Novel Iterative Methods for Solving Matrix Equations Arising from Partial Differential Equations

Ioana Teodora Vaduva

Department of Mathematics and Statistics,

University of Strathclyde,

Glasgow, U.K.

A thesis submitted for the degree of

Doctor of Philosophy

March 2022

Copyright Declaration

Chapter 3 of this thesis contains material which has been submitted for publication, reference [86]. The results contained therein are the result of the author's original research in collaboration with the article's co-authors, Dr Philip A. Knight and Dr Jennifer Pestana.

Signed:

Date:

Acknowledgements

First and foremost, I would like to express my deepest thanks to my supervisors, Jen Pestana and Phil Knight, for their guidance and support. Their patience to explain difficult topics, enthusiasm and encouragement, and continuous feedback, motivated me to push forward. I would also like to thank the University of Strathclyde and the Department of Mathematics and Statistics for funding my PhD studies.

I am also very grateful to have been able to meet and have stimulating conversations with many visitors to our Department, as well as during external events. Special mentions go to Kirk Soodhalter for his description of Krylov subspaces, to Catherine Powell who has been kind enough to share her code for rational projections, to Stefan Güttel for patiently explaining the rational Krylov framework, as well as to Stefano Massei for our discussion on the Zolotarev poles.

My time as a PhD student would not have been the same without the NA group, the SIAM-IMA Student Chapter, my office mates in LT1010 and my PhD buddies. They have all made research more pleasant. Special thanks and love go to my friends from home and to Adelina Manea whose passion for Mathematics inspired me to pursue it in the first place.

Finally, I thank Ovidiu for being there during the difficult times and for celebrating the good. Most importantly, a big thank you to my parents for their support and love. Their guidance and advice has been invaluable. Mulțumesc.

Abstract

In this thesis, we investigate the numerical solution of large-scale linear matrix equations arising from the discretisation of diffusion and convection–diffusion partial differential equations (PDEs). The matrix equations which arise can be Lyapunov, Sylvester or generalised Sylvester equations. Both Lyapunov and Sylvester equations can be solved using a rational Krylov approach, where orthonormal bases of rational Krylov subspaces are used to project the matrix equations. The projected matrix equation is then solved, and a low-rank solution is computed. As part of the generation of the orthonormal basis, the rational Krylov subspace requires a selection of poles, which, in this thesis, are chosen a priori. For Lyapunov equations arising from the discretisation of diffusion PDEs, we derive an explicit rational approximation for the solution for both 1- and 2-sided projections and provide a comparison of the two approaches. For both Lyapunov and Sylvester equations, we adapt the iterative rational Krylov algorithm (IRKA) from model order reduction to generate efficient pole choices for rational Krylov subspaces, which we then use in 2-sided projections. We perform thorough comparisons of pole choice approaches to solve the matrix equations from the practical point of view by considering convergence rates, computational costs and scalability. As a result, we can see that the IRKA poles can be competitive choices for a large set of realistic problems. For the generalised Sylvester equation we derive a stationary iterative method, determine a convergence criterion, and evaluate the performance via numerical results.

Contents

Acronyms and notation

Acronyms

SVD	Singular value decomposition
(P)CG	(Preconditioned) Conjugate Gradient method
GMRES	Generalised minimal residual method
ADI	Alternating-direction implicit method
IRKA	Iterative rational Krylov algorithm

Linear algebra

$\mathrm{diag}(a, b, c, \dots)$	Diagonal matrix with diagonal entries $a, b, c, \dots$
I	Identity matrix of appropriate dimensions
A^T (A^*)	The (conjugate) transpose of a matrix $A \in \mathbb{R}^{n\times n}$ $(A \in \mathbb{C}^{n\times n})$
$\sigma(A)$	The spectrum of matrix $A \in \mathbb{R}^{n\times n}$ (Definition 2.1.1)
$W(A)$	The field of values of matrix $A \in \mathbb{R}^{n\times n}$ (Definition 2.1.5)
$\sigma_\epsilon(A)$	The ϵ-pseudospectrum of matrix $A \in \mathbb{R}^{n\times n}$ (Definition 2.1.6)
$\lambda_{\min}$	The smallest eigenvalue of a Hermitian matrix
$\lambda_{\max}$	The largest eigenvalue of a Hermitian matrix
λ_k	The kth eigenvalue of a Hermitian matrix, with eigenvalues ordered $\lambda_1 \leq \cdots \leq \lambda_k \leq \cdots \leq \lambda_n$
$\rho(A)$	The spectral radius of matrix $A \in \mathbb{R}^{n\times n}$ (see (2.2))
$\lVert\cdot\rVert_2$	The 2-norm

$\lVert\cdot\rVert_F$	The Frobenius norm
$\lVert\cdot\rVert_A$	The norm induced by symmetric positive definite matrix $A \in \mathbb{R}^{n\times n}$
$\kappa(A)$	The 2-norm condition number of a matrix A
$A \otimes B$	The Kronecker product of matrices A and B (Definition 2.1.7)
$\mathrm{vec}(A)$	The vec operator which transforms a matrix into a vector
$\mathcal{P}_m$	The set of polynomials of degree at most m
$\mathcal{Q}_{m,n}$	The set of rational functions with numerator of degree at most m and denominator of degree at most n

Krylov subspaces

$\mathcal{K}_k(A,b)$	The Krylov subspace of degree k for matrix A and right-hand side b
$\mathcal{RK}_k(A,b,\mathcal{S})$	The rational Krylov subspace of degree k with poles $\mathcal{S}$ for matrix A and right-hand side b
$\mathcal{EK}_k(A,b)$	The extended Krylov subspace of degree k for matrix A and right-hand side b

Chapter 1

Introduction

Partial differential equations (PDEs) are widely used to model many real-world phenomena such as climate and biological systems, as well as in material sciences, and modern aircraft and spacecraft. The resulting models are often very complex and difficult to solve analytically, thus, they require efficient numerical solutions which can capture the characteristics of the physical system.

It is common to first discretise PDEs (using, for instance, finite differences, or finite elements), which can often lead to large, sparse linear systems. Historically, a lot of focus has been on developing methods for solving large scale systems of linear equations $\mathcal{A}x = b$, with $\mathcal{A} \in \mathbb{C}^{n\times n}$, $b \in \mathbb{C}^n$ in order to obtain computationally efficient and accurate numerical solutions. Such methods can be either direct or iterative.

For coefficient matrices of moderate dimensions, direct methods, such as LU or Cholesky factorisations, are often used. They can obtain accurate solutions in a finite number of steps and can take advantage of the sparsity pattern and/or structure of the coefficient matrix. For larger, sparse linear systems, iterative methods may be more efficient. Simple stationary iterations such as the Jacobi, Gauss–Seidel and SOR methods can be easily implemented, but they often are very slow or fail to converge. Multigrid methods are an appealing alternative that can solve some classes of linear systems by using mesh information from different discretisations of the PDE. However, they can also diverge or exhibit slow convergence.

A widespread approach for solving linear systems with large, sparse coefficient matrices is to use Krylov subspace methods. These methods are split into two main classes, based on whether the coefficient matrix is symmetric or not. For the symmetric case, the conjugate gradient method for positive definite matrices [42] and MINRES for indefinite matrices [62] are popular choices, while for non-symmetric coefficient matrices, we consider GMRES [73] in this thesis, with many competitive approaches developed, such as, for example, BiCGStab(ℓ) [79, 87], QMR [33] and IDR(s) [80]. Applying these Krylov subspace methods in their simplest forms often results in very slow convergence, which is remedied by using preconditioners. Preconditioning aims to replace the original linear system with one which is easier to solve. The development of good preconditioners is an active area of research, with suitable preconditioners often developed to tackle problem-specific issues.

The sizes of the coefficient matrices of linear systems have increased rapidly over time and, for some extremely large problems, even storing $\mathcal{A}$ and b can be very expensive. For some 2-dimensional PDEs, the problem can either be written as a linear system or as a matrix equation.

Linear matrix equations of the form $A_1XB_1 + \cdots + A_kXB_k = F$, with $A_i \in \mathbb{R}^{n\times n}$, $B_i \in \mathbb{R}^{m\times m}$, $F \in \mathbb{R}^{n\times m}$, can arise in many scientific areas, such as stability and control theory [1, 34], model order reduction of dynamical systems [8], PDE constrained optimisation [81], and data assimilation [32]. In this thesis we consider matrix equations arising from the discretisation of diffusion and convection-diffusion PDEs. We study two main types of linear matrix equations. One is given by Sylvester equations, $AX+XB=F$, which arise in, for instance, the discretisation of convection–diffusion PDEs [64], and block-diagonalisation of 2-by-2 block triangular matrices [88, Ch. 7.1.4]. The Sylvester equation can be generalised to a multi-term matrix equation $AX+XB+\sum_{i=1}^{m} N_i X M_i = F$, called generalised Sylvester equations [64, 71]. The other set of common matrix equations is given by Lyapunov equations, which take the form $AX+XA^T=F$, and can be considered a special case of Sylvester equations, with $B=A^T$. Lyapunov equations have an important role in, for example, control theory [1, 34], and the solution of algebraic Riccati equations [10, 12]. As for Sylvester equations, a multi-term version of the Lyapunov equation can be obtained, given by $AX+XA^T+\sum_{i=1}^{m} N_i X N_i^T = F$ and called a generalised Lyapunov equation. These can arise in the context of stochastic differential equations, see e.g. [5, 18, 75].

Numerous methods have been developed to work directly with the matrix equation formulation. As in the case of linear systems, there are multiple approaches to solving Lyapunov and Sylvester equations. The Bartels–Stewart method [3] is a direct method based on the Schur decomposition that is widely used for smaller problems. For large problems involving sparse matrices, the ADI method [91] is popular, as are methods based on projections into standard [45, 48, 50, 51, 65], extended [20, 53, 64, 65, 76] and rational Krylov subspaces [4, 21–23]. We remark that the ADI method and rational Krylov projections are equivalent when the poles coincide with the Ritz values of the rational Krylov projection [21].

In order to solve large matrix equations, it is common in projection methods to exploit the small numerical rank of the solution, which can be ensured by considering low rank right-hand sides. The idea is that these iterative methods are feasible when we do not require too many iterations. This is precisely the case when the solution has low rank or is well approximated by a low rank matrix. This phenomenon occurs because of the rapid decay to zero of the singular values of the solution matrix. Several theoretical results describe the singular value decay of the solution and how it can be approximated, e.g. [2, 5, 37, 69]. Large scale Sylvester and Lyapunov equations with right-hand sides which are not low rank have only been briefly analysed and the literature lacks efficient numerical methods in general, with banded right-hand sides being one of the exceptions [66].

Generalised Sylvester and Lyapunov equations are more difficult to analyse and solve, and fewer solution methods are available, than for standard matrix equations. Note that the approaches mentioned for the standard Sylvester and Lyapunov matrix equations cannot be straightforwardly extended. For generalised Lyapunov equations, ADI-preconditioned solvers [5, 18], a greedy low rank approach [54], and a stationary iterative method [75] have been proposed. The solution of generalised Sylvester equations has also been studied, with [49] presenting an approach for matrices with a low-rank commutativity property, and with [71] discussing a projection-based approach in the context of stochastic Galerkin matrix equations. For multi-term matrix equations arising from the discretisation of convection–diffusion PDEs, an extended Krylov subspace method is mentioned in [64].

1.1 Thesis outline

In this thesis, we focus on solving Lyapunov, Sylvester and generalised Sylvester equations arising from the discretisation of diffusion and convection-diffusion PDEs. This thesis is structured as follows. In Chapter 2 we present some preliminary results from linear algebra, as well as details about the discretisation of our model problems. We then discuss common practices for solving both linear systems and matrix equations. We conclude Chapter 2 by numerically solving equivalent linear systems and matrix equations, highlighting the advantages of solving matrix equation approaches over linear systems for our problems of interest.

We dedicate Chapter 3 to the study of the Lyapunov equation arising from the discretisation of the diffusion PDE. We begin by presenting an explicit rational approximation for the solution of the Lyapunov equation, and present 1- and 2-sided projections onto rational Krylov subspaces, as well as a numerical comparison between the two approaches. The rest of the Chapter focuses on the 2-sided projection approach and first presents a numerical study of the attainable accuracy for the projection method and a comparison with theoretical bounds. In the final part of Chapter 3 we explore pole choices, including the commonly used Zolotarev poles and the iterative rational Krylov algorithm (IRKA) poles [39] adapted from model order reduction of dynamical systems. We numerically compare our rational Krylov method with these a priori pole methods with the adaptive pole algorithm from [23] for a variety of test problems with both rank-1 and higher rank choices of right-hand sides and discuss the scalability of the solver.

In Chapter 4 we focus on the discretised convection-diffusion PDE. Depending on the convection wind, the PDE can lead to three types of matrix equations, a Lyapunov equation, a Sylvester equation, or a generalised Sylvester equation. We consider each of these matrix equations in turn. For the Lyapunov equations, we discuss the projection method and how it is influenced by the nonnormality of the coefficient matrices, and derive eigenvector and field of values bounds on the norm of the error. We then discuss a priori pole choices and present numerical results with those. For the Sylvester equation, we present the rational projection method, and compare pole choices in our numerical examples. The numerical experiments we show for this equation span a range of convection winds which lead to Sylvester equations. Finally, we discuss a stationary iterative method for the generalised Sylvester equation, and describe the convergence of the fixed point iteration. We finish by presenting numerical results for solving the generalised Sylvester equation with this stationary iterative method.

In Chapter 5 we present concluding remarks and discuss routes for future work.

All numerical tests presented in this thesis were performed on a MacBook Air (M1, 2020) macOS Monterey Version 12.0.1, 8-Core CPU, 7-Core GPU, 16 GB RAM (LPDDR4), MATLAB Version 9.10.0.1710957 (R2021a) 64-bit (maci64).

1.2 Aims and contributions

The aim of this thesis is to obtain novel methods for efficiently solving Lyapunov and Sylvester equations arising from the discretisation of diffusion and convection–diffusion PDEs. Justifying and validating the matrix equation approach, as well as understanding the scalability of the method as the problem size increases, will be two of the objectives of this thesis. Moreover, we are interested in understanding the most appropriate projection approaches to reduce the dimensions of the matrix equations described. To accomplish this, we consider projections onto rational Krylov subspaces and compare our results with standard and extended Krylov subspaces. Furthermore, we note that the rational Krylov subspace requires us to choose a set of poles, usually a priori, which we then use to construct the space. We wish to determine the most appropriate poles to use while generating the rational Krylov subspace, both accurately and efficiently. To do this, we provide both mathematical analysis and numerical studies. Finally, we consider a generalised Sylvester equation, for which we aim to understand how a stationary iterative approach can be used to solve the matrix equation. To this end, we make the following contributions.

- For Lyapunov equations arising from the discretisation of diffusion PDEs, in Section 3.2.2 and Section 3.2.3, we derive explicit rational approximations for the 1- and 2-sided projections, respectively. The 2-sided rational function has also been derived in [21] by using the skeleton approximation. Obtaining the rational function explicitly allows for better understanding of the rational approximation and, as a result, the best poles for the rational Krylov basis.

- To the best of our knowledge, a numerical comparison of the 1- and 2-sided rational projections has not been performed in the literature. We present in Section 3.2.4 such a comparison by considering the convergence curves and number of iterations of the two methods in Figure 3.1, as well as CPU times.

- In Section 3.3 we explore a very strict stopping tolerance in order to understand the attainable accuracy of our projection method for Lyapunov equations arising from the discretisation of diffusion PDEs. We compare standard, extended and rational Krylov bases, and determine when the two phases of the convergence (one corresponding to a reduction in the relative residual and one corresponding to noise) occur.

- In Algorithm 5 we adapt the IRKA approach developed in the field of model order reduction [39] to generate poles for the rational Krylov subspace for Lyapunov equations arising from the discretisation of diffusion PDEs. The IRKA poles have been used as ADI poles in [9, 31], and are mentioned in the context of rational Krylov projections in [93]. Despite being considered inefficient in these papers, we find in Section 3.6 that they can outperform other a priori chosen poles, as well as the state-of-the-art adaptive approach from [23] for a variety of test problems.

- For Lyapunov equations with nonsymmetric coefficient matrices, in Theorem 4.2.1 we present an eigenvector bound for the norm of the error of the rational approximation, while in Theorem 4.2.2 we present a field of values bound for the error. A field of values bound is also obtained in [23], but we note that this bound is looser than ours as the authors derive it with the aim of comparing the rational Krylov and ADI solutions.

- In Section 4.2.5 and Section 4.3.3 we present numerical results for rational projections with Zolotarev and IRKA poles for a wide variety of Lyapunov and Sylvester equations arising from the discretisation of convection-diffusion PDEs. We find that, as the problems become more convective, solving the matrix equations with Zolotarev poles is often outperformed by solving them with IRKA poles, highlighting their effectiveness for these types of problems.

- In Section 4.4 we adapt the stationary iterative method for generalised Lyapunov equations from [75] to work with a generalised Sylvester equation arising from the discretisation of convection–diffusion PDEs. We provide a convergence criterion for this in Lemma 4.4.2 and present numerical results in Table 4.5.

Chapter 2

Background material

Throughout this chapter we present background information related to the problems and methods which we will consider in this thesis. This chapter is split into four main parts. We begin with some results from linear algebra, which will include topics such as eigenvalues and eigenvectors of matrices, nonnormality, and Kronecker products. We then present the discretisation of our model problems and relate the matrix equation formulations to equivalent linear system forms. The last two sections of this chapter deal with linear systems and matrix equations, respectively, and include details about commonly used solvers. We finish the chapter with a numerical comparison between linear system and matrix equation solvers for both our discretised model problems, highlighting the advantages and disadvantages of both.

2.1 Linear algebra results

In this section we present some results from linear algebra that we will use throughout this thesis.

Definition 2.1.1. *$\lambda \in \mathbb{C}$ is an eigenvalue of $A \in \mathbb{R}^{n\times n}$ (or $\mathbb{C}^{n\times n}$) if a nonzero vector $v \in \mathbb{C}^n$ exists such that $Av = \lambda v$. The vector v is an eigenvector of A associated with λ. The set of all eigenvalues of A is called the spectrum of A and is denoted by $\sigma(A)$.*

The eigenvalues λ of a matrix A are also roots of the **characteristic polynomial** $\det(A - \lambda I) = 0$ and there are at most n distinct eigenvalues for a matrix A of size $n \times n$.

Furthermore, if matrix A is tridiagonal and Toeplitz, i.e.,

$$A = \begin{bmatrix} a & b & 0 \\ c & a & \ddots \\ 0 & \ddots & \ddots \end{bmatrix},$$

then the eigenvalues are known and can be determined by the formula [55, Theorem 2.2]

$$\lambda_k = a - 2\sqrt{bc}\cos\left(\frac{k\pi}{n+1}\right), \quad k = 1, 2, \ldots, n \tag{2.1}$$

where a is the entry on the main diagonal and b, c are the entries on the two off-diagonals of A. Using this, it is clear that we have complex eigenvalues if and only if $bc < 0$.

The modulus of the maximum eigenvalue is called the **spectral radius** and is denoted by $\rho(A)$, with

$$\rho(A) = \max_{\lambda\in\sigma(A)} |\lambda|. \tag{2.2}$$

Definition 2.1.2 (see [44])**.** *For $A \in \mathbb{R}^{n\times n}$ (or $\mathbb{C}^{n\times n}$),*

$$f(A) := \frac{1}{2\pi i}\int_\Gamma f(z)(zI - A)^{-1}dz,$$

where f is analytic on and inside a closed contour Γ that encloses the spectrum $\sigma(A)$.

For equivalent definitions of matrix functions, see [44, Ch. 1.2].

Definition 2.1.3. *A matrix $A \in \mathbb{R}^{n\times n}$ (or $\mathbb{C}^{n\times n}$) is diagonalisable if there exists an invertible matrix $Z \in \mathbb{C}^{n\times n}$ containing n right eigenvectors as columns such that*

$$Z^{-1}AZ = \mathrm{diag}(\lambda_1, \ldots, \lambda_n) =: \Lambda.$$

Otherwise, A is nondiagonalisable.

Definition 2.1.4. *A matrix $A \in \mathbb{R}^{n\times n}$ (or $\mathbb{C}^{n\times n}$), can be expressed as*

$$A = QUQ^*,$$

where Q is a unitary matrix, and U is an upper triangular matrix. This is called the Schur form of A.

A matrix $A \in \mathbb{R}^{n\times n}$ is said to be **normal** if there exists an orthonormal basis for the eigenvectors. There are equivalent characterisations of normality. For example, $A \in \mathbb{R}^{n\times n}$ is normal if and only if

$$A^TA = AA^T.$$

Otherwise, A is nonnormal.

When dealing with convection–diffusion problems in Chapter 4, we will be interested in how nonnormal the involved matrices are. There are several quantities that can be used to measure the degree of nonnormality and we discuss some of the most common here.

Given a diagonalisation of a matrix $A = Z\Lambda Z^{-1}$, then we say a matrix is highly nonnormal if the eigenvector condition number, $\kappa(Z) = \|Z\|\|Z^{-1}\| \gg 1$. Alternatively, we can characterise nonnormality using the field of values or pseudospectra.

Definition 2.1.5. *The field of values (or numerical range) of matrix $A \in \mathbb{C}^{n\times n}$ is given by*

$$W(A) = \{x^*Ax : x \in \mathbb{C}^n, x^*x = 1\}. \tag{2.3}$$

The field of values of a normal matrix is the convex hull of the eigenvalues [52]. Accordingly, a matrix is highly nonnormal if its field of values is much larger than the convex hull of the eigenvalues.

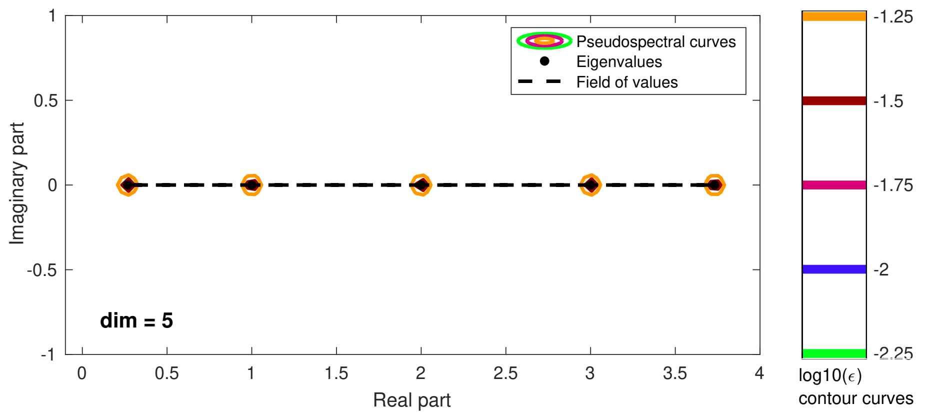
Another concept used for characterising nonnormality is the ϵ-pseudospectrum, which describes the region of the complex plane in which the eigenvalues may lie if the matrix A was perturbed by ϵ.

