEJA	AKP	KSS-EPI
25	1.8136e-06	1.0652e-07	1.0838e-07	1.8136e-06
	2.4172e-07	4.7414e-09	4.6168e-09	2.4172e-07
	3.7753e-08	2.9361e-09	1.6170e-10	3.7753e-08
	6.7439e-09	2.1235e-09	6.5355e-12	6.7439e-09
	2.2535e-09	2.8176e-09	2.5613e-12	2.2535e-09
50	1.4488e-05	1.0909e-07	1.1164e-07	9.0526e-06
	2.3530e-06	1.1528e-08	4.8339e-09	6.6851e-07
	2.3556e-07	3.8756e-09	1.7317e-10	1.1561e-07
	4.3887e-08	2.1031e-09	6.7663e-12	1.7210e-08
	2.9021e-09	2.3449e-09	2.5513e-12	2.7587e-09
150	9.3983e-05	1.1719e-05	1.1298e-07	5.1901e-05
	1.5337e-05	1.6509e-08	4.9085e-9	5.9497e-06
	3.1141e-06	1.6802e-08	3.0252e-10	1.1869e-06
	7.9372e-07	1.6824e-10	4.3692e-11	1.1416e-07
	2.5822e-07	6.4371e-09	6.0797e-12	1.6684e-08
300	1.6072e-04	2.3853e-05	1.1314e-07	0.0229
	3.1909e-05	4.2411e-06	4.8932e-09	0.0019
	9.9838e-06	4.3337e-07	2.8651e-10	1.7218e-05
	3.1886e-06	5.6779e-09	4.2807e-11	5.7163e-07
	7.3789e-07	9.9066e-09	7.9912e-12	9.4780e-08

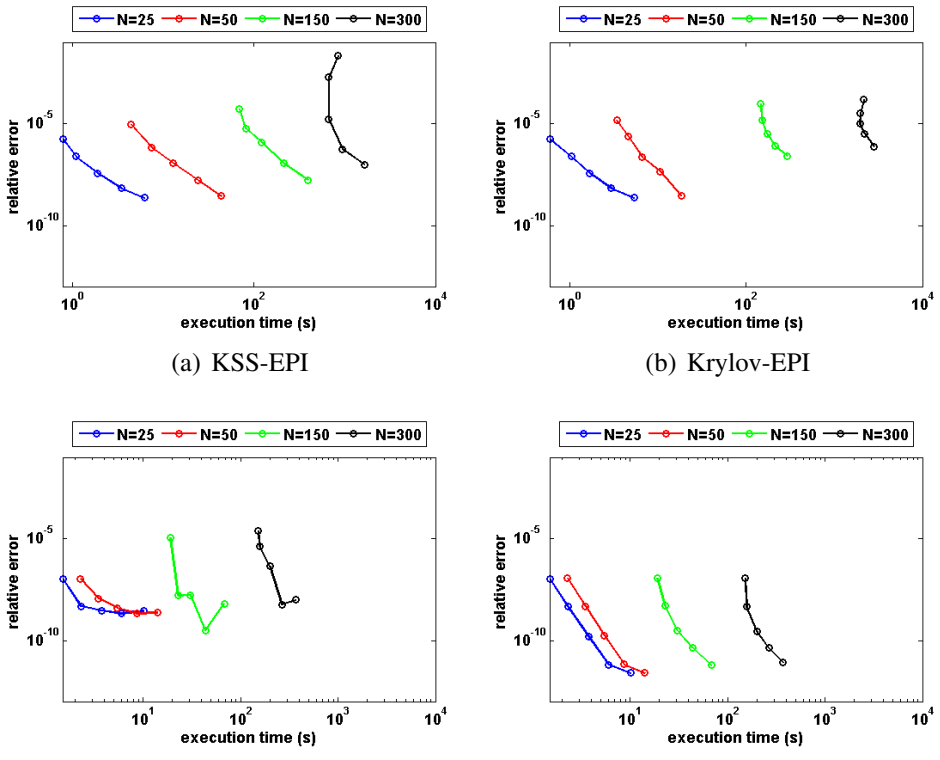
Table 5.47: Error for Brusselator equation, 5th order