Definition 2.1.6 (see [85]). *Let $A \in \mathbb{C}^{n\times n}$ and $\epsilon > 0$ be arbitrary. The ϵ-pseudospectrum $\sigma_\epsilon(A)$ of A is the set of $z \in \mathbb{C}$ such that*

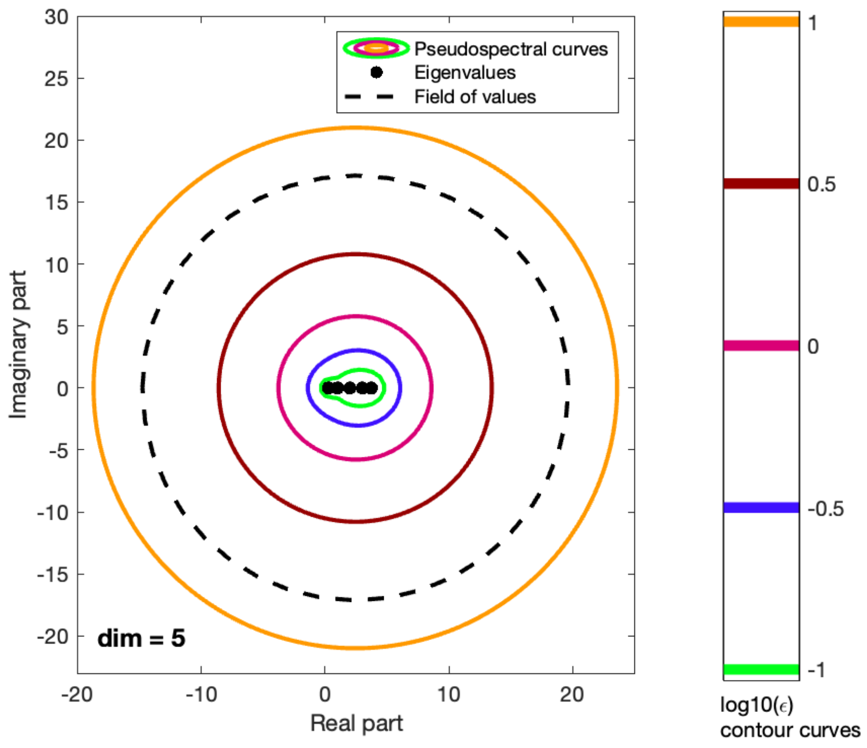
$$\|(zI - A)^{-1}\| > \epsilon^{-1}.$$

The ϵ-pseudospectrum of a normal matrix is the union of discs of radius ϵ centred at the eigenvalues of A [85]. If A is highly nonnormal, we expect the ϵ-pseudospectrum to enclose a much larger region of the complex plane.

We show in Figure 2.1 examples of the field of values and the pseudospectrum for both normal and nonnormal matrices. Both images have been obtained using the `eigtool` [94] package in `MATLAB`. We use small, 5×5 matrices, with equal eigenval-



(a) Example of field of values and pseudospectrum of a normal matrix T, with condition number 13.9282.



(b) Example of field of values and pseudospectrum of a nonnormal matrix G, with condition number 812.9143.

Figure 2.1: Examples of fields of values and pseudospectra for normal and nonnormal matrices.

ues to show this. The normal matrix is given by $T = \text{tridiag}(1, -2, 1)$, while the nonnormal matrix is given by $G = MTM^{-1}$, where M is a mildly ill-conditioned matrix. As expected, we can see that for the normal matrix, the field of values is represented by the convex hull of the eigenvalues (a straight line here), and the pseudospectrum is given by disjoint sets around each eigenvalue. For the nonnormal matrix, we see that the shapes cover a much larger area in the complex plane.

As this thesis focuses on solvers for matrix equations, it is important that we describe some preliminary results on Kronecker products, that connect linear systems and matrix equations.

Definition 2.1.7. *Let $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{p\times q}$. Then the Kronecker product (or tensor product) of A and B is the matrix*

$$A \otimes B = \begin{bmatrix} a_{11}B & \dots & a_{1n}B \\ \vdots & \ddots & \vdots \\ a_{m1}B & \dots & a_{mn}B \end{bmatrix} \in \mathbb{R}^{mp\times nq}.$$

The operator vec : $\mathbb{R}^{m\times n} \to \mathbb{R}^{mn}$ *is such that* $\mathrm{vec}(A)$ *is the vector obtained by placing the columns of the matrix A on top of one another.*

Proposition 2.1.1. *(see [56]) For any matrices of appropriate dimensions A, B, A_i, B_i, X, the following properties hold:*

$$(B^T \otimes A)\,\mathrm{vec}(X) = \mathrm{vec}(AXB) \tag{2.4}$$

$$(A_1 \otimes B_1)(A_2 \otimes B_2)\cdots(A_k \otimes B_k) = (A_1A_2\cdots A_k) \otimes (B_1B_2\cdots B_k). \tag{2.5}$$

For any function $f(\lambda)$ defined on the spectrum of a square matrix A, we have

$$\begin{aligned} f(A \otimes I) &= f(A) \otimes I, \\ f(I \otimes A) &= I \otimes f(A). \end{aligned} \tag{2.6}$$

Furthermore,

$$f(I \otimes A) f(A \otimes I) = f(A) \otimes f(A). \tag{2.7}$$

The eigenvalues of a Kronecker product $A \otimes B$ are products of the eigenvalues of the square matrices A and B [88]:

$$\sigma(A \otimes B) = \{\alpha_i \beta_j : \alpha_i \in \sigma(A), \beta_j \in \sigma(B)\}.$$

2.2 Discretisation of model problems

In this section we describe the discretisation of diffusion and convection–diffusion PDEs used in Chapters 3 and 4.

2.2.1 Diffusion PDE

A ubiquitous problem is the 2D Poisson PDE, which we will use as a model throughout Chapter 3:

$$-\Delta u = f, \quad \text{in} \quad \Omega = (0,1)^2, \quad \text{with} \quad u(x,y) = 0 \quad \text{on} \quad \partial\Omega. \tag{2.8}$$

We discretise the Poisson PDE (2.8) using second-order central finite differences in a square domain $\Omega = (0,1)^2$, as this discretisation leads to second-order local truncation error. Note that other finite difference schemes are possible, as well as other discretisation approaches. Throughout this section we follow the approach presented in [64] and, for completeness, outline the steps of the discretisation.

The idea behind the finite difference method is that we want to find an approximate numerical solution to a PDE. For simplicity of exposition, we use equally spaced grid points to present our discretisation, but we note that other meshes will be considered in our numerical examples in Section 3.6. Let our uniformly-spaced internal grid points be denoted by $\Omega_h = \{(x_i, y_j) : x_i = ih, y_j = jh; i, j = 1, 2, \ldots, n\}$, where $h = \frac{1}{n+1}$ is the grid spacing. Note that $i, j = 0$ and $i, j = n+1$ correspond to the boundary conditions, and since we consider only homogeneous Dirichlet conditions, we can ignore them here. To approximate the solution of the PDE, we use a grid function, X, which has the grid points as domain and which approximates the values of u. The value of X at the grid point (x_i, y_j) is denoted by $X_{i,j}$, with $i, j = 0, \ldots, n+1$. Note that $X_{i,j}$ now includes the boundary, unlike Ω_h, and that $X_{0,j} = X_{n+1,j} = X_{i,0} = X_{i,n+1} = 0$. Therefore, for each combination of $i, j = 1, 2, \ldots, n$, we have the approximations

$$u_{xx}(x_i, y_j) \approx \frac{X_{i-1,j} - 2X_{i,j} + X_{i+1,j}}{h^2} = \frac{1}{h^2} \begin{bmatrix} 1, -2, 1 \end{bmatrix} \begin{bmatrix} X_{i-1,j} \\ X_{i,j} \\ X_{i+1,j} \end{bmatrix},$$

and

$$u_{yy}(x_i, y_j) \approx \frac{X_{i,j-1} - 2X_{i,j} + X_{i,j+1}}{h^2} = \frac{1}{h^2} \begin{bmatrix} X_{i,j-1}, X_{i,j}, X_{i,j+1} \end{bmatrix} \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix}.$$

We can let X be an $n \times n$ matrix of the grid function values $X_{i,j}$. Similarly, we have a matrix $F \in \mathbb{R}^{n \times n}$ containing all values of the function f in (2.8) evaluated at each grid point. Note that here, and throughout, we assume that the resulting right-hand side matrix $F = CC^T$ has low rank, where $C \in \mathbb{R}^{n \times r}$ can be either a vector or a tall skinny matrix. We can then write the matrix form of the finite difference approximation [64], leading to the symmetric Lyapunov equation

$$AX + XA^T = F, \tag{2.9}$$

where

$$A = -\frac{1}{h^2}\begin{bmatrix} -2 & 1 & 0 & \\ 1 & -2 & \ddots & \\ 0 & \ddots & \ddots & \end{bmatrix} \in \mathbb{R}^{n\times n}. \tag{2.10}$$

We will present more information about the Lyapunov equation, as well as other matrix equations in Section 2.4. Note that we can write (2.9) as a linear system $\mathcal{A}\,\mathrm{vec}(X) = \mathrm{vec}(F)$, where $\mathcal{A}$ is given by the Kronecker formulation,

$$\mathcal{A} = I \otimes A + A \otimes I. \tag{2.11}$$

2.2.2 Convection–diffusion PDE

Another useful model problem is given by a convection–diffusion PDE. In this section, we present the finite difference discretisation of this PDE in two dimensions. It is given by

$$-\epsilon\Delta u + w \cdot \nabla u = f, \quad \text{on} \quad \Omega = (0,1)^2, \quad \text{with} \quad u(x,y) = 0 \quad \text{on} \quad \partial\Omega, \tag{2.12}$$

where $w = (w_1, w_2)$ is the convection wind, and ϵ is the positive and constant diffusion parameter. We are mainly interested in the convection-dominant case, where $|w| \gg \epsilon$, and we assume that w is incompressible, i.e., $\mathrm{div}(w) = 0$. Furthermore, we assume that the components of w are separable functions of the space variables, i.e., $w_1(x,y) = \phi_1(x)\psi_1(y)$, $w_2(x,y) = \phi_2(x)\psi_2(y)$, and that Ω is a square domain, $\Omega = (0,1)^2$. We will, once again, follow the derivation from [64].

Our set-up of the convection–diffusion problem is similar to that of the Poisson PDE in (2.8). In fact, note that the first term in (2.12) involves the Poisson operator. Accordingly, we will use the discretisation of the Poisson problem from Section 2.2.1 to discretise the convection–diffusion PDE. In particular, we will use the coefficient matrix for the Poisson problem given in (2.10). This means that we only need to present the discretisation for the second term in the convection–diffusion PDE: $w \cdot \nabla u = w_1 u_x + w_2 u_y$. Once again, we consider the set of internal grid points used for the approximation, $\Omega_h = \{(x_i, y_j) : x = ih, y = jh; i, j = 1, 2, \ldots, n\}$, where $h = \frac{1}{n+1}$. As in the discretisation of the diffusion problem, we use a finite difference scheme to discretise. Specifically, we use first-order central differences, as this is the simplest approach leading to second-order truncation error. Note that there exist other possible discretisations of the convection-diffusiuon problem. Since the finite difference solution can be obtained using the grid function X, we recall that X_{ij} represents the value of the grid function at a specific grid point (x_i, y_j), $i, j = 0, \ldots, n+1$. Using this, we then have

$$u_x(x_i, y_j) \approx \frac{1}{2h}(X_{i+1,j} - X_{i-1,j}) = \frac{1}{2h}\begin{bmatrix} -1 & 0 & 1 \end{bmatrix}\begin{bmatrix} X_{i-1,j} \\ X_{i,j} \\ X_{i+1,j} \end{bmatrix},$$

$$u_y(x_i, y_j) \approx \frac{1}{2h}(X_{i,j+1} - X_{i,j-1}) = \frac{1}{2h}\begin{bmatrix} X_{i,j-1} & X_{i,j} & X_{i,j+1} \end{bmatrix}\begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}.$$

The matrix B represents all the coefficients of the convection term that appear in the grid point approximation:

$$B = \frac{1}{2h}\begin{bmatrix} 0 & 1 & 0 & \\ -1 & 0 & \ddots & \\ 0 & \ddots & \ddots & \end{bmatrix} \in \mathbb{R}^{n \times n}. \tag{2.13}$$

Then, we can write the approximation to the convection–diffusion PDE from (2.12) as follows, using (2.9) to approximate the Poisson part of the PDE:

$$-\epsilon(AX + XA^T) + \Phi_1 BX\Psi_1 + \Phi_2 XB^T\Psi_2 = F, \tag{2.14}$$

where

$$\Phi_k = \mathrm{diag}(\phi_k(x_1), \phi_k(x_2), \ldots, \phi_k(x_n)),$$
$$\Psi_k = \mathrm{diag}(\psi_k(y_1), \psi_k(y_2), \ldots, \psi_k(y_n)),$$

$k = 1, 2$, and $F = CC^T$, as before. Note that, again, using (2.4), we can write (2.14) as a linear system $\mathcal{A}\,\mathrm{vec}(X) = \mathrm{vec}(F)$, where $\mathcal{A}$ is given by Kronecker formulation,

$$\mathcal{A} = -\epsilon(I \otimes A + A \otimes I) + \Psi_1^T \otimes (\Phi_1 B) + (B^T\Psi_2)^T \otimes \Phi_2. \tag{2.15}$$

2.3 Linear systems

A central theme in the history of numerical analysis is the progress in solving discretised PDEs efficiently. Depending on the size of the problem, different techniques and methods can be used. In this section, we present a short survey of the most commonly used methods for solving discretised PDEs as linear systems. A linear system takes the form

$$\mathcal{A}x = b, \tag{2.16}$$

where x and b are vectors and, in this thesis, $\mathcal{A}$ is a sparse, square, and invertible matrix. In the following sections we will present approaches for solving linear systems using direct, stationary iterative, and Krylov subspace methods.

2.3.1 Direct methods

In many cases, particularly when the coefficient matrix is small, direct methods based on LU or Cholesky factorisations are efficient as they can find the solution x accurately in a finite number of steps. Often, direct methods are optimised to exploit sparsity patterns and can be very fast [19, 25], however, if the matrices are very large (for example when they are impossible to store directly) or dense, then these approaches can be very costly.

We now briefly discuss one of the most straightforward direct methods for solving linear systems, the LU factorisation, based on Gaussian elimination. This involves decomposing the matrix $\mathcal{A}$ into a product of two simpler matrices, L and U, which are, respectively, lower and upper triangular. Once the factors are obtained, the algorithm consists of performing forward and backward substitutions [88]. If the matrix is Hermitian and positive definite, then the two factors, L and U can be made so that they are transposes of each other, so that $\mathcal{A} = LL^T$. This is called a Cholesky factorisation.

Often, the factors we obtain from an LU or a Cholesky factorisation are less sparse than the coefficient matrix $\mathcal{A}$, making these methods memory intensive for solving large linear systems. In order to deal with this issue, we can discard some of the elements that make our factor L denser than $\mathcal{A}$ [11]. These approaches are called incomplete factorisations, and a commonly considered one for Hermitian positive definite $\mathcal{A}$ is the incomplete Cholesky factorisation. Note that, even though we do not use these methods to directly solve linear systems, they can be useful when we discuss the development of preconditioners. We consider a dropping strategy based on sparsity. We choose a set, J, of index pairs, usually including the indices corresponding to the diagonal entries, and restrict all non-zero entries to places specified by these pairs of indices [88]. The LL^T decomposition would then be carried out as normal, with the only difference in the result that any unwanted entries whose indices are not in J are not included in L and L^T. This implies that we could store the entries in R, even though in practice we do not build this matrix. It is common to choose J to be the set of index pairs that correspond to non-zero entries in $\mathcal{A}$, such that $\mathcal{A} = LL^T + R$, where L is as sparse as the lower part of $\mathcal{A}$ and R is the residual matrix. Note that a similar approach is possible for LU factorisations of non-Hermitian or indefinite matrices $\mathcal{A}$.

The memory requirements and computational costs associated with using direct methods for solving large linear systems make these approaches expensive, and possibly unfeasible, when compared to other methods that we will cover. However, they are worth mentioning as the factors obtained are commonly used as preconditioners, as we will present in Section 2.3.4.

2.3.2 Stationary iterative methods

Iterative methods are a common approach for solving sparse linear systems of varying dimensions. The simplest such techniques are stationary iterative methods, and they can form a basis for more sophisticated algorithms. These usually require an initial guess which is then updated by performing a number of steps of the stationary method.

Understanding stationary iterative methods will be useful for us in Chapter 4, so we will briefly discuss the general case. We look to split the coefficient matrix $\mathcal{A}$ into two different simpler matrices, M and N, such that $\mathcal{A} = M - N$, with M nonsingular and easier to invert than $\mathcal{A}$. This means that a linear system of the form (2.16) becomes $(M - N)x = b$, i.e., $Mx = Nx + b$. Using this notation allows us to generate a sequence of iterates using $Mx_k = Nx_{k-1} + b$, i.e., a fixed point iteration:

$$x_k = M^{-1}Nx_{k-1} + M^{-1}b. \tag{2.17}$$

As M and N do not depend on k, we call these methods where we split matrix $\mathcal{A}$, stationary iterative methods, or stationary splittings. To develop some specific iterative methods mentioned, we use the notation $\mathcal{A} = D + L + U$, where D is the matrix containing only the diagonal of $\mathcal{A}$, L is the strictly lower triangular part of $\mathcal{A}$ and U is the strictly upper triangular part. Choosing different combinations of D, L and U for M and N gives different iterative methods.

One of the most common and simple to implement stationary iterative methods is the Jacobi method. For this, $M = D$ and $N = -(L + U)$. Computing the approximate solution x_k is simple as D is a diagonal matrix and straightforward to invert and so, having computed D^{-1}, we only require some matrix-vector multiplications at each Jacobi iteration. Note that, in practice, for large linear systems, these products can be quite costly to perform, but very straightforward. An extension to the Jacobi stationary splitting is obtained if we introduce a relaxation parameter, ω and solve $\omega\mathcal{A}x = \omega b$ instead. Then, the splitting is given by $\omega A = M - N$, with $M = D$ and $N = (1 - \omega)D - \omega(L + U)$, and it is called the damped Jacobi method.

We next describe two Gauss–Seidel methods. For the forward Gauss–Seidel approach we traverse the finite difference grid from left to right, and so we choose $M = D + L$ and $N = -U$ as the splitting for $\mathcal{A}$. Similarly, the backward Gauss–Seidel method uses the splitting $M = D + U$ and $N = -L$. In this case, on the finite difference grid, the values are updated from right to left. Both approaches are still straighforward, as the matrices M are either lower or upper triangular.

Another very common stationary splitting is the successive over-relaxation (SOR) iteration. This is generated by introducing a relaxation parameter $\omega \in (0, 2)$ in the Gauss–Seidel iteration. Specifically, we consider $\omega \mathcal{A} x = \omega b$ and introduce the splitting $\omega \mathcal{A} = M - N$, with $M = D + \omega L$ and $N = (1 - \omega) D - \omega U$. With $\omega = 1$, this is the equivalent of the forward Gauss–Seidel method [72, Ch. 4.1], and note that in a similar manner, we can generate an SOR iteration using the backward Gauss-Seidel method.

The convergence of a stationary iterative method (2.17) is strongly connected to the iteration matrix $G = M^{-1} N$. To be precise, since $x_k = G x_{k-1} + M^{-1} b$, the error at the kth iteration, $e_k = x - x_k$, is given by $e_k = G^k e_0$. Thus, the sequence of iterates, $(x_k)_{k=0}^{\infty}$ converges if $G^k \to 0$ as $k \to \infty$. Moreover, a sufficient condition for convergence is that the spectral radius $\rho(G) < 1$ [72, Ch. 4.2]. Finally, the spectral radius also dictates the asymptotic speed of convergence of these methods – the nearer it is to 1, the slower the convergence rate will be in general. For PDE problems, the convergence of stationary iterative methods can be very slow, and so, better methods for solving those linear systems have been sought.

2.3.3 Multigrid methods

Another approach for solving linear systems is given by multigrid methods. The description below follows that of [83]. The motivation for this class of methods is that when the linear system arises from discretising a PDE, we can compute the solution to a linear system (2.16) by using mesh information from different discretisations of the PDE. This is done by approximating the coefficient matrix and residuals on a coarser grid, solving the cheaper coarse grid problem, and then transforming the solution back to the initial fine grid. As a result, we replace the $n^2 \times n^2$ linear system with a $N \times N$ one, where $N < n^2$.

The first step of a multigrid method consists of relaxing, or smoothing, the approximation. Typical relaxation iterative processes include the Gauss–Seidel or SOR methods mentioned above, or more sophisticated smoothers such as incomplete LU factorisations. On their own, these methods converge rather slowly for large linear systems associated with PDEs, but here a few iterations before moving between grids can improve the initial guess, as well as making the initial solution smoother so that the residual can be well approximated on a coarse grid. If multigrid is used as a preconditioner, then it is common to also use a post-smoothing step for symmetric problems to ensure that the multigrid preconditioner is symmetric.

At the core of a multigrid iteration, we perform a number of intergrid operations. We begin with the fine grid, where the problem size is $n^2 \times n^2$. We then perform a restriction operation, denoted by R that transfers the matrix and residual from the fine grid to a coarser one. If it is small enough, the coarse grid problem is then solved directly, otherwise, further smoothing and restriction is performed until a small enough problem is obtained. The problem is then transfered back to the fine grid via a prolongation operator, P. For our purpose, $R = P^T$. More details about how multigrid methods can be generated and used in practice can be found in e.g., [14, 72].

There are different ways to traverse the grid hierarchy. Common methods are what is known as V-cycles and W-cycles. Note that in our numerical results we use a V-cycle, which only moves from a fine grid down to a coarse one and back up again, as in Figure 2.2, where h represents the finest grid, and all multiples of

h represent coarser grids.

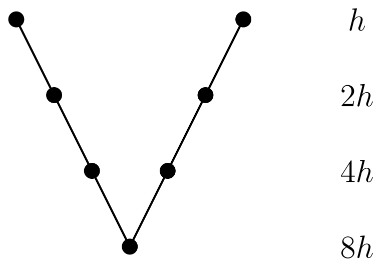


Figure 2.2: V-cycle with 4 levels

Multigrid methods are an attractive solver for large linear systems, especially those coming from discretised PDEs, with the computational work being proportional to the dimension of the discrete system for many problems [28]. This means that the convergence rate does not depend on the size of the mesh used. This is called mesh independence and is a characteristic that makes multigrid methods effective. Note that multigrid does not work for all problems, however. For example, a V-cycle is actually a stationary iterative method, which means that it can diverge or have slow convergence [28]. Despite the popularity and efficiency of multigrid methods as solvers for the discretised PDEs we are interested in, in this thesis we use multigrid as a preconditioner within the conjugate gradient and GMRES methods.