Grid Size	Krylov-EPI	LEJA	AKP	KSS-EPI
25	0.5928	4.5708	1.4976	0.7800
	1.0452	8.6581	2.3088	1.0920
	1.6848	17.7373	3.7596	1.9032
	2.9796	64.4606	6.0528	3.4632
	5.3664	67.0648	10.1557	6.2088
50	3.4164	5.3664	2.2776	4.4460
	4.6488	9.9841	3.5100	7.4256
	6.5988	19.5001	5.4756	12.7609
	10.6705	37.8458	8.6581	23.9774
	18.4237	72.7121	14.1337	43.5243
150	145.9857	20.4673	19.0477	69.1240
	152.4754	30.2330	22.7605	81.6509
	173.8943	50.9811	30.7478	122.3360
	213.3782	93.5538	43.7739	211.7870
	294.6547	174.0971	68.4220	395.1661
300	2.1684e+03	180.3372	152.4442	848.3958
	1.9709e+03	205.4221	157.1710	664.5487
	1.9515e+03	232.2855	199.7125	669.0727
	2.1222e+03	293.5315	265.2173	949.8901
	2.8139e+03	514.1325	372.2652	1.6401e+03

Table 5.48: Computation time for Brusselator equation, 5th order

Grid Size	Krylov-EPI	LEJA	AKP	KSS-EPI
25	9.1333	15.2333	7.3667	9.1333
	7.4000	12.2667	5.2333	7.4000
	6.3417	11.4417	3.8792	6.3417
	5.6750	10.2458	2.9750	5.6750
	5.0792	9.5479	2.3219	5.0688
50	13.7333	22.9333	11.3833	11.8000
	10.1000	15.0500	7.1917	8.8833
	7.5750	11.8167	4.7417	7.0000
	6.2125	9.8792	3.4229	6.0542
	5.4208	9.4063	2.5760	5.3792
150	34.4667	68.1333	27.5000	18.7000
	23.5000	30.9000	19.1667	11.2667
	16.3500	22.1267	11.6667	7.9500
	11.6458	16.4542	7.7083	6.3458
	8.8438	11.8917	4.9271	5.4938
300	65.5000	242.4333	93.1667	38.6667
	43.7833	122.1500	42.7750	19.9333
	29.4250	54.1667	23.9708	10.5586
	20.5625	25.2833	13.3771	7.0958
	15.0750	18.4604	7.6135	5.6813

Table 5.49: Number of iterations for Brusselator equation, 5th order



(a) KSS-EPI (b) Krylov-EPI

(c) Leja (d) AKP

Figure 5.27: Brusselator equation, 5th order

5.2.7 Discussion of Efficiency

From these five 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 (4.9). 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.

- In this dissertation, 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.

Chapter 6

CONCLUSIONS

We have demonstrated that when solving stiff systems of nonlinear ODE derived from the spatial discretization of a nonlinear PDE, the eigenvalue problem for the block tridiagonal matrix produced by the block Lanczos algorithm decouples in the limit as the frequency increases. As a result, we were able to accurately compute block Guassian quadrature nodes for all frequencies much more efficiently than the computational cost of evaluating them directly. Also, we have seen that an increase in the number of grid points in the spatial discretization of the PDE does not necessarily require a corresponding increase in the number of Krylov projection steps needed to maintain high-order accuracy in time. By employing a componentwise approach to the computation of $\varphi(\tau A)\mathbf{b}$ 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 φ, the Krylov subspace dimension can be bounded independently of the grid size and instead determined by the desired temporal order of accuracy.

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 chapter, combination with other methods for matrix function-vector products, including Leja interpolation and adaptive Krylov projection, will be investigated. Finally, it will be essential to generalize the approach demonstrated in this dissertation for estimating frequency-dependent nodes to other classes of differential operators. Given that these nodes tend to be smooth functions of the wave number, this generalization could be accomplished via interpolation.

BIBLIOGRAPHY

[1] Atkinson, K.: *An Introduction to Numerical Analysis, 2nd Ed.* Wiley (1989)

[2] Bergamaschi, L., Caliari, M., Martínez, A., Vianello, M.: Comparing Leja and Krylov approximations of large scale matrix exponentials. *Lecture Notes in Comput. Sci.*, *6th International Conference, Reading, UK, May 28–31, 2006, Proceedings, Part IV*, Alexandrov, V. N., van Albada, G. D., Sloot, P. M. A., Dongarra, J. (eds.), Springer (2006) 685-692.

[3] Carliari, M., Vianello, M., Bergamaschi, L.: Interpolating discrete advection-diffusion propagators at Leja sequences. J. Comput. Appl. Math. 172(2004) 79-99.