2.3.4 Krylov subspace methods and preconditioning

Another class of methods whose dominant cost is given by vector-matrix products [46] are Krylov subspace methods. These are some of the most commonly used methods for solving linear systems, and over time, a lot of research has focused on these methods. The Krylov subspace of degree k generated with a matrix $\mathcal{A}$

and a vector b is given by

$$\mathcal{K}_k(\mathcal{A}, b) = \text{span}\{b, \mathcal{A}b, \mathcal{A}^2 b, \dots, \mathcal{A}^{k-1}b\}. \tag{2.18}$$

Clearly, the elements of $\mathcal{K}_k(\mathcal{A}, b)$ are of the form $p_{k-1}(\mathcal{A})b$, where $p_{k-1}(\mathcal{A}) \in \mathcal{P}_{k-1}$ is a polynomial in $\mathcal{A}$ of degree at most $k-1$. When $x = \mathcal{A}^{-1}b$ can be well approximated by a low degree polynomial, then the Krylov subspace represents a good space in which to seek an approximate solution of the linear system [46]. This often is not the case for unpreconditioned PDE problems, hence the slow convergence rates. Although $x = \mathcal{A}^{-1}b$ can be expressed as a polynomial of degree at most s, where s is the degree of the minimum polynomial of a matrix $\mathcal{A}$, we typically want a good approximation of $\mathcal{A}^{-1}b$ in fewer iterations. This means we want to minimise the number of iterations taken for Krylov subspace algorithms to converge.

Starting from an initial guess x_0, Krylov subspace methods work by generating a sequence of approximate solutions x_k of $\mathcal{A}x = b$ such that $x_k \in x_0 + \mathcal{K}_k(\mathcal{A}, r_0)$, where $r_0 = b - \mathcal{A}x_0$ is the initial residual. The iterative process ends when the corresponding residual r_k is small enough. We mentioned that the solution x can be expressed as a polynomial of the same degree as the minimum polynomial of $\mathcal{A}$. This means that a Krylov subspace method has an opportunity to converge fast if the minimal polynomial has low degree s [46], or is well-approximated by a low degree polynomial. In Section 2.3.5 we will see in more detail some commonly used Krylov subspace solvers.

Preconditioning

The convergence of Krylov subspace methods can be very slow for large linear systems, including those arising from discretised PDEs. In order to combat the issue of slow convergence, preconditioning techniques can be implemented. Suppose we are given a matrix P that approximates the original coefficient matrix $\mathcal{A}$, with the property that P is easier to invert than $\mathcal{A}$. Then, $\mathcal{A}x = b$ is equivalent to $P^{-1}\mathcal{A}x = P^{-1}b$, but the convergence rate of a given Krylov subspace method may be much faster for the latter system. This is known as left preconditioning. Right or split preconditioning could be performed instead.

Developing good preconditioners is important for quick convergence of preconditioned Krylov subspace methods. There are two things to take into consideration. First, we needed to choose our preconditioner P such that $P^{-1}\mathcal{A}$ gives a system that is easier to solve than the initial one. Secondly, our preconditioner needs to be easy to construct and apply [11].

A common situation in practice is given by the case when the preconditioner is available in factored form $P = P_1P_2$, where P_1 and P_2 are, for example, triangular factors of an incomplete LU decomposition. In these cases, we precondition the linear system as follows

$$P_1^{-1}\mathcal{A}P_2^{-1}y = P_1^{-1}b, \tag{2.19}$$

where $x = P_2^{-1}y$ [72]. This is often called split preconditioning. We often want our preconditioned system to be symmetric. This can be achieved by split preconditioning, as in (2.19), with $P_1 = P_2^T$. We note that when using preconditioned conjugate gradient or preconditioned MINRES, it is sufficient to have (a symmetric positive definite) P, i.e.,, P_1 and $P_2 = P_1^T$ are not required.

We now mention three straightforward preconditioners which are commonly used in practice.

- A simple and straighforward set of preconditioners is given by the stationary splittings from Section 2.3.2. Recall that we wrote $\mathcal{A} = M - N$, where M and N were chosen to be different combinations of the diagonal, upper and lower triangular parts of $\mathcal{A}$. This allows for any M coming from stationary splittings to be used as a preconditioner.

- In Section 2.3.1 we briefly discussed incomplete factorisations. Note that these can be used as preconditioners. For example, for symmetric positive definite problems, the idea behind preconditioning with incomplete Cholesky is to set our preconditioner to be $P = LL^T$. Although $\mathcal{A}$ and P are not the same matrix, we hope that P^{-1} will approximate $\mathcal{A}^{-1}$ well. Similarly, for nonsymmetric problems, we can precondition with the incomplete LU factorisation.

- A powerful preconditioner used for improving the convergence rate of Krylov methods comes from multigrid methods as these are often good approximations of the action of $\mathcal{A}^{-1}$ on a vector. Depending on the multigrid implementation details, a variety of multigrid preconditioners can be obtained. Details about multigrid preconditioners can be found in, for example, [83].

2.3.5 The conjugate gradient method and GMRES

In this section we present two Krylov subspace solvers, the conjugate gradient method, for linear systems with symmetric positive definite (SPD) coefficient matrices, and GMRES for nonsymmetric systems. We will present the algorithms and discuss the convergence of these methods.

Symmetric positive definite problems: the conjugate gradient method

For linear systems with SPD $\mathcal{A}$ arising from the discretisation of PDEs, a commonly used solver is the conjugate gradient (CG) method [42].

Solving the linear system $\mathcal{A}x = b$ is equivalent to minimising the linear functional $\Phi : \mathbb{R}^n \to \mathbb{R}$ [88]

$$\Phi(x) = \frac{1}{2}x^T\mathcal{A}x - x^Tb.$$

In CG, the iterates are given by

$$x_k = x_{k-1} + \alpha_k p_k,$$

where the search directions p_k are such that the value of $\Phi(x_{k-1} + \alpha p_k)$ is minimised over all $p_k \in \mathbb{R}^n$. The CG method must terminate in exact arithmetic, i.e., it must reach the exact solution in a finite number of steps because $x = \mathcal{A}^{-1}b$ lies in the Krylov subspace $\mathcal{K}_n(\mathcal{A}, b)$. This means that $x_n = x$ and so the conjugate gradient algorithm in Algorithm 1 must stop at the nth step.

The conjugate gradient method minimises the $\mathcal{A}$-norm of the error over the Krylov subspace, so in the following we briefly discuss the accuracy and convergence of

Algorithm 1 The conjugate gradient algorithm

```
1: Choose $x_0$, and let $r_0 = b - \mathcal{A}x_0$
2: for $k = 1, 2, \dots$ do
3:     Check convergence and continue if necessary
4:     if k=1 then
5:         $p_1 = r_0$
6:     else
7:         $\beta_k = r_{k-1}^T r_{k-1} / r_{k-2}^T r_{k-2}$
8:         $p_k = r_{k-1} + \beta_k p_{k-1}$
9:     end if
10:    $\alpha_k = r_{k-1}^T r_{k-1} / p_k^T \mathcal{A} p_k$
11:    $x_k = x_{k-1} + \alpha_k p_k$
12:    $r_k = r_{k-1} - \alpha_k \mathcal{A} p_k$
13: end for
```

the method. The error at step k is given by

$$e_k = x - x_k = q_k(\mathcal{A})e_0,$$

where $q_k(z) = 1 - zq_{k-1}(z)$, and it is bounded in the $\mathcal{A}$-norm as follows:

$$\frac{\|e_k\|_{\mathcal{A}}}{\|e_0\|_{\mathcal{A}}} \leq \min_{\substack{q \in \mathcal{P}_k \\ q(0)=1}} \max_{\lambda \in \sigma(\mathcal{A})} |q(\lambda)|.$$

We can see from this error bound that the convergence of CG is dependent on the distribution of the eigenvalues of $\mathcal{A}$.

The $\mathcal{A}$-norm error at step k is minimised over the Krylov subspace and there exists a characterisation of the polynomial that achieves the minimisation [28]. In practice, the method can lose accuracy due to rounding errors, and cancellation error can cause the search vectors to lose $\mathcal{A}$-orthogonality. As a result, the conjugate gradient method on its own may still require a large number of iterations to converge. Even in exact arithmetic, convergence of the unpreconditioned algorithm is typically slow for PDE problems, and is not recommended for large linear systems.

The convergence rate of the conjugate gradient method can be improved by using a preconditioner. We call the CG method which allows for a preconditioner the preconditioned conjugate gradient method, or PCG. Because we must ensure that the preconditioned coefficient matrix is symmetric, we apply the preconditioner symmetrically as in (2.19), with $P = P_1 P_1^T$. Each of the three types of preconditioner mentioned in Section 2.3.4 can be used with the PCG solver. More details including algorithms and convergence details can be found in, e.g., [72].

Nonsymmetric problems: GMRES

Nonsymmetric linear systems, such as those arising from the discretisation of convection–diffusion PDEs, require different solvers than those mentioned above. Recall that the conjugate gradient method minimises the error in the $\mathcal{A}$-norm over the Krylov subspace. However, when the coefficient matrix $\mathcal{A}$ is no longer symmetric, it no longer defines a norm, which means that the same minimisation strategy cannot be used. Furthermore, for nonsymmetric problems, we cannot easily measure the error in any norm. Hence, it seems reasonable to consider choosing x from the Krylov subspace in such a way that we minimise the residual with respect to the 2-norm.

One of the most popular methods that allows us to solve nonsymmetric linear systems is the generalised minimal residual (GMRES) method [73]. GMRES is a projection method into the Krylov subspace $\mathcal{K}_k(\mathcal{A}, r_0)$, where $r_0 = b - \mathcal{A}x_0$. We present the steps in Algorithm 2 as in [72, Algorithm 6.9]. Furthermore, note that GMRES can be used on its own, as well as with a preconditioner.

Algorithm 2 GMRES algorithm

```
 1: Choose $x_0$, and compute $r_0 = b - \mathcal{A}x_0, \beta = \|r_0\|_2, v_1 = r_0/\beta$
 2: for $j = 1, 2, \dots, m$ do
 3:     Compute $w_j = \mathcal{A}v_j$
 4:     for $i = 1, \dots, j$ do
 5:         $h_{ij} = (w_j, v_i)$
 6:         $w_j = w_j - h_{ij}v_i$
 7:     end for
 8:     $h_{j+1,j} = \|w_j\|_2$. If $h_{j+1,j} = 0$ set $m = j$ and go to step 11
 9:     $v_{j+1} = w_j/h_{j+1,j}$
10: end for
11: Define the $(m+1) \times m$ Hessenberg matrix $\bar{H}_m = \{h_{ij}\}_{1\le i\le m_1, 1\le j\le m}$.
12: Compute $y_m$, the minimiser of $\|\beta e_1 - \bar{H}_m y\|_2$ and $x_m = x_0 + V_m y_m$.
```

The convergence of GMRES is often described using three distinct approaches: eigenvalues and the eigenvector condition number, the field of values, and pseudospectra [29].

- If matrix $\mathcal{A}$ is diagonalisable and not too far from normal, in the sense that the eigenvector condition number, $\kappa(Z)$, is not too large and if a polynomial q whose size decreases quickly on the spectrum of $\mathcal{A}$ can be found, then GMRES converges quickly [84] and the convergence rate can often be described by the eigenvalues of $\mathcal{A}$ [60]. The eigenvector bound is given by

$$\frac{\|r_k\|_2}{\|r_0\|_2} \le \min_{\substack{q\in\mathcal{P}_k\\ q(0)=1}} \|Z\|_2\|Z^{-1}\|_2\|q(\Lambda)\|_2 = \kappa(Z) \min_{\substack{q\in\mathcal{P}_k\\ q(0)=1}} \max_{\lambda\in\sigma(A)} |q(\lambda)|.$$

- For matrices which have eigenvalues that are highly sensitive to small perturbations in the matrix entries, we can use the field of values, $W(\mathcal{A})$, with the bound given by [29]

$$\frac{\|r_k\|_2}{\|r_0\|_2} \le 2 \min_{\substack{q\in\mathcal{P}_k\\ q(0)=1}} \max_{z\in W(\mathcal{A})} |q(z)|.$$

- The ϵ-pseudospectrum of $\mathcal{A}$ can also be used to derive a GMRES bound. This, with $\mathcal{L}(\Gamma_\epsilon)$ as the contour length of the boundary of $\sigma_\epsilon(\mathcal{A})$, is given by [29]
$$\frac{\|r_k\|_2}{\|r_0\|_2} \leq \frac{\mathcal{L}(\Gamma_\epsilon)}{2\pi\epsilon} \min_{\substack{q\in\mathcal{P}_k \\ q(0)=1}} \max_{z\in\sigma_\epsilon(\mathcal{A})} |q(z)|.$$

2.4 Matrix equations

For large problems, the high degree of sparsity means that the linear system $\mathcal{A}x = b$ from (2.16) can still be solved with good preconditioners and iterative methods. Even so, this can require a significant amount of time and memory for convergence. As a result, numerous methods have been developed to work directly with the matrix equation formulations, e.g., (2.9) and (2.14). We describe below the matrix equation framework and discuss some well-established methods for solving matrix equations.

Definition 2.4.1 (General linear matrix equation). *For $A_i \in \mathbb{R}^{n\times n}$, $B_i \in \mathbb{R}^{m\times m}$, with $i = 1,\ldots,k$ and $F \in \mathbb{R}^{n\times m}$, a linear matrix equation is given by*
$$A_1XB_1 + A_2XB_2 + \cdots + A_kXB_k = F.$$

Any linear matrix equation can be written in Kronecker form to get a linear system formulation. For the general linear matrix equation this is given by $\mathcal{A}\,\mathrm{vec}(X) = \mathrm{vec}(F)$, where $\mathcal{A} = B_1^T \otimes A_1 + \cdots + B_k^T \otimes A_k$. We will mainly consider matrix equations with $k = 2$, called Lyapunov or Sylvester equations, depending on the characteristics of $A_{1,2}$ and $B_{1,2}$, as the PDEs described in Section 2.2 often result in these matrix equations. The Lyapunov and Sylvester equations are presented below. There are, of course, other types of matrix equations, but these are beyond the scope of this thesis.

Definition 2.4.2 (Lyapunov equation)**.** *For $A \in \mathbb{R}^{n\times n}$ and $F \in \mathbb{R}^{n\times n}$, a continuous-time, algebraic Lyapunov equation is of the form*

$$AX + XA^T = F.$$

Definition 2.4.3 (Sylvester equation)**.** *For $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{m\times m}$ and $F \in \mathbb{R}^{n\times m}$, a continuous-time, algebraic Sylvester equation is of the form*

$$AX + XB = F. \tag{2.20}$$

The solvability of the Lyapunov and Sylvester equations with right-hand side $F \neq 0$ is described below.

Remark 2.4.1 (Solvability)**.** *For the Lyapunov equation to have a nontrivial solution, it is sufficient to have an invertible coefficient matrix A.*
The Sylvester equation has a nontrivial solution if the spectra of A and $-B$ are disjoint. Furthermore, the solution X of the Sylvester equation can be written in closed form as an integral of exponentials [77]

$$X = -\int_0^\infty \mathrm{e}^{At}\mathrm{e}^{Bt}\mathrm{d}t,$$

where e^{Mt} is the matrix exponential of Mt.

Note that other representations of the solution can be found in [77].

In Section 2.2 we discretised both the Poisson and the standard convection–diffusion PDEs. We note now that solving the discretised Poisson PDE corresponds to solving a Lyapunov equation with SPD coefficient matrix A. Solving the convection–diffusion PDE, however, can correspond to solving one of three matrix equations, depending on the separable convection wind, $w = (\phi_1(x)\psi_1(y), \phi_2(x)\psi_2(y))$. If this is a constant, i.e., $w = \alpha(1, 1)$ then we have to solve a Lyapunov equation with nonnormal coefficient matrix A. If the wind looks like $w = (\phi_1(x), \psi_2(y))$, then we have to solve a Sylvester equation, and if we have wind $w = (\phi_1(x)\psi_1(y), \phi_2(x)\psi_2(y))$, then the matrix equation has four terms, as in (2.14), with contributions from both the diffusion and convection parts of the PDE.

We distinguish approaches for solving matrix equations based on the size of the problem to which they are applied. For a small matrix A, there are well established direct methods, while for matrix equations with large coefficient matrices, the ADI method [68] and projection methods are often used. We will briefly present each of these in the following and justify our choices for this thesis.

2.4.1 Direct methods

When the size of Lyapunov or Sylvester equation is relatively small (say, n is up to a thousand), they can be solved using a set of direct methods [3, 36] based on Schur decompositions. The most well-known of these is the Bartels–Stewart method [3], which is implemented in the built-in `MATLAB` function `lyap`. The Bartels–Stewart method first reduces the coefficient matrices to real Schur form using a QR transformation. The resulting upper triangular matrices are then used to transform the matrix equation into one with upper triangular structure. The new matrix equation is then solved element by element using backward substitution [58, 77]. The steps of the algorithm for a Sylvester equation are given in Algorithm 3.

Algorithm 3 Main steps of the Bartels–Stewart method for the Sylvester equation $AX + XB = F$ [77]

```
1: Compute the Schur forms: $A^T = URU^T, B = VSV^T$ with $R, S$
          upper triangular;
2: Solve by substitution $R^TY + YS = U^TFV$ for $Y$;
3: Compute $X = UYV^T$.
```

For large (n is greater than a few thousand) matrix equations, the Bartels–Stewart method is not efficient because of the increased computational cost of generating the Schur forms explicitly [77]. As a result, other methods, such as the ADI method and projection-based approaches, need to be employed to deal with large equations.

2.4.2 ADI

When the coefficient matrix A of a Lyapunov equation is large and sparse, one can use the alternating direction implicit (ADI) method introduced by Peaceman and Rachford [68] for solving elliptic linear systems. This method was adapted to deal with matrix equations in [26], with many theoretical and practical advances obtained, especially by Ellner, Wachspress and Lu, e.g., [27, 58, 90].

The main steps of the ADI iteration for computing factored iterates $X_j = Z_jZ_j^T$ of the solution of a Lyapunov equation $AX + XA^T = bb^T$ with rank-1 right-hand side are

$$\begin{aligned} X_0 &= 0 \\ (A + s_jI)X_{j-\frac{1}{2}} &= bb^T - X_{j-1}(A^T - s_jI) \\ (A + s_jI)X_j &= bb^T - (X_{j-\frac{1}{2}})^T(A^T - s_jI), \quad j = 1, \dots, k. \end{aligned}$$

The s_j are poles which can be real or complex and which are implemented cyclically, as we describe next. At each iteration, the algorithm uses one pole. If the algorithm has used all the poles available, but has not converged yet, then it continues using the original poles in the same order, repeating this until convergence. For more details about this iteration, the selection of poles, as well as the ADI method for Sylvester equations, see [77] and the references therein.

The ADI method is very well studied for matrix equations and pole choices are extensively considered in the literature, see e.g., [9, 31, 74]. However, theoretical comparisons between the ADI method and projection-based methods indicate that projection-based aproaches can be better for a wider variety of pole choices than ADI, with ADI giving much larger residuals than rational projections for poor poles [77]. Therefore, in this thesis, we focus on solving matrix equations with a rational projection which we will describe in future chapters. Note that, in practice, good poles for the rational Krylov projection method are often good poles for the ADI method, so although we are not explicitly considering the ADI method, the pole choices we present could also be useful in the context of ADI.

2.4.3 Projection methods

As for linear systems, another effective approach for solving large matrix equations is to project onto a smaller space, solve the reduced problem and then to project back the solution into the original space [48, 51, 75, 82].

In order to present projection methods, we require knowledge of Krylov subspaces. Recall that the standard (polynomial) Krylov subspace in (2.18) is $\mathcal{K}_k(A, b) = \text{span}\{b, Ab, A^2b, \ldots, A^{k-1}b\}$. In this thesis, we derive results for rational Krylov subspaces. For a given matrix A, right-hand side b and possibly repeated poles $\mathcal{S} = \{s_1, s_2, \cdots, s_k\}$, we define the k-dimensional rational Krylov subspace as

$$\begin{aligned} \mathcal{RK}_k(A, b, \mathcal{S}) &= \left\{b, (A + s_1 I)^{-1} b, \ldots, \prod_{i=1}^{k} (A + s_i I)^{-1} b\right\} \\ &= \mathcal{K}_k(A, q_k^{-1}(A) b), \end{aligned} \tag{2.21}$$

where $q_k(z) = (z + s_1) \cdots (z + s_k)$. Note that we can recover the standard Krylov subspace from (2.21) by setting $s_i = \infty$, for $i = 1, \ldots, k$. Furthermore, we can also obtain the extended Krylov subspace [20]

$$\begin{aligned} \mathcal{EK}_k(A, b) &= \mathcal{K}_k(A, b) + \mathcal{K}_k(A^{-1}, A^{-1} b) \\ &= \operatorname{span}\{b, A^{-1} b, Ab, A^{-2} b, A^2 b, \ldots, A^{-k} b, A^{k-1} b\}, \end{aligned} \tag{2.22}$$

by using poles that alternate between 0 and ∞.

Besides these specific choices of poles which generate the standard and extended Krylov subspaces, it is common to use the so-called Zolotarev poles for the rational Krylov subspace. These poles are considered near-optimal for many methods that aim to solve matrix equations [74]. More details about Zolotarev poles, as well as other methods for generating poles will be discussed in later chapters.

The space used for projection methods for matrix equations can be one of the three presented above, with the extended [64, 76, 82] and rational [21, 23, 71] Krylov subspace methods being the most popular.

Solving discretised PDEs as matrix equations instead of linear systems is strongly connected with the existence of low-rank factors of the solution. This property is known to have been exploited by numerous algorithms which output approximate solutions that have singular values decaying rapidly to zero. Often, this is ensured by right–hand sides which have low-rank. In the literature, low-rank approximations are known to exist for many standard Lyapunov and Sylvester equations [57, 59, 65, 89, 92], as well as multiterm matrix equations [71, 75, 77]. For more details see, for example, [69]. Note that if this low-rank property is not satisfied, then matrix equations might not be the most efficient way to approximate the solution to discretised PDEs, and so solving linear systems might be more appropriate. However, as we will see for a test problem in the next section, solving matrix equations via projections into rational Krylov subspaces provides a computationally superior approach to solving linear systems for cases when the solution allows low-rank factors, and so, this motivates our choice of studying matrix equations for this type of problem.

2.5 Linear systems vs matrix equations

In this section, we are interested in comparing the two main ways to solve discretised PDEs discussed in this chapter, namely linear systems and matrix equations.

2.5.1 Diffusion problem

We begin by considering the Poisson PDE, which gives the discretised problem in Section 2.2.1. We will solve the linear system with matrix given by (2.16) using PCG with a multigrid preconditioner consisting of a single V-cycle, which has three damped Jacobi iterations for both pre- and post–smoothing, with a damping parameter $\omega = 2/3$ and four grid levels. The matrix equation is solved using a rational Krylov subspace with 12 Zolotarev poles, and a right-hand side given by ones. Details of the matrix equation solver are given in Chapter 3. The size of the matrix equation coefficient matrix is $n = 1023$, so that the corresponding linear system coefficient matrix is of size $n^2 = 1023^2$. We choose this size for the problem because it is convenient for the prolongation and restriction operators in our multigrid preconditioner to have $n = 2^k - 1$ for some integer k [83]. We compare both the number of iterations and the time taken to solve these problems, and note that we stop the solvers when $\|r_k\|_2/\|\text{vec}(F)\|_2 < 10^{-8}$ for the linear system, with $r_k = \text{vec}(F) - \mathcal{A}x_k$ and $\|R_k\|_F/\|F\|_F < 10^{-8}$ for the matrix equations, where $R_k = F - AX_k - X_kA$. Note that the stopping criteria for the two solvers are equivalent, so that our comparison is relevant. In Figure 2.3 we see the convergence curves of both methods.

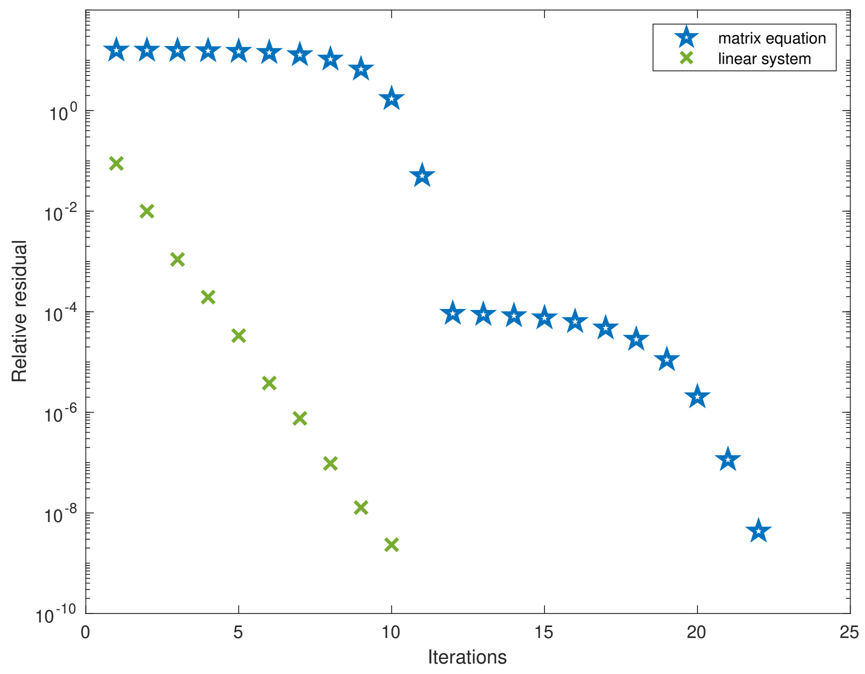


Figure 2.3: Convergence curves for the discretised Poisson PDE from Section 2.2.1 solved using the preconditioned conjugate gradient method and the matrix equation solver described in Chapter 3.

We can clearly see that the number of iterations necessary for PCG to solve the linear system is about half of that necessary for the matrix equation solver. It may seem like the linear system solver is superior, however, when we take into consideration the computational time required, the matrix equation solver is almost 13 times faster, taking 0.71 seconds, as compared to 8.92 seconds for PCG. It shows that, for SPD problems, matrix equations can show superior computational times, therefore, making them a preferable approach.

2.5.2 Convection–diffusion problem

Since we are interested in not only diffusion problems, but also convection–diffusion PDEs, we will now compare the efficiency of solving such problems via linear systems and matrix equations. We will consider the simplified convection–diffusion PDE from (2.12), with wind given by $w = (\phi_1(x), \psi_2(y)) = 1 - (2x+1)^2, 1 - y^2)$. The matrix equation corresponding to this has the form $-\epsilon(AX + XA^T) + \Phi_1 BX + XB^T\Psi_1 = CC^T$, which is a Sylvester equation. We will use $\epsilon = 0.0167$ for these tests so that we are in a convection-dominated case. We choose the right–hand side to be a vector of ones, and the size to be $n = 1000$. This means that we will solve a linear system of size $n^2 = 10^6$. For this, we will use GMRES with the algebraic multigrid preconditioner MI20 from [13]. In the setup of the MI20 preconditioner, we have made the following changes to the default settings: `control.smoother` was set to 1, corresponding to the damped Jacobi smoother, `control.damping` was set to 2/3 and `control.v_iterations` was set to 2, meaning we perform 2 V-cycles. The initial guess throughout was set to the zero vector, and the GMRES relative residual tolerance used was 10^{-8}. For the matrix equation formulation, we will use a rational Krylov subspace approach with 12 Zolotarev poles generated using the left-hand side matrix $-\epsilon A + \Phi_1 B$. As for the diffusion problem in Section 2.5.1, we solve the problems at a relative residual tolerance of 10^{-8}. Note that the stopping criteria are equivalent, so that the two solvers have the same solution accuracy. The convergence curves are shown in Figure 2.4.

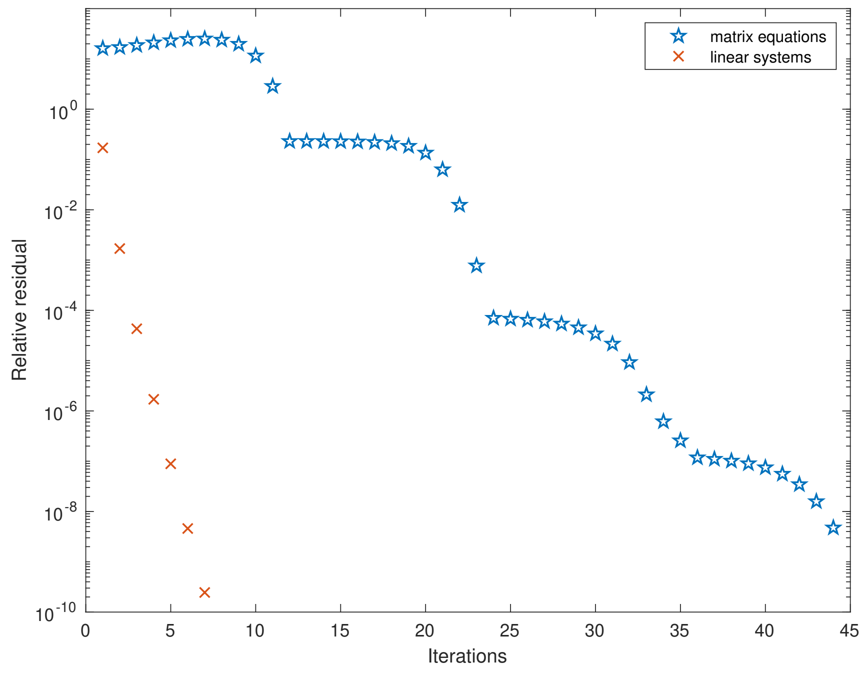


Figure 2.4: Convergence curves for the discretised convection–diffusion PDE from Section 2.2.2 solved using GMRES and the Sylvester equation solver described in Chapter 4.

As in the diffusion case, we can see in Figure 2.4 that the number of iterations taken to solve the matrix equation is larger than that necessary to solve the linear system formulation. However, once again, it is the computational time that makes the more compelling argument as to which approach is more efficient. The matrix equation method took 1.5762 seconds, while the linear system took 5.0522 seconds. This means that the matrix equation approach was 3.2 times faster for this problem. This is, indeed, a smaller ratio than when we looked at the diffusion problem, but it still indicates that solving matrix equations can be superior to solving the equivalent linear system.

Chapter 3

Diffusion problems

In this chapter we present theoretical and numerical results for matrix equations arising from the discretisation of diffusion problems. The theoretical results we discuss hold for all Lyapunov equations, however, our numerical results will focus on two-dimensional diffusion equations on tensor product grids, with both rank-1 and higher rank right-hand sides. The work described in Sections 3.2.3, 3.5 and 3.6 is also presented in the submitted paper [86].

3.1 Introduction

We have seen in the literature presented in Section 2.4.3 that Krylov and extended Krylov subspaces can provide effective spaces to look for solutions, but as we will see in Section 3.2, rational Krylov spaces are known to contain rich spectral information [65, 77]. This makes them an appealing approach to solve matrix equations. The idea of projecting onto a rational Krylov subspace is the subject of much recent research such as [21–23, 59, 71]. Rational subspace solvers are an attractive approach to solve matrix equations since, for a broad choice of poles [77], they have been shown to yield similar or better results than the equivalent ADI method [26]. In this chapter we provide new insight into the choice of poles. We note that, in practice, good pole choices for rational Krylov methods are often good pole choices for ADI as well, so our results may also prove useful for the latter method.

We start this chapter by deriving an explicit rational approximation for the solution of (2.9) for both 1- and 2-sided projections. The 2-sided result was also obtained by Druskin, Knizhnerman and Simoncini in [21] by means of the skeleton approximation. Obtaining the rational function explicitly allows for a better understanding of the rational approximation and, as a result, the best poles for the rational Krylov subspace. In the literature, there are two common approaches to choosing poles for rational Krylov subspaces: an a priori selection of the poles [71], and an adaptive method which computes the poles "on-the-fly" [23, 24]. Poles for rational Krylov methods for matrix functions have been treated in [41] and the so-called Zolotarev [95] poles are a popular choice in the literature [59, 71], but the ideal number of poles for rational Krylov methods has not previously been studied in detail for rational Krylov projections. We note, however, that the thesis by Sabino [74] presents a thorough assessment of the number of poles in the context of the ADI method. We also discuss the convergence of our approach by comparing it to the upper bound for Galerkin approximations from [4], and look at the attainable accuracy of our method.

In the second part of this chapter, we explore pole choices, such as the aforementioned Zolotarev ones and the IRKA poles [39], adapted from model order reduction of dynamical systems. The IRKA poles have been previously considered as poles for the ADI method [9, 31], and mentioned within rational Krylov apporaches in [93]. Despite not being considered competitive in these papers, we show in Section 3.6 that for a range of discretised diffusion problems, the IRKA poles are more efficient than other options. Furthermore, we compare the solver with a priori poles with the adaptive poles algorithm from [23], showing that the a priori computed IRKA poles can be as good, or better than the adaptive ones for a wide variety of diffusion problems.

3.2 Rational approximation

Recall, the Lyapunov equation is given in (2.9) by

$$AX + XA^T = F.$$

We solve this via a projection onto a rational Krylov subspace. For simplicity of exposition, the results presented throughout this section are for a symmetric rank-1 right-hand side given by bb^T, where $b \in \mathbb{R}^{n\times 1}$, but are readily generalisable to the right-hand side $F = CC^T$, with $\in \mathbb{R}^{n\times m}$, $m \ll n$. We will focus on both 1-sided and 2-sided Galerkin projection methods, with more emphasis on the latter. We first present the main steps of our projection method in very general terms.

3.2.1 Projection method

In this section we describe the 2-sided projection. Suppose we generate an orthonormal basis for an approximation space of dimension k in $\mathbb{R}^n$, with $k \ll n$. If we collect the k basis vectors into a matrix V of size $n \times k$, then we can seek an approximate solution to (2.9) of the form $X_k = VY_kV^T \approx X$. The matrix Y_k is found by insisting that the residual

$$R = AX + XA^T - bb^T \tag{3.1}$$

satisfies a Galerkin orthogonality condition given, for the 2-sided projection, by

$$V^T R V = 0. \tag{3.2}$$

Substituting X_k into R and applying (3.2) yields the reduced problem

$$A_k Y_k + Y_k A_k^T = b_k b_k^T, \tag{3.3}$$

where $A_k = V^T A V \in \mathbb{R}^{k\times k}$ and $b_k = V^T b \in \mathbb{R}^{k\times 1}$. We can solve this reduced problem either by transforming it into a linear system, using (2.4), and solving this, or by applying the built-in `MATLAB` function `lyap`, which is based on the Bartels–Stewart method [3] from Algorithm 3. In this thesis, the resulting reduced problem is small enough that we use the latter approach. Once we have obtained the solution of the reduced matrix equation, we compute the eigenvalue decomposition of Y_k and generate the low rank factor $\tilde{X}$ of the approximate solution $X_k = \tilde{X}\tilde{X}^T$. This method is encapsulated in Algorithm 4.

Algorithm 4

```
1: INPUT: $A, b, \mathcal{S} = \{s_1, s_2, \dots\}$, maximum iterations $maxit$.
2: Set $V_0 = b/\|b\|, j = 1$.
3: while not converged and $j < maxit$ do
4:     Expand basis using the poles: $V_j = getbasis(A, s_j, V_{j-1})$.
5:     Project the matrix $A_j = V_j^T A V_j$ and the right-hand side $b_j = V_j^T b$.
6:     Solve $A_j Y_j + Y_j A_j^T = b_j b_j^T$.
7:     Factorise $Y_j$ as $Y_j = W_j W_j^T$ and set $X_j = \tilde{X}\tilde{X}^T,$ with $\tilde{X} = V_j W_j$.
8:     Check convergence and if satisfied set $k = j$ and stop.
9: end while
10: OUTPUT: solution factor $\tilde{X},$ final residual.
```

In step 4 of Algorithm 4 we use the poles to expand the basis V_j. At each iteration of the solver, a new pole s_j is used to solve a shifted linear system $(A + s_j I)x = v_{j-1}$, where v_{j-1} is the previously generated column of V_j. The solution of each shifted linear system is then orthonormalised against all previously computed basis vectors and added on. We cycle through the poles, so that we are not limited by the number of a priori selected poles.

The algorithm stops when the stopping criterion is satisfied, i.e., when the relative residual is below a given tolerance:

$$\frac{\|R_j\|_F}{\|bb^T\|_F} < \tau, \tag{3.4}$$

where $R_j = AX_j + X_jA^T - bb^T$ is the residual at the jth iteration.

In step 6 of Algorithm 4, we apply the 2-sided projection—this means that we impose the orthogonality condition on both sides of the residual. The 2-sided approach is normally used for discretised PDEs [63] since the projected problem involves $k \times k$ matrices. The 1-sided approach was used in [71], where the left matrix, i.e., the matrix situated on the left-hand side of the unknown X, is of much larger dimension than the matrix on the right-hand side.

Although the 2-sided approach has a lower cost per iteration than the 1-sided approach for PDE problems, we are still interested in the performance of both approximations. As a first step, we will derive expressions for the rational approximation in each case.

3.2.2 1-sided approach

In this section, we describe the 1-sided projection approach. Given a basis V, we seek an approximate solution to (2.9) of the form $X_k = VY_k \approx X$. Using this approximation, the matrix Y_k is found by insisting that the residual R in (3.1) satisfies the orthogonality condition,

$$V^T R = 0. \tag{3.5}$$

Substituting X_k into R and applying (3.5), now yields the reduced problem

$$A_k Y_k + Y_k A^T = b_k b^T, \tag{3.6}$$

where $A_k = V^T A V$ and $b_k = V^T b$, as before. Since we only applied the projection on the left, the right matrix remains unchanged (size $n \times n$), while the left matrix has reduced in size (from $n \times n$ to $k \times k$, $k \ll n$).

We continue our discussion about the 1-sided approach by deriving an explicit expression for the rational function generated when the rational Krylov method is used to solve the Lyapunov equation $AX + XA^T = bb^T$ from (2.9) via a 1-sided projection. We start by noting that the matrix A from (2.9) is SPD. To simplify the exposition, we will first transform the Lyapunov equation to one with diagonal coefficient matrices, by using an eigenvalue decomposition of A. We can write the eigenvalue decomposition of A as $A = Z\Lambda Z^T$, where Z is the square $n \times n$ matrix whose ith column is the eigenvector z_i of A, and Λ is the diagonal matrix whose ith diagonal element is the corresponding eigenvalue $\Lambda_{ii} = \lambda_i$. Replacing A by its eigenvalue decomposition in (2.9) yields

$$\Lambda \hat{X} + \hat{X}\Lambda = \hat{b}\hat{b}^T, \tag{3.7}$$

where $\Lambda = \text{diag}(\lambda_1, \ldots, \lambda_n)$, $\hat{X} = Z^T X Z$ and $\hat{b} = Z^T b$. To understand the rational approximation, consider the jth column of (3.7):

$$\begin{aligned} (\Lambda + \lambda_j I)\hat{x}_j &= \beta_j \hat{b}, \\ \text{i.e.,} \quad \hat{x}_j &= \beta_j (\Lambda + \lambda_j I)^{-1} \hat{b}, \end{aligned} \tag{3.8}$$

where β_j is the jth entry of the vector $\hat{b}^T$.

Let $V \in \mathbb{R}^{n \times k}$ have as its columns an orthonormal basis of dimension k for an approximation space in $\mathbb{R}^n$, with $k \ll n$. We want to apply an orthogonality condition on our residual. Accordingly, we let $\bar{x}_j = V y_j$ be an approximation to $\hat{x}_j$ such that

$$(\Lambda + \lambda_j I)\bar{x}_j - \beta_j \hat{b} \perp V.$$

This is equivalent to

$$V^T((\Lambda + \lambda_j I)V y_j - \beta_j \hat{b}) = 0,$$

which is clearly a 1-sided projection for (3.7).

We wish to obtain a rational approximation for the solution of (3.7). To do this, we can use the rational FOM [72] (or the rational Arnoldi approximation) to create an orthonormal basis of the rational Krylov subspace. When building our basis V, we let $v_0 = \hat{b}/\|\hat{b}\|_2$ be the first column. Then $\hat{b} = \|\hat{b}\|_2 v_0 = \|\hat{b}\|_2 V e_1$, where e_1 is the first column of the identity matrix. Note, we also have $V^T V = I_k$, the identity matrix of dimension k. Then, using (3.8), at the kth step, we have the jth column of the projected residual given by

$$\begin{aligned} &V^T((\Lambda + \lambda_j I)V y_j - \beta_j \|\hat{b}\|_2 V e_1) = 0, \\ &\text{i.e.,} \quad y_j = \beta_j \|\hat{b}\|_2 (\Lambda_k + \lambda_j I_k)^{-1} e_1 \end{aligned}$$

where $\Lambda_k = V^T \Lambda V$. This then gives the approximate solution to the jth column of $\hat{X}$, $\bar{x}_j = V y_j \approx \hat{x}_j$:

$$\bar{x}_j = \beta_j \|\hat{b}\|_2 V(\Lambda_k + \lambda_j I_k)^{-1} e_1.$$

The rational approximation to the Lyapunov equation (3.7) satisfies [40, Theorem 5.8]:

$$\bar{x}_j^{(k)} = \beta_j \frac{p_{k-1}}{q_{k-1}}(\Lambda)\hat{b} = \beta_j \hat{r}_k^{[\lambda_j]}(\Lambda)\hat{b},$$

where p_{k-1} and q_{k-1} are polynomials of degree at most $k-1$, i.e., $q_{k-1}, p_{k-1} \in \mathcal{P}_{k-1}$ and $\hat{r}_k^{[\lambda_j]}(z)$ must satisfy the following properties of the rational FOM approximation:

- It is in $\mathcal{Q}_{k-1,k-1}$, the set of rational functions with both numerator and denominator of degree $k-1$.
- It interpolates the function $(z+\lambda_j)^{-1}$ at the Ritz values, $\sigma(\Lambda_k) = \{\rho_1, \cdots \rho_k\}$.
- The denominator $q_{k-1}(z)$ can be factored as $q_{k-1}(z) = (z-s_1)\cdots(z-s_{k-1})$, where the s_i's are the poles used to generate the basis V.

To study the effect of the choice of poles on the quality of the rational Krylov approximation, it is helpful to have an explicit representation of $\hat{r}_k^{[\lambda_j]}(z)$. Let $t_k \in \mathcal{Q}_{k,k-1}$, with roots $\sigma(\Lambda_k)$ and denominator $q_{k-1}(z)$, i.e., $t_k(z) = \frac{\det(zI-\Lambda_k)}{q_{k-1}(z)}$. Then, we postulate that

$$\hat{r}_k^{[\lambda_j]}(z) = \frac{1 - \frac{t_k(z)}{t_k(-\lambda_j)}}{z+\lambda_j}. \tag{3.9}$$

We will demonstrate that $\hat{r}_k^{[\lambda_j]}(z)$ from (3.9) is indeed the rational FOM approximation. In order to do this, we will consider each of the three properties from above and prove that $\hat{r}_k^{[\lambda_j]}(z)$ satisfies them.

We begin by showing that $\hat{r}_k^{[\lambda_j]}(z) \in \mathcal{Q}_{k-1,k-1}$. Look at the numerator of (3.9) and consider its roots. The expression $1 - \frac{t_k(z)}{t_k(-\lambda_j)}$ has k roots since $\det(zI - \Lambda_k)$ is a degree k polynomial in z. Clearly these k roots satisfy $\frac{t_k(z)}{t_k(-\lambda_j)} = 1$. One of the values of z that satisfies $t_k(z) = t_k(-\lambda_j)$ is $z = -\lambda_j$, which means that $z = -\lambda_j$ is a root of $1 - \frac{t_k(z)}{t_k(-\lambda_j)}$, i.e.,

$$\begin{aligned} 1 - \frac{t_k(z)}{t_k(-\lambda_j)} &= \frac{t_k(-\lambda_j) - t_k(z)}{t_k(-\lambda_j)} \\ &= \frac{t_k(-\lambda_j)q_{k-1}(z) - \det(zI - \Lambda_k)}{t_k(-\lambda_j)q_{k-1}(z)} \\ &= \frac{p_k(z)}{t_k(-\lambda_j)q_{k-1}(z)} = \frac{(z+\lambda_j)p_{k-1}(z)}{t_k(-\lambda_j)q_{k-1}(z)}, \end{aligned} \tag{3.10}$$

so that $\hat{r}_k^{[\lambda_j]}(z) = \frac{p_{k-1}(z)}{t_k(-\lambda_j)q_{k-1}(z)} \in \mathcal{Q}_{k-1,k-1}$, which shows the first property.

Next, we want to check that $\hat{r}_k^{[\lambda_j]}(z)$ interpolates $(z + \lambda_j)^{-1}$. This is very straightforward to do. Consider $\hat{r}_k^{[\lambda_j]}(z)$ evaluated at one of the Ritz values, say ρ_ℓ:

$$\hat{r}_k^{[\lambda_j]}(\rho_\ell) = \frac{1 - \frac{t_k(\rho_\ell)}{t_k(-\lambda_j)}}{\rho_\ell + \lambda_j}.$$

Note that $t_k(\rho_\ell) = \frac{\det(\rho_\ell I - \Lambda_k)}{q_{k-1}(\rho_\ell)} = 0$ since ρ_ℓ is a Ritz value. Hence $\hat{r}_k^{[\lambda_j]}(\rho_\ell) = (\rho_\ell + \lambda_j)^{-1}$, i.e., $\hat{r}_k^{[\lambda_j]}(z)$ interpolates $(z+\lambda_j)^{-1}$ at the eigenvalues of Λ_k.