[4] Golub, G. H., Meurant, G.: Matrices, Moments and Quadrature. *Proceedings of the 15th Dundee Conference*, June-July 1993, Griffiths, D. F., Watson, G. A. (eds.), Longman Scientific & Technical (1994)

[5] Golub, G. H., Underwood, R.: The block Lanczos method for computing eigenvalues. *Mathematical Software III*, J. Rice Ed., (1977) 361-377.

[6] Golub, G. H., van Loan, C. F.: *Matrix Computations*, 3rd Ed., Johns Hopkins University Press (1996).

[7] Guidotti, P., Lambers, J. V., Kim, Y: Image Restoration with a New Class of Forward-Backward Diffusion Equations of Perona-Malik Type. *SIAM Journal on Imaging Sciences* **6**(3) (2013) 1416-1444.

[8] Hairer, E., Wanner, G.: *Solving Ordinary Differential Equations II*, 2nd revised edition, Springer (2002).

[9] Hochbruck, M., Lubich, C.: A Gautschi-type method for oscillatory second-order differential equations, *Numer. Math.* **83** (1999) 403-426.

[10] Hochbruck, M., Lubich, C.: On Krylov Subspace Approximations to the Matrix Exponential Operator. *SIAM J. Numer. Anal.* **34** (1996) 1911-1925.

[11] Hochbruck, M., Lubich, C., Selhofer, H.: Exponential Integrators for Large Systems of Differential Equations. *SIAM J. Sci. Comput.* **19** (1998) 1552-1574.

[12] Lambers, J. V.: Enhancement of Krylov Subspace Spectral Methods by Block Lanczos Iteration. *Electron. T. Numer. Ana.* **31** (2008) 86-109.

[13] Lambers, J. V.: Implicitly Defined High-Order Operator Splittings for Parabolic and Hyperbolic Variable-Coefficient PDE Using Modified Moments. *International Journal of Computational Science* **2** (2008) 376-401.

[14] Loffeld, J., Tokman, M.: Comparative Performance of Exponential, Implicit and Explicit Integrators for Stiff Systems of ODEs. Submitted.

[15] Palchak, E. M., Cibotarica, A., Lambers, J. V.: Solution of Time-Dependent PDE Through Rapid Estimation of Block Gaussian Quadrature Nodes. Submitted.

[16] Tokman, M.: Efficient integration of large stiff systems of ODEs with exponential propagation iterative (EPI) methods. *J. Comp. Phys.* **213** (2006) 748-776.

[17] Tokman, M.: A new class of exponential propagation iterative methods of Runge-Kutta type (EPIRK). *J. Comp. Phys.* **230** (2011) 8762-8778.

[18] Lambers, J. V.: An Explicit, Stable, High-Order Spectral Method for the Wave Equation Based on Block Gaussian Quadrature. *IAENG Journal of Applied Mathematics* **38** (2008) 333-348.

[19] Lambers, J. V.: Explicit High-Order Time-Stepping Based on Component-wise Application of Asymptotic Block Lanczos Iteration. *Num. Lin. Alg. Appl.* **19**(6) (2012) 970-991.

[20] Lambers, J. V.: A Spectral Time-Domain Method for Computational Electrodynamics. *Advances in Applied Mathematics and Mechanics* **1**(6) (2009) 781-798.

[21] Lambers, J. V. Krylov Subspace Spectral Methods for the Time-Dependent Schrödinger Equation with Non-Smooth Potentials. *Numerical Algorithms* **51** (2009) 239-280.

[22] Lefever, R., Nicolis, G.: Chemical instabilities and sustained oscillations. *J. Theor. Biol.* **3** (1971) 267-284.

[23] Rainwater, G., Tokman, M.: A new class of split exponential propagation iterative methods of RungeâĂŞKutta type (sEPIRK) for semilinear systems of ODEs.*J. Comp. Phys.* **269** (2014) 40-60.

[24] Richardson, M., Lambers, J. V.: Krylov Subspace Spectral Methods for Polar and Cylindrical Geometries. In preparation.

[25] Moret, I., Novati, P.: RD-rational approximation of the matrix exponential operator. *BIT* **44** (2004) 595-615.

[26] Saad, Y.: *Numerical Methods for Large Eigenvalue Problems*. Halsted Press, New York (1992).

[27] Guidotti, P., Lambers, J. V., Sølna, K.: Analysis of 1-D Wave Propagation in Inhomogeneous Media. *Numer. Funct. Anal. Opt.* **27** (2006) 25-55.