Lastly, we show the third property of the FOM approximation, i.e., that the denominator $q_{k-1}(z) = (z - s_1)\cdots(z - s_{k-1})$. This is easily seen from (3.10) since by construction, we have created q_{k-1} to have as factors the poles used to generate the basis V.

As we now have shown that $\hat{r}_k^{[\lambda_j]}(z)$ in (3.9) is the rational FOM approximation of $\bar{x}_j$, we can evaluate the error. Recall that the error is given by

$e_j = \hat{x}_j - \bar{x}_j$, i.e.,

$$\begin{aligned} e_j &= \beta_j(\Lambda + \lambda_j I)^{-1}\hat{b} - \beta_j(\Lambda + \lambda_j I)^{-1}\Big(I - \frac{t_k(\Lambda)}{t_k(-\lambda_j)}\Big)\hat{b} \\ &= \beta_j(\Lambda + \lambda_j I)^{-1}\frac{t_k(\Lambda)}{t_k(-\lambda_j)}\hat{b}. \end{aligned}$$

Note that this error is small when $\frac{t_k(\Lambda)}{t_k(-\lambda_j)}$ is small. Thus, we seek poles $s_1, \ldots, s_k$ that minimise $t_k(\Lambda)$, i.e.,

$$\min_{s_i \in \mathcal{S}} \max_{z \in \sigma(\Lambda)} \left| \frac{t_k(z)}{t_k(-\lambda_j)} \right|.$$

To make the minimisation problem easier to deal with, it is common to replace it by

$$\min_{s_i \in \mathcal{S}} \max_{z \in [\lambda_{\min}, \lambda_{\max}]} \left| \frac{t_k(z)}{t_k(-\lambda_j)} \right|, \tag{3.11}$$

which is the third Zolotarev problem [22, 47, 71]. Note that the small, projected matrix equation (3.6) depends on the s_i through their role as poles of t_k. The negative Ritz values $-\rho_j$ are necessarily distinct from these poles.

Transforming back to the original variables, we find that at step k, the jth column of the rational approximation is given by $x_j^{(k)} = \beta_j \hat{r}_k^{[\lambda_j]}(A)b$.

3.2.3 2-sided approach

Next, we consider a 2-sided projection for the Lyapunov equation in (2.9), $AX + XA^T = bb^T$, as we will show in Section 3.2.4 that this is a more efficient approach. In order to obtain the rational function that is implicitly constructed by the 2-sided rational approximation, we will use a number of transformations between the Kronecker form of the problem and the matrix equation form. These useful properties were presented in Section 2.1.

Using the 2-sided projection method, as described in Algorithm 4, we now explicitly derive the rational approximation to the solution. Combining (3.2) with (2.4) gives the corresponding Kronecker product form

$$(V^T \otimes V^T)\,\mathrm{vec}(R) = 0. \tag{3.12}$$

Next, before applying the Galerkin orthogonality condition from (3.12), we transform R into the corresponding vector form using the Kronecker product,

$$\mathrm{vec}(R) = \mathrm{vec}(bb^T) - (A \otimes I + I \otimes A)\,\mathrm{vec}(X_k).$$

Hence, the 2-sided orthogonality condition becomes

$$(V^T \otimes V^T)\,\mathrm{vec}(R) = (V^T \otimes V^T)(\mathrm{vec}(bb^T) - (A \otimes I + I \otimes A)\,\mathrm{vec}(X_k)) = 0,$$

so that

$$(V^T \otimes V^T)(A \otimes I + I \otimes A)\,\mathrm{vec}(X_k) = (V^T \otimes V^T)\,\mathrm{vec}(bb^T). \tag{3.13}$$

Recall that $\mathrm{vec}(X_k) = \mathrm{vec}(VY_kV^T) = (V \otimes V)\,\mathrm{vec}(Y_k)$ and since $V^TV = I$, we can write $\mathrm{vec}(Y_k) = \mathrm{vec}(V^TX_kV) = (V^T \otimes V^T)\,\mathrm{vec}(X_k)$. So, the left-hand side of (3.13) becomes

$$\begin{aligned}(V^T \otimes V^T)(A \otimes I + I \otimes A)\,\mathrm{vec}(X_k) &= (V^T \otimes V^T)(A \otimes I + I \otimes A)(V \otimes V)\,\mathrm{vec}(Y_k)\\ &= (V^T \otimes V^T)(A \otimes I + I \otimes A)(V \otimes V)(V^T \otimes V^T)\,\mathrm{vec}(X_k)\\ &= (A_k \otimes I + I \otimes A_k)(V^T \otimes V^T)\,\mathrm{vec}(X_k),\end{aligned}$$

where $A_k = V^TAV$. Hence, we obtain

$$\mathrm{vec}(X_k) = (V \otimes V)(A_k \otimes I + I \otimes A_k)^{-1}(V^T \otimes V^T)\,\mathrm{vec}(bb^T). \tag{3.14}$$

This represents the approximate vectorised solution to the Lyapunov equation from (2.9). It makes use of the basis V and of the Kronecker product.

Before we continue our derivation of the rational approximation, we require a useful result coming from the field of approximation of matrix functions, which was proved in [40, Lemma 3.9].

Lemma 3.2.1. *Suppose the columns of V form a basis for $\mathcal{RK}_k(A, b, \mathcal{S})$, and define $q = q_{k-1}(A)^{-1}b$, where $q_{k-1} \in \mathcal{P}_{k-1}$ is a polynomial of degree at most $k-1$, whose roots are the poles $\mathcal{S}$. Then, the following holds*

$$p_{k-1}(A)q = Vp_{k-1}(A_k)V^Tq,$$

for any polynomial $p_{k-1} \in \mathcal{P}_{k-1}$.

We will use this together with (3.14) to obtain a new rational approximation, as presented in our submitted paper [86].

Let $f(A_k \otimes I, I \otimes A_k) = (A_k \otimes I + I \otimes A_k)^{-1}$, $q = q_{k-1}(A)^{-1}b$ and $p_{k-1}(A)q = Vp_{k-1}(A_k)V^Tq$, where p_{k-1} and q_{k-1} are polynomials of degree at most $k-1$, i.e., $q_{k-1}, p_{k-1} \in \mathcal{P}_{k-1}$. We first look at $\operatorname{vec}(bb^T)$ in (3.14):

$$\begin{aligned}\operatorname{vec}(bb^T) &= \operatorname{vec}((q_{k-1}(A)q)(q_{k-1}(A)q)^T)\\ &= \operatorname{vec}(Vq_{k-1}(A_k)V^Tqq^TVq_{k-1}(A_k)V^T)\\ &= (V \otimes V)q_{k-1}(A_k \otimes I)q_{k-1}(I \otimes A_k)(V^T \otimes V^T)\operatorname{vec}(qq^T),\end{aligned} \tag{3.15}$$

using (2.7). Next, we substitute (3.15) into (3.14) to obtain

$$\operatorname{vec}(X_k) = (V \otimes V)P_k(A_k \otimes I, I \otimes A_k)(V^T \otimes V^T)\operatorname{vec}(qq^T), \tag{3.16}$$

where $P_k(A_k \otimes I, I \otimes A_k) = f(A_k \otimes I, I \otimes A_k)q_{k-1}(A_k \otimes I)q_{k-1}(I \otimes A_k)$. We begin to simplify P_k by noting that $f(A_k \otimes I, I \otimes A_k)$ can be written as a polynomial of degree at most $k-1$. We can write any polynomial as $c(A_k)q(A_k) + r(A_k)$, using the remainder theorem, where $c(A_k)$ is the minimum polynomial of A_k, with $c(A_k) = 0$ and $q(A_k)$, $r(A_k)$ are polynomials of smaller degrees. Note also that $A_k \otimes I$ commutes with $I \otimes A_k$. Then, we have

$$\begin{aligned}
f(A_k \otimes I, I \otimes A_k) &= \sum_{i=0}^{k^2-1} \alpha_i (A_k \otimes I + I \otimes A_k)^i \\
&= \sum_{i=0}^{k^2-1} \alpha_i \sum_{j=0}^{i} \binom{i}{j} A_k^{i-j} \otimes A_k^j, \quad i > j \\
&= \sum_{i=0}^{k^2-1} \sum_{j=0}^{i} \beta_{ij} A_k^{i-j} \otimes \sum_{\ell=0}^{k-1} \gamma_\ell A_k^\ell \\
&= \sum_{i=0}^{k^2-1} \sum_{j=0}^{i} \beta_{ij} \sum_{\ell=0}^{k-1} \gamma_\ell A_k^{i-j} \otimes A_k^\ell \\
&= \sum_{s=0}^{k-1} \delta_s [c(A_k)q(A_k) + r(A_k)] \otimes A_k^s \\
&= \sum_{s,v=0}^{k-1} a_{sv} A_k^v \otimes A_k^s,
\end{aligned} \tag{3.17}$$

where $\alpha, \beta, \gamma, \delta$ are constant coefficients of the polynomials in A_k. Using $q = q_{k-1}(A)^{-1}b$ from Lemma 3.2.1 and (2.7), we have that (3.16) is

$$\begin{aligned}
\operatorname{vec}(X_k) &= P_k(A \otimes I, I \otimes A) q_{k-1}(A \otimes I)^{-1} q_{k-1}(I \otimes A)^{-1} \operatorname{vec}(bb^T) \\
&= r_{k-1}(A \otimes I, I \otimes A) \operatorname{vec}(bb^T),
\end{aligned} \tag{3.18}$$

where $r_{k-1}(x, y) = \frac{P_k(x,y)}{q_{k-1}(x)q_{k-1}(y)}$. Note that this is the same rational approximation obtained in Remark 3.5 of [21], by means of skeleton approximation, but here instead we directly utilise properties of rational Krylov subspaces. Hence, $P_k(x, y)$ interpolates $f(x, y)q_{k-1}(x)q_{k-1}(y)$ at the Ritz values $\sigma(A_k)$ and so, $r_{k-1}(x, y)$ interpolates $f(x, y)$ at the Ritz values $\sigma(A_k)$.

As in the 1-sided case, we can see from the previous discussion that the rational approximation $r_{k-1}(x, y)$ in (3.18) satisfies some specific properties. In this case:

- It is in $\mathcal{Q}_{k-1,k-1}$, the set of rational functions with both numerator and denominator of degree $k-1$.
- It interpolates the function $(x+y)^{-1}$ at the Ritz values, $\sigma(A_k) = \{\rho_1, \ldots, \rho_k\}$.
- The denominator $q_{k-1}(x)q_{k-1}(y)$ can be factored as $q_{k-1}(x)q_{k-1}(y) = (x + s_1)\cdots(x+s_{k-1})(y+s_1)\cdots(y+s_{k-1})$, where the s_i's are the poles used to generate the basis V.

When $s_i = \rho_i$, $i = 1, \ldots, k$, the rational function takes the form [61]:

$$r_{k-1}(x, y) = \frac{1}{x+y}\Big(1 - \prod_{i=1}^{k-1} \frac{(x-\rho_i)(y-\rho_i)}{(x+\rho_i)(y+\rho_i)}\Big). \tag{3.19}$$

Having obtained the rational approximation, we are now interested in determining suitable poles for the rational Krylov subspace. We begin by minimising the largest error. This is given by $\mathrm{vec}(E(A \otimes I, I \otimes A)) = \mathrm{vec}(X) - \mathrm{vec}(X_k)$, where

$$\begin{aligned} E(x,y) &= (x+y)^{-1} - (x+y)^{-1}\Big[1 - \prod_{i=1}^{k-1} \frac{(x-\rho_i)(y-\rho_i)}{(x+\rho_i)(y+\rho_i)}\Big] \mathrm{vec}(bb^T) \\ &= (x+y)^{-1} \prod_{i=1}^{k-1} \frac{(x-\rho_i)(y-\rho_i)}{(x+\rho_i)(y+\rho_i)} \mathrm{vec}(bb^T), \end{aligned}$$

where x and y are $A \otimes I$ and $I \otimes A$, respectively, with $\frac{1}{x}$ to be interpreted as the inverse x^{-1}. The error is small when $\|\prod_{i=1}^{k-1} \frac{(x-\rho_i)(y-\rho_i)}{(x+\rho_i)(y+\rho_i)}\|_2$ is small. Since we want to minimise the largest error, note that $\|\prod_{i=1}^{k-1} \frac{(x-\rho_i)(y-\rho_i)}{(x+\rho_i)(y+\rho_i)}\|_2$ is maximised over the spectral interval when $x = y$ [61]. Therefore, we seek the poles $s_1, \dots, s_{k-1}$ that minimise this largest error, and so, we must solve the following minimax problem

$$\min_{s_i \in \mathcal{S}} \max_{x \in \sigma(A)} \left| \prod_{i=1}^{k-1} \frac{(x - s_i)^2}{(x + s_i)^2} \right|. \tag{3.20}$$

Relaxing this so that we maximise the error over the spectral interval, $[\rho_{\min}, \rho_{\max}]$ yields the following minimax problem

$$\min_{s_i \in \mathcal{S}} \max_{x \in [\rho_{\min}, \rho_{\max}]} \left| \prod_{i=1}^{k-1} \frac{(x - s_i)^2}{(x + s_i)^2} \right|. \tag{3.21}$$

This corresponds to the third Zolotarev problem [47, 71], the solution of which yields the poles used to generate the basis V.

3.2.4 Comparison

In Section 3.2.2 and Section 3.2.3 we presented the two ways in which we can project the Lyapunov equation $AX + XA^T = bb^T$ from (2.9). In this section we compare the two projection approaches. We apply both methods to a diffusion problem of the form (2.8), discretised as in Section 2.2.1. We solve the Lyapunov equation of size $n = 1000$ using a projection into the rational Krylov subspace, and use 12 Zolotarev poles. The right-hand side is given as $b = [1, \dots, 1]^T$, i.e., $F = bb^T$ is a full matrix of all ones. We want to see if there is any difference in convergence by considering the number of iterations and the computational time.

From Figure 3.1, we see that the two convergence curves follow a similar trajectory and that both methods reach the desired residual norm reduction of 10^{-8} in the same number of iterations. This is perhaps unsurprising given that the properties of the 2-sided and 1-sided rational approximations are very similar.

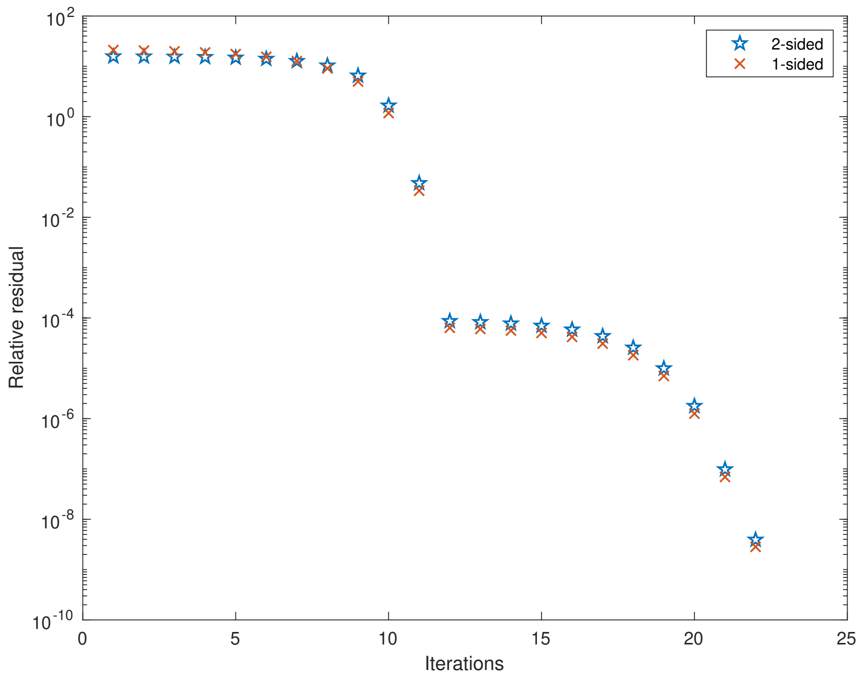


Figure 3.1: Comparison of 1-sided and 2-sided projections for the rational Krylov subspace approach.

However, these projections are not the same. In particular, the projected (reduced size) problem is different. As part of the 1-sided approach, we need to solve a projected problem where one of the coefficient matrices has dimension $k \times k$, $k \ll n$, and one has size $n \times n$. This implies that the 1-sided approach requires more work to solve this projected matrix equation. On the other hand, the projected problem in the 2-sided approach involves only $k \times k$ matrices. Therefore, despite the number of iterations and convergence curves being the same, the computational time of the two approaches differs significantly, with the 2-sided approximation being approximately 5 times quicker than the 1-sided approach. Therefore, in the rest of this thesis, we use the 2-sided approximation for our numerical results. However, we note that the 1-sided approach might be more appropriate for Sylvester equations where the discretisation leads to coefficient matrices of different dimensions.

3.3 Attainable accuracy

In this section we consider the attainable accuracy of Algorithm 4. Studying this for our 2-sided method will allow us to determine an appropriate criterion to stop the iterative process. After each iteration, we check the Frobenius norm of the relative residual, i.e.,

$$\frac{\|R_k\|_F}{\|F\|_F} < \tau,$$

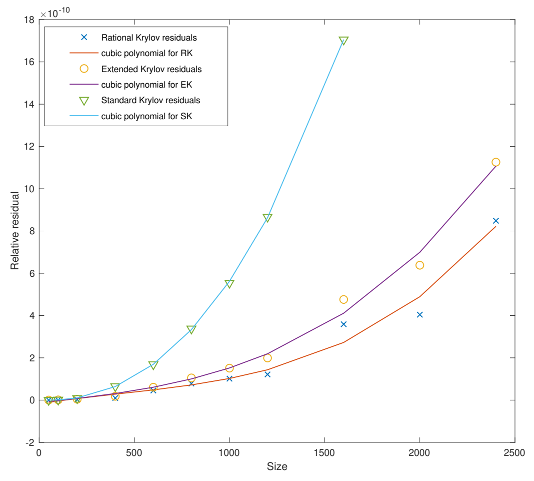
where $R_k = AX_k + X_k A^T - F$ is the residual at the kth iteration, and τ is a chosen tolerance. In order to determine the best stopping criterion for our method, we will explore a very strict stopping tolerance and observe what happens to the norm of the residual. The analysis in [43] shows that, for Sylvester equations, “the relative residual is guaranteed to be bounded by a modest multiple of the unit round-off” when the Bartels–Stewart algorithm is used, and indicates a dependence in terms of the problem dimensions on this ‘modest multiple’. In the context of linear systems, this ‘modest multiple’ is expressed as a low degree polynomial in n (when the right-hand side is of dimensions $n \times 1$). It is reasonable to infer from the analysis that this also applies for the Sylvester equation in [43]. We see in the following discussion, that this also appears to be the case for our Krylov algorithm for solving Lyapunov equations.

As mentioned in Section 2.4, we can use any of the three Krylov subspaces to generate the basis used in our 2-sided projection: standard, extended and rational. We compare the attainable accuracy of our algorithm for all three. In order to generate the rational Krylov subspace, we compute 8 Zolotarev poles and cycle through them. To determine the attainable accuracy results presented below, we have run Algorithm 4 in `MATLAB` with the tolerance on the residual set to $\tau = 10^{-14}$ and the maximum number of possible iterations set to 100 for the rational Krylov basis, 200 for the extended Krylov basis, and 1000 for the standard Krylov basis.

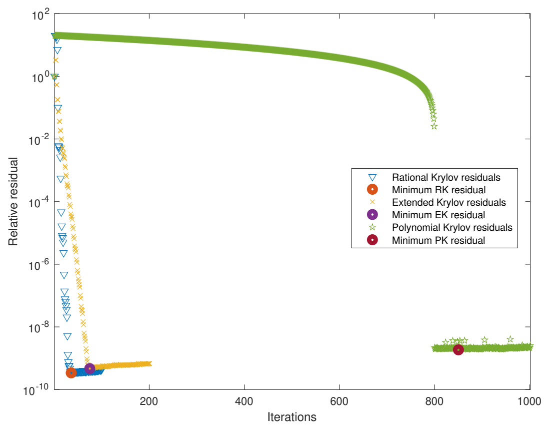
In Figure 3.2(a) we have plotted the minimum relative norm of the residual that our method can reach for each of the three Krylov subspaces described, depending on the size of the matrices used in the Lyapunov equation, on a linear scale, and fitted cubic polynomials through this data in order to determine the reliability of our method. Note that we had previously fitted a quadratic through this data but this was not as descriptive as the cubic. We can see that the size of the problem has an impact on how small the residual can get. The fact that we lose some accuracy for large problems is inevitable due to rounding errors, but acceptable since the linear scale graph presents low-degree polynomial behaviour, which implies that our method will converge to reasonably accurate solutions. This means that we do not expect to lose too much accuracy if we increase the size of our Lyapunov equation. Furthermore, we can see that the rational Krylov basis with 8 Zolotarev poles attains the smallest residual norm out of the three bases tested for large problems.

Next, we have considered each of the three bases in order to study what happens before and after the minimum residual is reached in Figure 3.2(a). In Figure 3.2(b) we present the residual obtained by running Algorithm 4 for each of our three bases for a larger size of the Lyapunov equation, $n = 1600$. Note that on our plot, the circles correspond to the minimum relative norm of the residual attained by each of the bases. For each of the methods, this minimum residual is also presented in Figure 3.2(a).

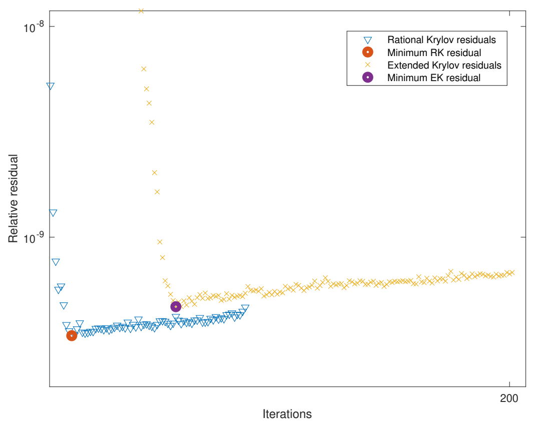
Looking at Figure 3.2(b), we see that there are two main phases that appear in the iterative process. First, we notice a phase of convergence, i.e., a phase



(a) Minimum residuals.



(b) Comparison between the three bases.



(c) Oscillations due to noise for rational and extended Krylov residuals.

Figure 3.2: Attainable accuracy plots.

where the residual decreases until it reaches a "certain value". Upon reaching this minimum attainable accuracy, it enters the second phase. This phase does not hold useful information about the convergence of the method and basis we use, as these oscillations, seen in Figure 3.2(c) for rational and extended Krylov residuals, represent noise appearing because of rounding errors. It is the first phase that we are interested in as this presents behaviour that is specific to the Krylov subspace used. We can see clearly that the minimum norm of the residual is usually reached after some stagnation for the polynomial basis, and also that it requires the largest number of iterations to reach this minimum residual. This is followed by the extended Krylov subspace, which also requires more iterations to reach the minimum norm of the residual than the rational Krylov subspace, showing the superiority of the rational Krylov subspace for our problem with regards to how quickly the minimum residual is reached.

The attainable accuracy analysis presented in this section suggests that in practice, it is inappropriate to consider such a strict tolerance on the residual norm, as rounding errors prevent this from being attained. Of course, the tolerance is chosen by the user, who decides the acceptable level of error in the computed solution. When a very small residual error is required, we have seen that the rational Krylov approach performs best. However, in practical situations, where less strict tolerances might be used, the rational and extended bases prove to be robust. The decision on which Krylov subspace to use should also be guided by other properties of the approaches, such as speed.

3.4 Upper bounds

In this section we consider two bounds to see if they are descriptive of actual convergence rates for diffusion problems. One is an upper bound derived in [4, Corollary 2.5] for the residual of a 2-sided rational projection. The other one [21, Theorem 4.9] is asymptotic and holds for $k \to \infty$. We are interested to see if this is also descriptive for small k.

3.4.1 Beckermann bound

For the case of the Lyapunov equation, the bound in [4, Corollary 2.5], which we call the Beckermann bound, depends only on the condition number of matrix A and the choice of poles used. Letting $\mathcal{S} = \{s_1, \dots, s_k\}$ be the poles we use and $\lambda_{\min}, \lambda_{\max}$ be the smallest and largest eigenvalues of matrix A, respectively, and using the notation in [4, Corollary 2.5], the bound is given by

$$\frac{\|R_k\|_F}{\|b\|_F^2} \leq (4 + 4\sqrt{2\kappa(A)})\gamma, \tag{3.22}$$

where $R_k = AX_k + X_kA^T - bb^T$ is the residual at the kth iteration, $\kappa(A) = \frac{\lambda_{\max}}{\lambda_{\min}}$ is the condition number of A, and $\gamma = \max_{z \in [\lambda_{\min}, \lambda_{\max}]} u_{A,k}(z)$, with

$$u_{A,k}(z) = \prod_{j=1}^{k} \left| \frac{\sqrt{\frac{z-\lambda_{\max}}{z-\lambda_{\min}}}\sqrt{\frac{s_j-\lambda_{\min}}{s_j-\lambda_{\max}}} - 1}{\sqrt{\frac{z-\lambda_{\max}}{z-\lambda_{\min}}}\sqrt{\frac{\bar{s}_j-\lambda_{\min}}{\bar{s}_j-\lambda_{\max}}} + 1} \right|.$$

We have computed this bound for a choice of 8 Zolotarev poles, as this showcases convergence curves that make the graph easier to understand. Since the bound does not depend on the right-hand side, we are interested to see how descriptive it is for our problem and to understand whether it can guide our choice of poles.

In this test we have considered three different vectors b: a vector of ones, a vector of alternating 1s and -1s, and a vector of normally distributed random numbers. Our results are presented in Figure 3.3.

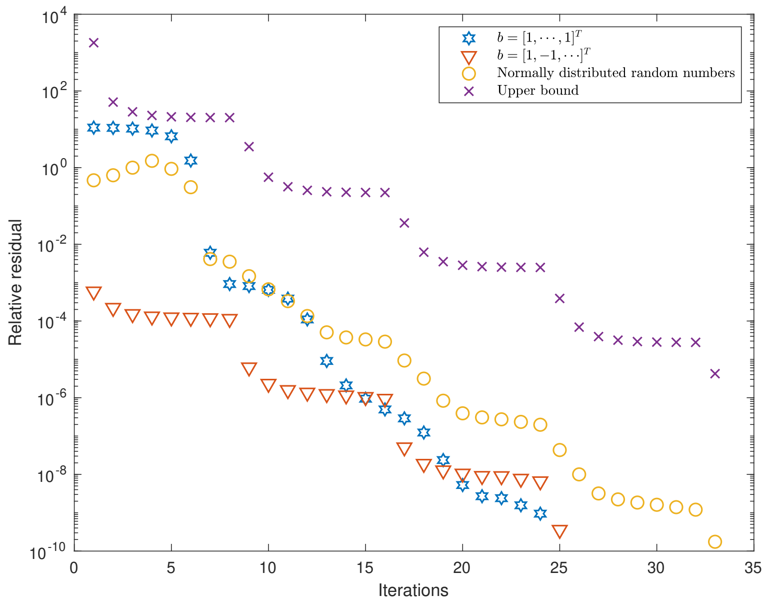


Figure 3.3: Upper bound from (3.22) for different choices of right-hand side vector b.

We notice from this figure that the bound is descriptive of our solver convergence for most choices of right-hand side b, with an almost identical qualitative behaviour to the residual in the case of the alternating 1s and -1s.

3.4.2 Asymptotic error bound

Let $W = [d, 1]$, with $0 < d < 1$, be the field of values of the self-adjoint matrix A. The asymptotic error estimate for the rational Krylov error from [21, Theorem 4.9] is given by

$$\overline{\lim_{k\to\infty}} \|X - X_k\|^{\frac{1}{k}} \leq \exp\Big(-\frac{\pi K(c)}{2K'(c)}\Big),$$

where K and K' are the principal and complimentary elliptic integrals of modulus c, with

$$c = \sqrt{\frac{(\delta-\alpha)(\gamma-\beta)}{(\delta-\beta)(\gamma-\alpha)}},$$

$$\alpha = -\frac{3+d}{1-d} - \sqrt{\left(\frac{3+d}{1-d}\right)^2 - 1}, \quad \delta = \frac{1}{\alpha}$$

$$\beta = -\frac{1+3d}{1-d} - \sqrt{\left(\frac{1+3d}{1-d}\right)^2 - 1}, \quad \gamma = \frac{1}{\beta}.$$

In Figure 3.4 we have plotted this asymptotic bound on the same figure as the actual error and the convergence curves. The setup of the solver is given by 8 Zolotarev poles and right-hand side of ones for size $n = 1000$.

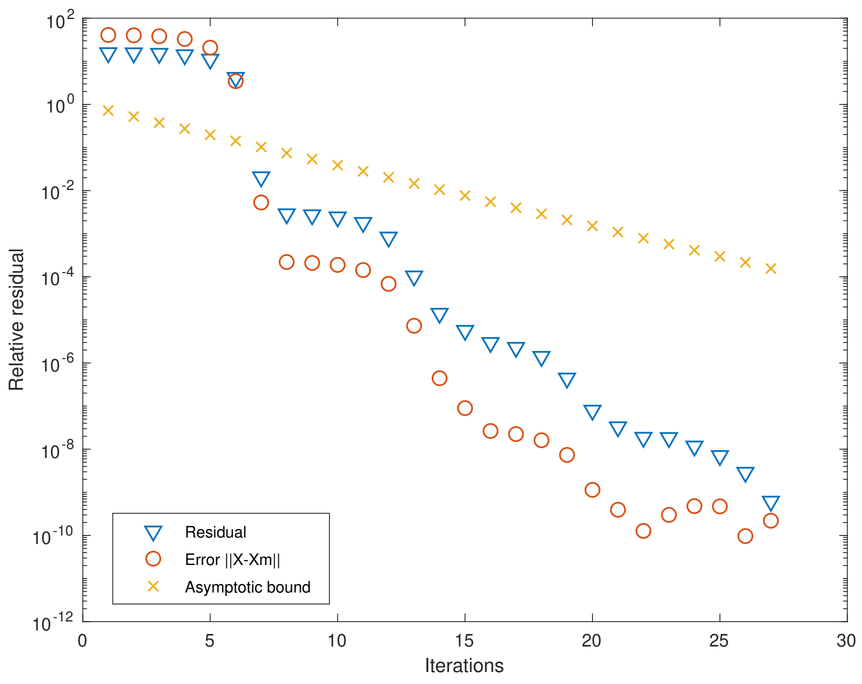


Figure 3.4: Asymptotic upper bound from [21, Theorem 4.9] alongside the error and residual for Lyapunov equation of size $n = 1000$.

From Figure 3.4, we notice that the residual and error convergence curves show similar trajectories, but the asymptotic bound does not follow this. We believe this is because of the asymptotic nature of the bound, and the lack of influence from the poles or right-hand side choice may also contribute.

3.5 Pole choices

In this section we discuss approaches for generating a priori and adaptive poles for the rational Krylov projection method. Later in the chapter we compare these for a number of diffusion problems.

3.5.1 Zolotarev poles

We claimed in Section 3.2.3 that relaxing equation (3.20) leads to the third Zolotarev problem. To describe this problem, let $E = \{x \in \mathbb{R}, |x| \leq 1\}$ and $F = \{x \in \mathbb{R}, |x| \geq 1/m\}$, $m < 1$ and $\mathcal{Q}_{n,n}$ be the collection of irreducible ordinary rational functions whose numerator and denominator are polynomials with degree at most n. Then the third Zolotarev problem is given by the following [47, 95].

Problem. *Find $r_n^* \in \mathcal{Q}_{n,n}$ that achieves the minimum*

$$\sigma_n = \min_{r_n \in \mathcal{Q}_{n,n}} \max_{z \in E} |r_n(z)|$$

among all rational functions of degree n subject to

$$\min_{z \in F} |r_n(z)| = 1.$$

The denominator of the solution yields poles that we can use to generate the rational basis V. In practice, we compute the Zolotarev poles by the approach implemented by Sabino [74], which uses elliptic integrals, but we note that there has been a lot of research into the Zolotarev problem, both with real and complex E and F [91]. In his thesis, Sabino also mentions that the Zolotarev poles can be well approximated by logarithmically spaced values in the spectral interval, and we will also show results using these logarithmic poles.

3.5.2 IRKA poles

The pole choices discussed so far overlook the influence of the right-hand side in the convergence of the Lyapunov solver. This can be addressed by using the iterative rational Krylov algorithm (IRKA) [6, 7, 39], a well established approach in the context of model order reduction of dynamical systems of the form

$$\begin{aligned} E\dot{\boldsymbol{x}}(t) &= A\boldsymbol{x}(t) + C\boldsymbol{u}(t), \\ \boldsymbol{y}(t) &= C\boldsymbol{x}(t), \end{aligned}$$

where $\boldsymbol{x}(t), \boldsymbol{u}(t)$, and $\boldsymbol{y}(t)$ are the state, control, and output of the system, respectively. The stability properties of this dynamical system can be characterised by a Lyapunov equation of the form $AXE^T + EXA^T = CC^T$ [7]. Note that in this thesis, we only consider Lyapunov equations where $E = I$. For more details about dynamical systems and model order reduction, see [39].

In model order reduction, IRKA is used to produce reduced order models that satisfy some interpolation-based first-order conditions. As part of this procedure, a number of poles are used to construct the reduced order model. By relating dynamical systems to corresponding matrix equations, we can use the IRKA method to generate a set of poles and then use them to solve (2.9) by a rational Krylov method, similarly to [9, 93] for ADI. The algorithm for generating the poles is presented in Algorithm 5 and has been adapted from [7, Alg. 1]. The algorithm presented uses a block right-hand side C and requires a set of tangential directions. The tangential directions are obtained from the SVD of C and then updated as described in [7, Sec. 3.1]. Note that when we work with a rank-1 right-hand side, tangential directions are superfluous.

Algorithm 5 IRKA poles for Lyapunov equation

```
1: INPUT: coefficient matrix $A$, right-hand side $C$, initial tangential directions
   $\hat{C} = [\hat{c}_1, \ldots, \hat{c}_k]$, initial poles $\mathcal{S} = \{s_1, \ldots, s_k\}$, tolerance $tol$.
2: **while** relative change in $s_i > tol$ **do**
3:     Compute orthonormal $V$ so that its columns form a basis for
           $\mathrm{span}_{i=1,\ldots,k}\{(s_i I + A)^{-1} C \hat{c}_i\}$.
4:     Project $A_k = V^T A V, C_k = V^T C$.
5:     Compute eigenvalue decomposition $A_k Z = Z\Lambda$.
6:     Update $s_i = \mathrm{diag}(\Lambda)$ and $\hat{C} = C_k^T Z$.
7: **end while**
8: OUTPUT: new poles $s_i$.
```

The convergence of Algorithm 5 depends on the initial poles but we rarely see a big difference in the number of iterations necessary for IRKA to converge whether the initial poles are in the spectral interval or all zeros. The computational cost is expected to be higher than for the Zolotarev poles, in particular if the problem has a large condition number. This is because for k initial poles, at each iteration we have to solve k shifted linear systems, and only make use of the final set of poles computed. However, the shifted linear systems we work with are tridiagonal and in most cases presented, the cost of generating the IRKA poles is comparable with the cost of generating the Zolotarev poles, as will be shown in Section 3.6. The steps of the algorithm are repeated until the relative change in the poles is below a given tolerance. In our experience, the

IRKA poles work well with a fairly loose stopping tolerance of 10^{-2}. Having a stricter tolerance does not improve the poles sufficiently to make a big difference in solver convergence and so would not justify the increased cost.

3.5.3 Adaptive pole choices

In this section we mention another approach for computing the poles for the rational Krylov subspace. In [23], the authors present an adaptive algorithm for solving the Lyapunov equation using a rational Krylov subspace. In this method, at each iteration, a new pole is computed "on-the-fly". This method uses the rational function, r_{k-1}, from (3.19), which interpolates $(x + y)^{-1}$ at the Ritz values of A from (2.9). The algorithm in [23] obtains the next pole s_{k+1} so that

$$s_{k+1} = \arg\left(\max_{s \in \delta v_k} \frac{1}{|r_{k-1}(s)|}\right), \tag{3.23}$$

where $v_k \subset \mathbb{C}^+$ approximates the mirrored spectral region of A and δv_k is its boundary. Once this pole is computed a new column for the basis is generated and used to project the matrix A until convergence, at each step discarding the previous pole.

For our numerical tests, we slightly modify Algorithm 4 to incorporate the adaptive pole choice technique from [23, Alg. 1]. We present a numerical comparison between our algorithm with the a priori poles described in Section 3.5.1 and Section 3.5.2 and with this adaptive approach in Section 3.6.

3.6 Numerical results – pole choices

We begin investigating the effect of the poles on the convergence rate of our method numerically by considering the Lyapunov equation $AX + XA^T = bb^T$ arising from the discretisation of the Poisson PDE (2.8). For this problem, we consider three right-hand sides to showcase different behaviours. These are a vector of ones; alternating 1s and -1s; and the right-hand side corresponding to the choice of $f(x, y)$ which leads to the solution $u(x, y) = x^2(1-x)^2y^2(1-y)^2$. We call the latter right-hand side choice the polynomial right-hand side.

We solve the resulting Lyapunov equations by the rational Krylov method in Algorithm 4 for these sets of poles: Zolotarev poles, logarithmically spaced values in the spectral interval, IRKA poles, and poles obtained as part of the adaptive method in [23, Alg. 1]. We cycle through each set of poles in descending order, and the solver terminates when $\|R\|/\|bb^T\| < \tau$. So that the algebraic error is commensurate with the discretisation error, we consider $\tau = 10^{-4}$ and $\tau = 10^{-8}$.

To better understand the behaviour induced by the different choices of right-hand side, as well as how convergence is affected by the different types of poles, we also consider two other classes of test problem. The first is a 2D Poisson PDE with variable diffusion coefficient, while the second is the original Poisson problem discretised using non-uniform meshes. We finish our discussion about the poles by exploring the case of higher rank right-hand sides. These are chosen by combining permutations of the three rank-1 variants.

3.6.1 Rank-1

Poisson problem – uniform mesh

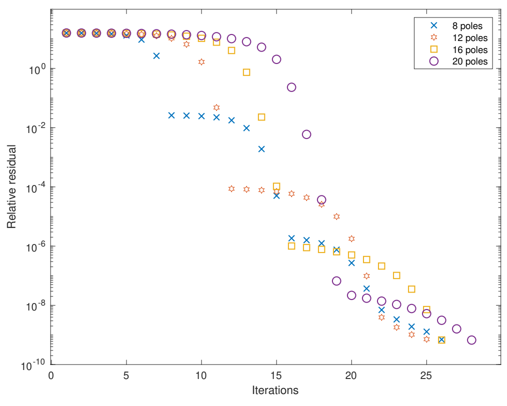
We begin our exploration by comparing different numbers of a priori poles in order to identify those that offer faster convergence rates for the Poisson problem with uniform mesh. We present our findings in Figure 3.5. In this case, the logspace poles resemble the Zolotarev ones so closely that the convergence is almost identical and we omit the results.

In general, we notice that most choices of poles and right-hand side combinations show good convergence for 12 or 16 poles. More extensive testing confirms this pattern more generally and for clarity, we limit subsequent results to 16 poles, unless otherwise stated.

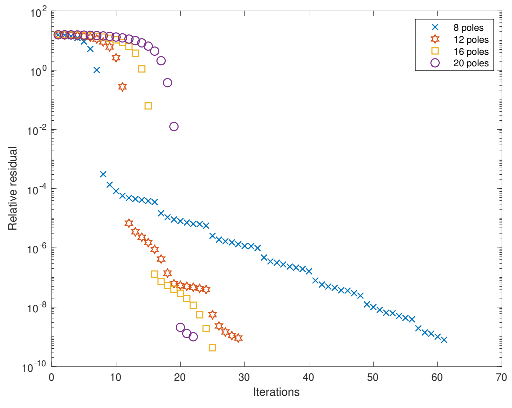
RHS	Poles	Time poles	$\tau = 10^{-4}$ It.	$\tau = 10^{-4}$ Time	Total time	$\tau = 10^{-8}$ It.	$\tau = 10^{-8}$ Time	Total time
Ones	Zolotarev	0.0322	16	0.3809	0.4131	25	0.6589	0.6911
	Logspace	0.0063	16	0.3993	0.4056	25	0.5814	0.5877
	IRKA	0.0373	16	0.3881	0.4254	23	0.5114	0.5487
Alt. $1, -1$	Zolotarev	0.0342	1	0.0428	0.0770	17	0.4151	0.4493
	Logspace	0.0060	1	0.0379	0.0439	17	0.4243	0.4303
	IRKA	0.0340	1	0.0356	0.0696	11	0.2764	0.3104
Polynomial	Zolotarev	0.0316	15	0.3109	0.3425	16	0.3823	0.4139
	Logspace	0.0026	15	0.3292	0.3318	16	0.3428	0.3454
	IRKA	0.0361	16	0.3837	0.4198	16	0.3737	0.4098

Table 3.1: Summary of a priori pole choices for Poisson problems with rank-1 right-hand sides. Total time columns represents the pole time and solver time added together at $\tau = 10^{-4}$ and $\tau = 10^{-8}$, respectively.

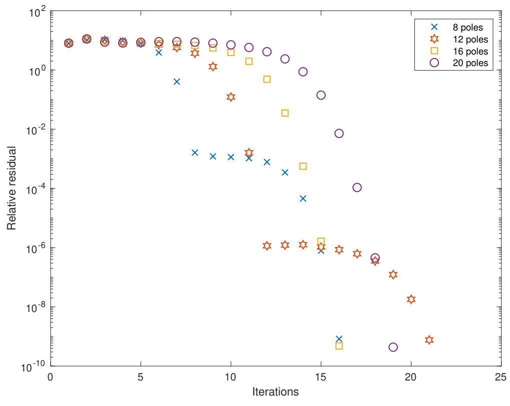
As well as the number of iterations, it is important to consider the time taken to generate poles and to solve the Lyapunov equation. Results are given in Table 3.1, from which we see that the three sets of poles denoted Zolotarev,



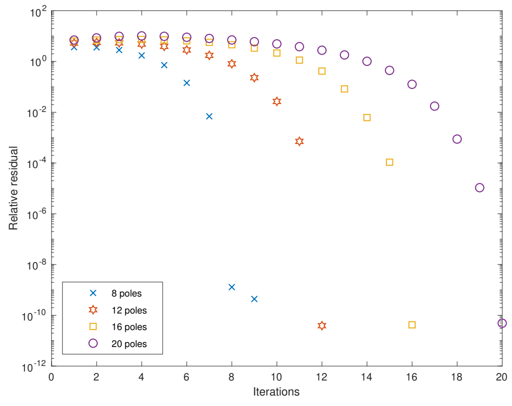
(a) Zolotarev poles. RHS ones.



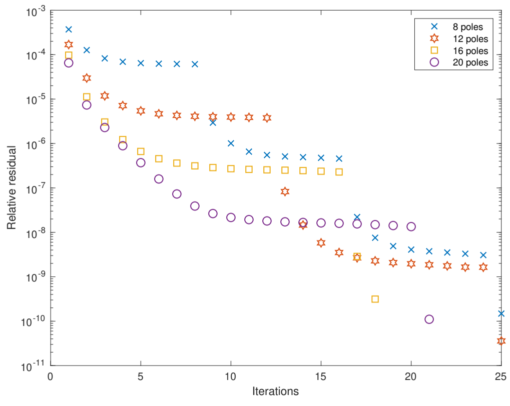
(b) IRKA poles. RHS ones.



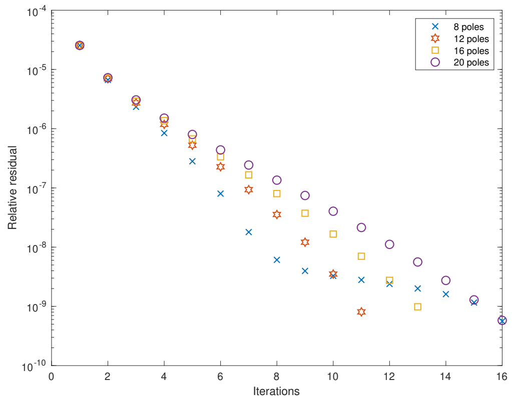
(c) Zolotarev poles. RHS polynomial.



(d) IRKA poles. RHS polynomial.



(e) Zolotarev poles. Alternating RHS.



(f) IRKA poles. Alternating RHS.

Figure 3.5: Convergence for the uniform mesh Poisson equation for different types and numbers of poles and for different right-hand sides.

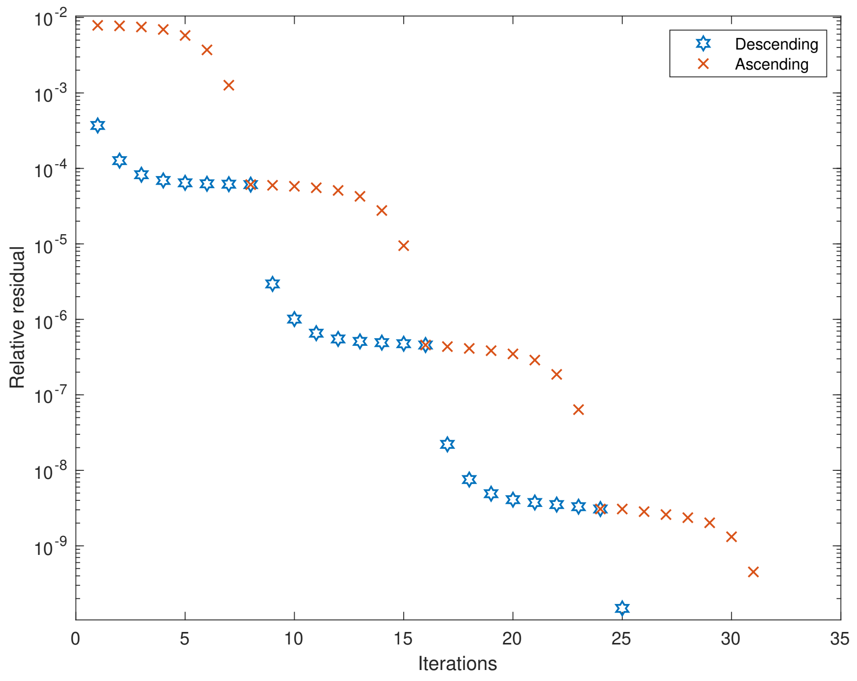
logspace, and IRKA perform similarly in terms of number of iterations of the Lyapunov solver at $\tau = 10^{-4}$. The IRKA poles always outperform the Zolotarev poles at a residual tolerance of $\tau = 10^{-8}$: substantially so for the alternating right-hand side, showing that faster convegence rates can be achieved for the rational Krylov subspace for pole choices other than the Zolotarev poles.

RHS	$\tau = 10^{-4}$		$\tau = 10^{-8}$	
	It.	Time	It.	Time
Ones	14	0.3557	23	0.5407
Alternating $1, -1$	2	0.1967	11	0.3144
Polynomial	7	0.2718	14	0.3504

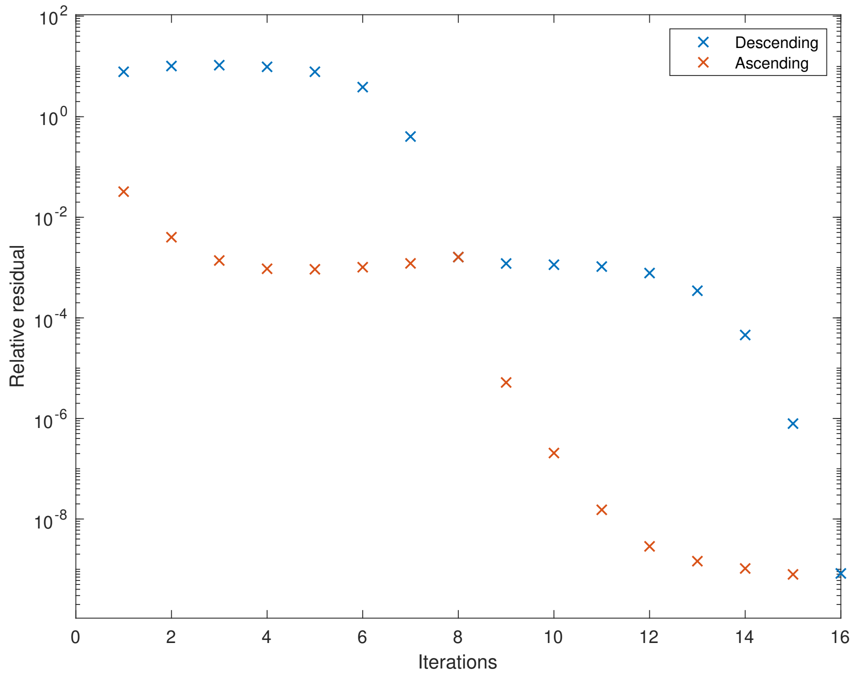
Table 3.2: Iteration counts and time (in seconds) to solve the Lyapunov equation with the adaptive poles approach from [23].

We compare our a priori results for Zolotarev, logarithmically spaced, and IRKA poles with the efficient and commonly used approach of adaptively computing the poles as part of the solver, as described in [23, Alg. 1] and in Section 3.5.3. We obtain in Table 3.2 iteration counts and time required to solve the Lyapunov equation for our three choices of right-hand side at both $\tau = 10^{-4}$ and $\tau = 10^{-8}$. We can see that, with the exception of the alternating right-hand side, the adaptive approach outperforms our a priori one at $\tau = 10^{-4}$. However, at $\tau = 10^{-8}$, the adaptive approach performs as well as our approach using the IRKA poles both in terms of iteration counts and convergence times.

The iteration counts for the alternating right-hand side at $\tau = 10^{-4}$ suggest that the ordering of the a priori poles might be an important factor in convergence. This is perhaps unsurprising because, in contrast to the other two choices, this right-hand side is highly oscillatory. In fact, Table 3.3 shows that the alternating right-hand side is almost orthogonal to the eigenvector corresponding to the smallest eigenvalue of A, but has a relatively small subspace angle with the eigenvector corresponding to the largest eigenvalue. This motivates us to examine the effect of ordering the poles in ascending and descending order.



(a) Right-hand side alternating 1 and -1.



(b) Right-hand side polynomial.

Figure 3.6: Comparison of convergence for the uniform mesh Poisson equation with 8 Zolotarev poles and both ascending and descending poles orderings.

The results are illustrated in Figure 3.6. We use 8 poles rather than 16 as the behaviour is more clearly demonstrated, although a similar phenomenon occurs for other numbers of poles.

RHS	angle w/ small eigenvec	angle w/ large eigenvec
alternating 1 and -1	90°	25°
ones	25°	90°
polynomial	9°	90°

Table 3.3: Subspace angles between choices of right-hand sides and eigenvectors corresponding to both smallest and largest eigenvalue for the Poisson PDE.

From Figure 3.6, we can see that convergence is indeed affected by the ordering of the a priori poles, but we also note that once we complete a full cycle through the poles, the residuals are the same. This is because once we iterate through all the poles, the rational Krylov subspaces are the same, whichever ordering we consider.

Clearly, the ordering of the a priori poles can improve convergence results, so, if possible, we recommend taking this into account when selecting the set of poles to be used. Note that in [74], multiple orderings are considered for poles in the context of ADI.

Variable diffusion coefficients

We now present results for the Poisson PDE with variable diffusion coefficients. Similarly to the uniform problem, we discretise the PDE

$$-\nabla \cdot (a(x)a(y)\nabla u) = f,$$

where $a(\cdot)$ is a bounded, positive function, using standard centred second-order finite differences on a uniform mesh to obtain the matrix equation

$$TUD + DUT = \tilde{F},$$

where $D = \operatorname{diag}(a(x_1), a(x_2), a(x_3), \cdots)$, and

$$T = \frac{1}{h^2}\begin{bmatrix} a(x_1 - \frac{h}{2}) + a(x_1 + \frac{h}{2}) & -a(x_1 + \frac{h}{2}) & & \\ -a(x_2 - \frac{h}{2}) & a(x_2 - \frac{h}{2}) + a(x_2 + \frac{h}{2}) & -a(x_2 + \frac{h}{2}) & \\ & -a(x_3 - \frac{h}{2}) & a(x_3 - \frac{h}{2}) + a(x_3 + \frac{h}{2}) & \\ & \ddots & \ddots & \end{bmatrix}.$$

Even though T is nonsymmetric, pre- and post-multiplying by $D^{-1/2}$ gives the symmetric matrix A which we use in the Lyapunov equation

$$AX + XA^T = F,$$

where $A = D^{-1/2}TD^{-1/2}$, $X = D^{-1/2}UD^{-1/2}$ and $F = D^{-1/2}\tilde{F}D^{-1/2}$. We solve this Lyapunov equation with the same right-hand side choices for F as before and record timings and numbers of iterations in each case. The times and iteration counts at the different τ, as well as the total poles and solver times for size $n = 1000$ with 16 poles are presented in Table 3.4 for diffusion coefficients given by the functions [35]

$$\begin{aligned} a_1(x) &= \sin(x), \\ a_2(x) &= 1 + 50\exp(-5x^2), \\ a_3(x) &= \begin{cases} 1, & \text{for } 0 < x < 1/2 \\ 10^4, & \text{for } 1/2 < x < 1. \end{cases} \end{aligned}$$

Diff. coeff.	RHS	Poles	Time poles	$\tau = 10^{-4}$ It.	$\tau = 10^{-4}$ Time	Total time	$\tau = 10^{-8}$ It.	$\tau = 10^{-8}$ Time	Total time
$a_1(x)$	Ones	Zolotarev	0.0223	16	0.3591	0.3814	29	0.4174	0.4397
		Logspace	0.0007	16	0.3324	0.3331	28	0.4131	0.4138
		IRKA	0.0305	16	0.2962	0.3267	28	0.3973	0.4278
	Alt.	Zolotarev	0.0348	5	0.0774	0.1122	20	0.2656	0.3004
		Logspace	0.0064	5	0.0708	0.0772	20	0.2646	0.2710
		IRKA	0.0226	6	0.1433	0.1659	18	0.2119	0.2345
	Poly.	Zolotarev	0.0283	16	0.3053	0.3336	27	0.5616	0.5899
		Logspace	0.0024	16	0.3542	0.0772	27	0.5281	0.5305
		IRKA	0.0279	16	0.3137	0.3416	16	0.3137	0.3416
$a_2(x)$	Ones	Zolotarev	0.0216	16	0.2430	0.2646	29	0.4751	0.4967
		Logspace	0.0025	16	0.2318	0.2343	29	0.4360	0.4385
		IRKA	0.0293	16	0.2203	0.2496	26	0.3700	0.3993
	Alt.	Zolotarev	0.0195	2	0.0848	0.1043	17	0.2901	0.3096
		Logspace	0.0032	1	0.0048	0.0080	17	0.3249	0.3281
		IRKA	0.0219	1	0.0405	0.0624	12	0.1835	0.2054
	Poly.	Zolotarev	0.0193	16	0.3747	0.3940	28	0.5700	0.5893
		Logspace	0.0020	16	0.4001	0.4021	28	0.5717	0.5737
		IRKA	0.0262	16	0.3119	0.3381	16	0.3119	0.3381
$a_3(x)$	Ones	Zolotarev	0.0191	16	0.4051	0.4242	25	0.6615	0.6806
		Logspace	0.0030	16	0.3909	0.3939	25	0.6154	0.6184
		IRKA	0.0501	16	0.3307	0.3808	23	0.5526	0.6027
	Alt.	Zolotarev	0.0190	6	0.2557	0.2747	18	0.4802	0.4992
		Logspace	0.0029	5	0.2353	0.2382	17	0.4554	0.4583
		IRKA	0.0452	6	0.2325	0.2777	16	0.4337	0.4789
	Poly.	Zolotarev	0.0190	15	0.5054	0.5244	16	0.5334	0.5524
		Logspace	0.0014	15	0.5213	0.5227	16	0.5475	0.5489
		IRKA	0.0499	16	0.4707	0.5206	16	0.4707	0.5206

Table 3.4: Summary of a priori pole choices for variable diffusion coefficient Poisson problems with uniform mesh spacing and rank-1 right-hand sides. Total time column represents the pole time and solver time added together at $\tau = 10^{-4}$ and $\tau = 10^{-8}$, respectively.

From Table 3.4 we see that, as in the uniform Poisson case, all our tested problems show very similar numbers of iterations at $\tau = 10^{-4}$. Furthermore, the diffusion coefficients $a_1(x) = \sin(x)$ and $a_2(x) = 1 + 50\exp(-5x^2)$ show similar performance to each other for all choices of right-hand side and a priori poles even at $\tau = 10^{-8}$. In these cases, we notice that using 16 IRKA poles results in fewer iterations. In the case of the piecewise diffusion coefficient, and the polynomial right-hand side, the IRKA poles results are consistent with those from the other two, while the Zolotarev and logspace poles seem to perform better than before.

Diff. coeff.	RHS	$\tau = 10^{-4}$		$\tau = 10^{-8}$	
		It.	Time	It.	Time
$a_1(x)$	Ones	18	0.3849	29	0.4737
	Alternating $1, -1$	6	0.2834	20	0.3802
	Polynomial	10	0.3202	20	0.4520
$a_2(x)$	Ones	17	0.3853	28	0.4556
	Alternating $1, -1$	2	0.1841	13	0.3382
	Polynomial	7	0.2875	17	0.4315
$a_3(x)$	Ones	14	0.3749	23	0.4306
	Alternating $1, -1$	5	0.2688	19	0.4703
	Polynomial	7	0.2952	14	0.4279

Table 3.5: Iteration counts and time (in seconds) required to solve the variable diffusion Lyapunov equation using the adaptive poles approach from [23].

As in the uniform case, we compare our a priori pole choices with the adaptive approach from Section 3.5.3. Table 3.5 contains iteration counts and computational times for the adaptive approach. We can see that the a priori generated IRKA poles outperform the adaptively computed poles for two out of the three diffusion coefficients we considered, with the piecewise setup benefiting more from the adaptive approach. As in the uniform case, we are interested in assessing how the solver with a priori poles compares with the adaptive approach. These results reiterate the usefulness of the IRKA poles for Lyapunov equation arising from the discretisation of some variable diffusion PDEs, since they show great performance alongside the adaptive pole choice.

In the cases of the alternating right-hand side, we still notice that very few iterations are required at $\tau = 10^{-4}$. As in the case of the uniform Poisson problem, we have computed the subspace angle between the right-hand side vector and the eigenvectors corresponding to the smallest and largest eigenvalues. Our findings for diffusion coefficient $a_1(x) = \sin(x)$ can be found in Table 3.6. Note that the other diffusion coefficients show similar results, so we omit them here.

RHS	angle w/ small eigenvec	angle w/ large eigenvec
alternating 1 and -1	90°	54°
ones	54°	90°
polynomial	10°	90°

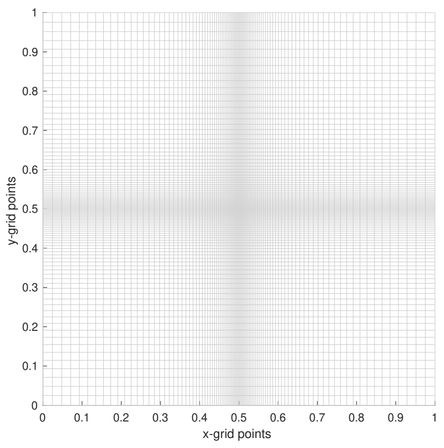
Table 3.6: Subspace angles between choices of right-hand sides and eigenvectors corresponding to both smallest and largest eigenvalue for the PDE with diffusion coefficient $a(x) = \sin(x)$.

We notice from Table 3.6 that, as in the case of the Poisson problem, the highly oscillatory right-hand side choice of alternating 1s and -1s has a smaller angle with the eigenvector corresponding to the largest eigenvalue, explaining again why there is an initial decrease in the norm of the residual when using a set of a priori poles where the largest one is used first.

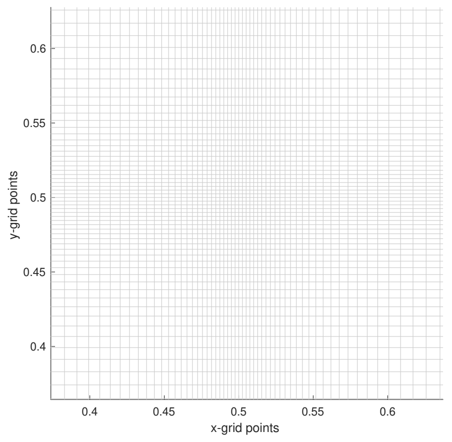
Poisson problem – nonuniform mesh

We now present results for the Poisson PDE generated with nonuniform meshes. This first set of results uses the graded mesh generator in [70] with $curve = 2$ and $weight = 0.1$, where the curve adjusts the steepness in the change of spacing, and the weight adjusts the ratio between the larger spacing at the boundaries and the smaller spacing in the centre of the domain. The other set of results is for a geometric mesh generated on $[0, 1/2]$ using a geometric sequence with ratio $r = 0.99$, which is mirrored on $[1/2, 1]$. In Figure 3.7 we present the two meshes for 100 grid points. Furthermore, in Figure 3.7(b), we zoom in on the graded mesh in order to show the change in spacing in the middle of the grid.

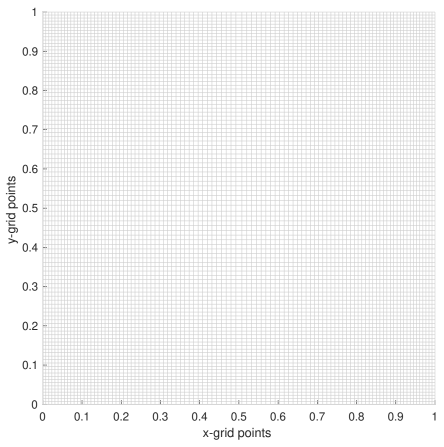
Note that the coefficient matrix obtained using the geometric mesh is more ill-conditioned than the others presented ($\kappa(A) = 7 \times 10^8$ for the geometric mesh,



(a) Graded mesh with 100 grid points.



(b) Closer look at the graded mesh with 100 grid points.



(c) Geometric mesh with 100 grid points.

Figure 3.7: Nonuniform meshes.

as compared to $\kappa(A) = 8 \times 10^6$ for the graded mesh). This affects both convergence rate and pole computation. To be precise, the Zolotarev poles require high precision arithmetic in order to generate the poles, as the elliptic integral computation cannot be obtained with sufficient accuracy in standard double precision. In order to overcome this, we use an asymptotic approximation of the elliptic integral given by the formula [15] $K = \ln(4) - 0.5\ln((\lambda_{\min}/\lambda_{\max})^2)$, where $\lambda_{\min}$ and $\lambda_{\max}$ are the smallest and largest eigenvalues of the coefficient matrix, respectively. The results for geometric mesh with Zolotarev poles in Table 3.7 use this approximation for the elliptic integral of first kind.

Mesh	RHS	Poles	Time poles	$\tau = 10^{-4}$		Total time	$\tau = 10^{-8}$		Total time
				It.	Time		It.	Time	
Graded	Ones	Zolotarev	0.0175	16	0.3488	0.3663	29	0.5328	0.5503
		Logspace	0.0016	16	0.3253	0.3269	28	0.5052	0.5068
		IRKA	0.0207	16	0.3070	0.3277	16	0.3070	0.3277
	Alt.	Zolotarev	0.0173	7	0.1946	0.2119	26	0.3971	0.4144
		Logspace	0.0013	7	0.1886	0.1899	33	0.4705	0.4718
		IRKA	0.0187	9	0.1927	0.2114	21	0.3033	0.3220
	Poly.	Zolotarev	0.0194	17	0.4435	0.4629	30	0.5989	0.6183
		Logspace	0.0012	17	0.4567	0.4579	30	0.6394	0.6406
		IRKA	0.0213	16	0.3002	0.3215	16	0.3002	0.3215
Geom.	Ones	Zolotarev	0.0165	29	0.5376	0.5541	43	0.6841	0.7006
		Logspace	0.0028	29	0.5517	0.5545	43	0.6196	0.6224
		IRKA	0.0793	33	0.5682	0.6475	139	4.2140	4.2933
	Alt.	Zolotarev	0.0191	20	0.4030	0.4221	39	0.6244	0.6435
		Logspace	0.0032	20	0.3873	0.3905	39	0.6149	0.6181
		IRKA	0.0253	14	0.6747	0.7000	81	1.8331	1.8584
	Poly.	Zolotarev	0.0192	29	0.5535	0.5727	33	0.7088	0.7280
		Logspace	0.0030	29	0.5932	0.5962	33	0.7085	0.7115
		IRKA	0.0684	16	0.2349	0.3033	16	0.2349	0.3033

Table 3.7: Summary of a priori pole choices for Poisson problems with nonuniform mesh spacing and rank-1 right-hand sides. Total time column represents the pole time and solver time added together at $\tau = 10^{-4}$ and $\tau = 10^{-8}$, respectively.

We can clearly see from Table 3.7 that the IRKA poles outperform the other two sets of poles for the graded mesh, for all choices of right-hand side at $\tau = 10^{-8}$, while they all perform similarly at $\tau = 10^{-4}$. This shows that having information about the right-hand side can improve the convergence of the solver for this type of problem. This is, however, not the case for the geometric mesh. In this case, it is very likely that because of the ill-conditioning of the coefficient matrix, the IRKA poles cannot improve solver convergence. In fact, they usually show much worse iteration counts than the Zolotarev or logspace poles. An exception is the polynomial right-hand side, in which case the IRKA poles result in solver convergence similar to the other two sets of poles. We believe this happens because of the existence of a low-rank solution of the Lyapunov problem in this case. Furthermore, we note that in two cases, the time that the solution with IRKA poles requires to converge to $\tau = 10^{-4}$ is smaller than for the other pole choices, which means that at this tolerance, the IRKA poles are preferrable.

Mesh	RHS	$\tau = 10^{-4}$		$\tau = 10^{-8}$	
		It.	Time	It.	Time
Graded	Ones	13	0.4072	29	0.4737
	Alternating $1, -1$	12	0.3166	24	0.4257
	Polynomial	9	0.2818	19	0.4320
Geometric	Ones	29	0.4726	19	0.6947
	Alternating $1, -1$	17	0.3472	37	0.5288
	Polynomial	15	0.3811	26	0.5872

Table 3.8: Iteration counts and time (in seconds) to solve the nonuniform mesh Lyapunov equation using the adaptive poles approach from [23].

Once again, as in the uniform and variable diffusion coefficient cases, we compare the a priori pole approaches with the adaptive method from Section 3.5.3. The adaptive pole generation results are presented in Table 3.8 for both graded and geometric meshes. We see from Table 3.7 and Table 3.8 that the IRKA poles outperform the adaptive approach at both $\tau = 10^{-4}$ and $\tau = 10^{-8}$ for the graded mesh in terms of both iteration numbers and convergence times, with the only exception appearing for the polynomial right-hand side at $\tau = 10^{-4}$. Meanwhile, for the geometric mesh, the adaptive poles show better convergence than the a priori pole approaches in all but one case given by the polynomial right-hand side, where the IRKA poles are superior.

3.6.2 Scalability

The numerical results for solving the Lyapunov equation we have presented so far have been for coefficient matrices of dimension $n = 1000$. In this section we show how our solver performs as we increase the size of the coefficient matrices. We do this in order to be able to draw conclusions about the most suitable set of poles. Before we present our results, we mention that we have changed our residual computation for the results in this section. To be precise, we no longer use the exact residual norm computation $\|R\|_F = \|AX_k + X_kA - bb^T\|_F$ as this is very expensive for large n and should not be computed in practice. In its place, we use a low cost residual norm computation of $\|R_k\|_F$ as described in [23, Proposition 4.1], which is mathematically equivalent, but slightly different in practice. Our stopping criterion is still given by $\|R_k\|_F/\|bb^T\|_F < 10^{-4}$.

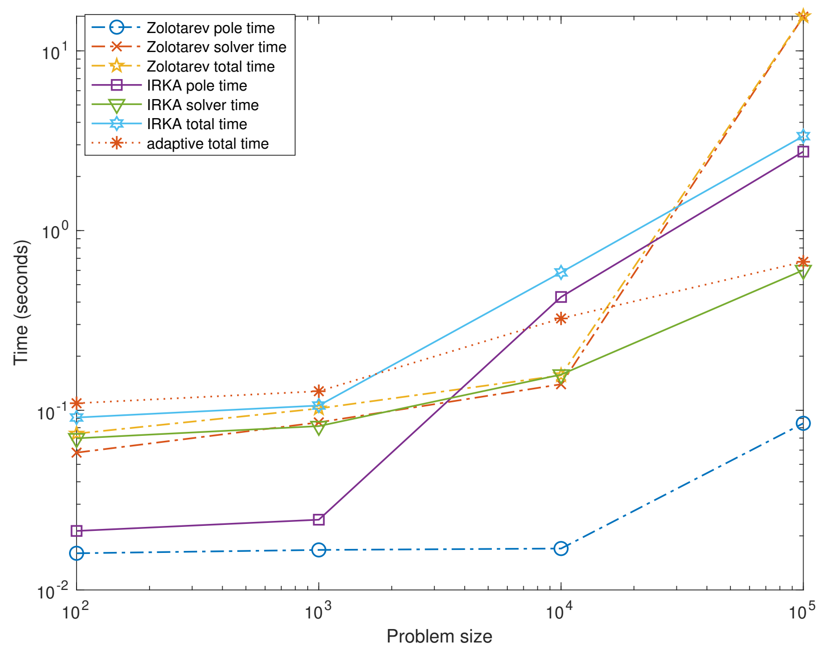
In our tests, we show the behaviours for different pole choices for Lyapunov equations of sizes of up to $n = 100000$, for three of the examples presented before, i.e., the original Poisson problem on a uniform mesh, the variable diffusion problem with $a_1(x) = \sin(x)$ and the graded nonuniform mesh problem. We consider the behaviour of Zolotarev, IRKA and adaptive poles for a right-hand side vector of ones in order to make a recommendation about the most suitable set of poles.

Uniform mesh

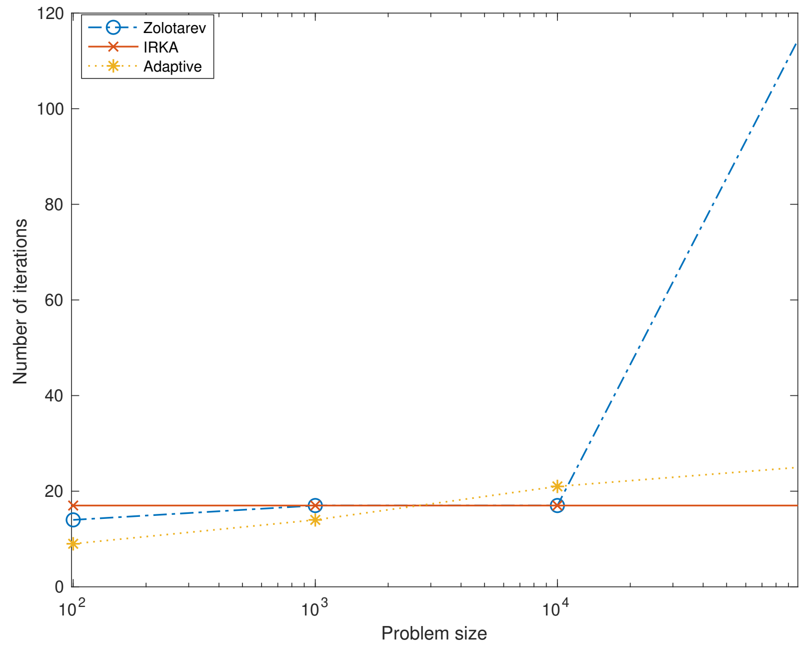
We first present results for the uniform mesh. In Figure 3.8(a) we present the times required for the a priori Zolotarev and IRKA poles to solve the Lyapunov equation, as well as the adaptive approach times. We compare the times required to generate the poles, the times required to solve the equation, as well as the total times for both sets of a priori poles with the total adaptive times. We can clearly see that despite the small pole generation time, the Zolotarev poles require a much longer time to solve the Lyapunov equation than the IRKA poles, which have a slightly higher pole generation time, but a cheaper solver time. As a result, the total IRKA times are much smaller overall than the total Zolotarev times as we increase the size of the problem. This is because, as we see in Figure 3.8(b), the number of iterations required to solve the Lyapunov equation with IRKA poles is constant, while with Zolotarev poles, this grows as we increase n. Therefore, for uniform mesh Lyapunov equations, the IRKA poles are a much better choice than the Zolotarev poles, as they seem to be optimal in terms of iteration counts. Comparing the a priori poles with the adaptive approach, we can see that, as we increase the problem size, the adaptive approach takes less time to solve the Lyapunov equation, despite showing a slight increase in the number of iterations.

Variable diffusion coefficient

We now present the convergence behaviour as we increase the problem size for a variable diffusion coefficient given by $a_1(x) = \sin(x)$. As in the uniform case, we show the times required to solve the Lyapunov equation with both a priori pole choices, Zolotarev and IRKA poles, as well as with the adaptive approach. The results are presented in Figure 3.9(a), in which we can see that, as in the uniform



(a) Times (in seconds) required to solve the Lyapunov equation using Zolotarev, IRKA, and adaptive poles.



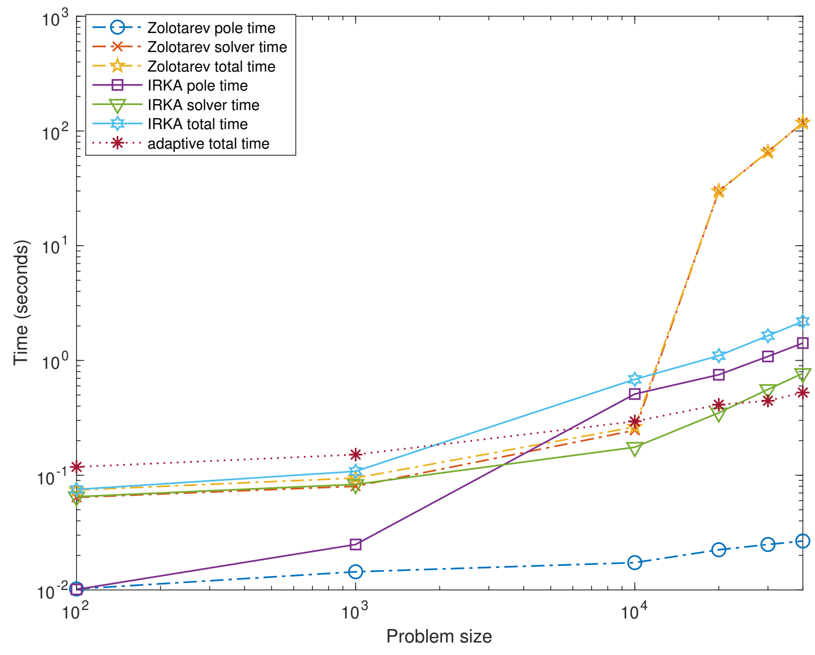
(b) Number of iterations required to solve the Lyapunov equation using Zolotarev, IRKA, and adaptive poles.

Figure 3.8: Comparison of time and iterations as the problem size increases for uniform mesh Lyapunov equations.

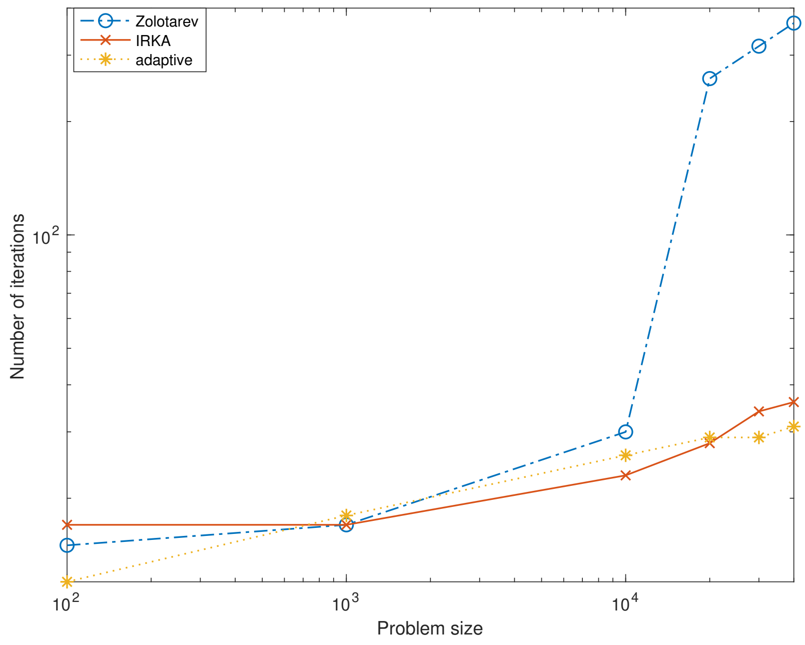
case, the cost of generating the IRKA poles is what drives the overall cost up with these poles. In the case of the Zolotarev poles, it is the solver time that increases drastically as we increase the size n, so much so, that we only show results up to $n = 40000$. This solver time for Zolotarev poles is so high as a result of the large number of iterations required. We see in Figure 3.9(b) that the solver with Zolotarev poles requires 365 iterations for $n = 40000$, in contrast to the much smaller number required by the solver with IRKA poles and by the adaptive approach. The number of iterations for the IRKA solver are no longer constant, as in the case of the uniform mesh, but slightly increase as we increase the size of the problem. This might be due to the problem becoming more ill-conditioned with the size. Despite this increase, as in the uniform case, in terms of a priori poles, we consider the IRKA poles a much better choice than the Zolotarev ones for the variable diffusion problem. However, when comparing the a priori results with the adaptive approach, we can see that, as we increase the size, solving the Lyapunov equation with the adaptive pole choice is less costly than solving with the a priori poles.

Graded mesh

Finally, we show computational time and iteration counts results for the Lyapunov equation with graded mesh for problems of size up to $n = 100000$. In Figure 3.10 we see the same curves that we saw for the uniform mesh and variable diffusion coefficient cases, however these present different properties than before. For the a priori poles, in the graded mesh case, we see that the total time required to solve the Lyapunov equation with IRKA poles takes longer than with Zolotarev poles. This is because of the time required to generate the IRKA poles, which grows similarly to the time required to solve the problem with Zolotarev poles. If we only compare solver times, then we see that using the IRKA poles gives smaller



(a) Times (in seconds) required to solve the Lyapunov equation using Zolotarev, IRKA, and adaptive poles.



(b) Number of iterations required to solve the Lyapunov equation using Zolotarev, IRKA, and adaptive poles.

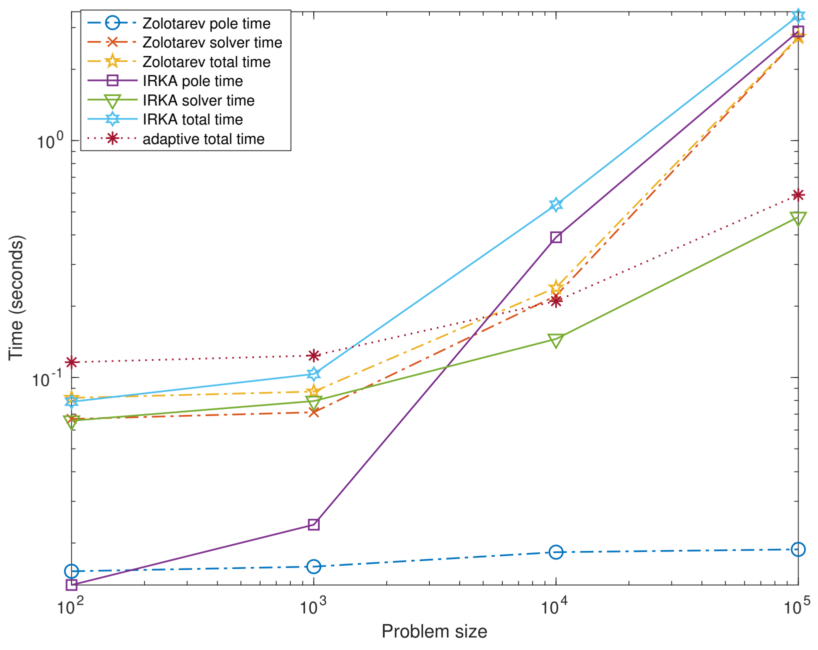
Figure 3.9: Comparison of time and iterations as the problem size increases for variable diffusion coefficient Lyapunov equations.

computational times. This results from the constant number of iterations required by the solver to converge when using IRKA poles, as seen in Figure 3.10(b). Therefore, as in the case of the uniform mesh, the IRKA poles seem to be optimal in terms of the number of solver iterations. Furthermore, as we have seen in the two previous problems, the adaptive poles approach outperforms both of the a priori pole choices, being quicker than the solver with the IRKA poles. This makes the adaptive poles more performant for solving large-scale Lyapunov equations with graded mesh.

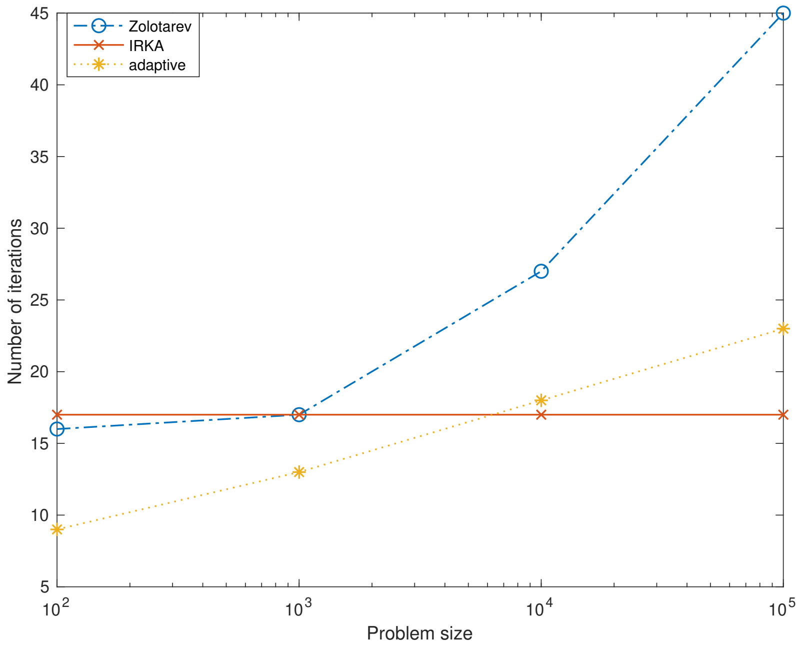
3.6.3 Higher rank right-hand sides

In this section we present some results which use a higher-rank right-hand side, i.e., $F = CC^T$, with $\text{rank}(C) > 1$, where the columns of C are formed from combinations of the rank-1 choices. We are interested to see effects of the rank of C on the convergence of the solver. We show our results for the Poisson problem with uniform and nonuniform meshes, and the diffusion problem with coefficient $a(x) = \sin(x)$. Note that the remaining two diffusion coefficient choices behave similarly to the one presented. We also only consider the Zolotarev and IRKA a priori poles (as the logarithmically spaced poles perform similarly to the Zolotarev ones), as well as the adaptive pole approach from Section 3.5.3. Furthermore, note that for the geometric mesh results, we have used the same elliptic integral approximation for the Zolotarev poles as in the rank-1 case.

We can see from Table 3.9 that, despite being slightly more expensive to compute, the IRKA poles perform somewhat better than the Zolotarev ones and than the adaptive approach in most cases at $\tau = 10^{-8}$ for the uniform mesh, variable diffusion coefficient and graded mesh cases. This means that they are a reliable set of poles for more accurate results in these cases. However, for the geometric



(a) Times (in seconds) required to solve the Lyapunov equation using Zolotarev, IRKA, and adaptive poles.



(b) Number of iterations required to solve the Lyapunov equation using Zolotarev, IRKA, and adaptive poles.

Figure 3.10: Comparison of time and iterations as the problem size increases for graded mesh Lyapunov equations.

Setup	RHS	Poles	Time poles	$\tau = 10^{-4}$		Total time	$\tau = 10^{-8}$		Total time
				It.	Time		It.	Time	
Uniform	Poly., Ones	Zolotarev	0.0146	13	0.5892	0.6038	23	0.9019	0.9165
		IRKA	0.0352	14	0.5243	0.5595	19	0.6781	0.7133
		Adaptive	-	12	-	0.6978	22	-	1.0215
	Poly., Alt.	Zolotarev	0.0171	14	0.6092	0.6263	17	0.6730	0.6901
		IRKA	0.0242	15	0.5490	0.5732	16	0.5909	0.6151
		Adaptive	-	8	-	0.4754	17	-	0.8382
	Ones, Alt.	Zolotarev	0.0158	15	0.6153	0.6311	25	0.9454	0.9612
		IRKA	0.0308	16	0.6135	0.6443	26	0.9882	1.0190
		Adaptive	-	15	-	0.7241	25	-	1.0918
Diff. coeff.	Poly., Ones	Zolotarev	0.0170	15	0.3170	0.3340	25	0.5574	0.5744
		IRKA	0.0329	15	0.3087	0.3416	23	0.4981	0.5310
		Adaptive	-	15	-	0.4612	25	-	0.6764
	Poly., Alt.	Zolotarev	0.0165	14	0.3690	0.3855	18	0.4008	0.4173
		IRKA	0.0323	14	0.3170	0.3493	18	0.3887	0.4210
		Adaptive	-	9	-	0.3564	21	-	0.6049
	Ones, Alt.	Zolotarev	0.0163	15	0.3138	0.3301	25	0.5664	0.5827
		IRKA	0.0338	16	0.3243	0.3581	25	0.5433	0.5771
		Adaptive	-	18	-	0.5289	28	-	0.7378
Graded	Poly., Ones	Zolotarev	0.0210	15	0.3314	0.3524	17	0.3681	0.3891
		IRKA	0.0437	14	0.2912	0.3349	15	0.3223	0.3660
		Adaptive	-	11	-	0.4141	17	-	0.5378
	Poly., Alt.	Zolotarev	0.0182	16	0.3508	0.3690	27	0.6136	0.6318
		IRKA	0.0396	16	0.3535	0.3931	28	0.6120	0.6516
		Adaptive	-	24	-	0.6427	35	-	0.9728
	Ones, Alt.	Zolotarev	0.0179	16	0.3546	0.3725	29	0.6580	0.6759
		IRKA	0.0417	16	0.3395	0.3812	28	0.6090	0.6507
		Adaptive	-	16	-	0.4685	31	-	0.9659
Geometric	Poly., Ones	Zolotarev	0.0158	22	0.5244	0.5402	33	0.7688	0.7846
		IRKA	0.0867	20	0.4212	0.5081	129	5.9077	5.9944
		Adaptive	-	18	-	0.5362	34	-	0.9259
	Poly., Alt.	Zolotarev	0.0175	20	0.4861	0.5036	39	0.8820	0.8995
		IRKA	0.0422	16	0.3299	0.3721	33	0.7225	0.7647
		Adaptive	-	21	-	0.7063	37	-	1.1531
	Ones, Alt.	Zolotarev	0.0195	29	0.6240	0.6435	43	1.0056	1.0251
		IRKA	0.0603	29	0.6197	0.6800	65	1.7396	1.7999
		Adaptive	-	32	-	0.7816	45	-	1.3052

Table 3.9: Summary of pole choices for Poisson problems with uniform mesh spacing and rank-2 right-hand sides. Total time column represents the pole time and solver time added together at $\tau = 10^{-4}$ and $\tau = 10^{-8}$, respectively.

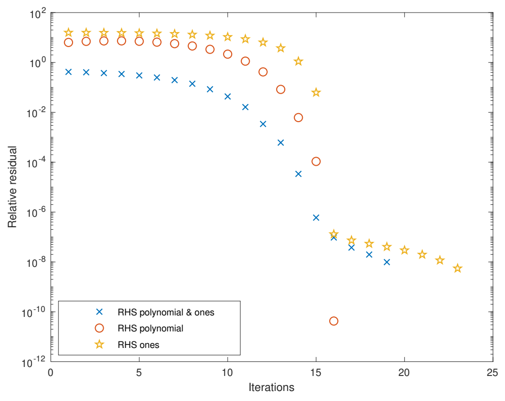
mesh, they lead to slow solver convergence. We noticed similar behaviour in the rank-1 choices of polynomial and ones. This happens because of the ill-conditioning of the matrix A and so, the Zolotarev poles are more appropriate for the geometric mesh spacing. Furthermore, note that unlike what is suggested in the literature, the adaptive approach takes longer per iteration, and overall, in all cases, making them less advantageous than the a priori options.

Comparing the results in Table 3.9 with the results for the rank-1 right-hand sides from Section 3.6.1, we notice that the iteration numbers for rank-2 right-hand sides seem to match the results of one of the two columns more than the other. To see if this is indeed the case, we have plotted the relative residual norms of the rank-2 right-hand side problems on the same graph as the individual columns. We have done this for three cases where the relationship between rank-1 and rank-2 results are more clearly shown, but this behaviour can be seen in general. We can see in Figure 3.11 that indeed, the behaviour of the higher-rank problems follows that of the rank-1 right-hand side which takes longest to converge since the "worse-off" column needs to be accounted for by the solver.

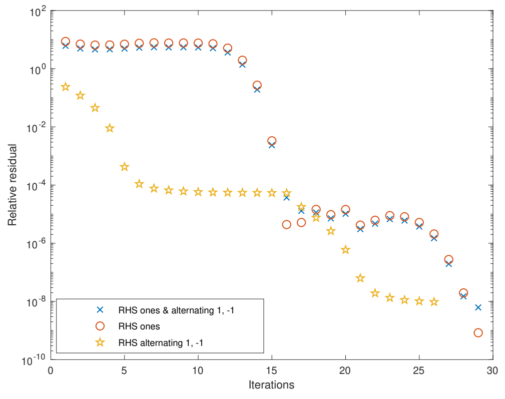
3.7 Conclusions

In this chapter, we have derived both 1- and 2-sided projections into rational Krylov subspaces, as well as explicit representations of the rational functions generated by these methods to approximately solve Lyapunov equations. We have also adapted the IRKA algorithm to generate poles for the rational Krylov subspace. We showed how the convergence of our solver compares to upper bounds for rational Krylov solvers and described the attainable accuracy of our approach. We also presented a numerical comparison between the 1-sided and 2-sided projections and concluded that the 2-sided approach is more appropriate for our problems.

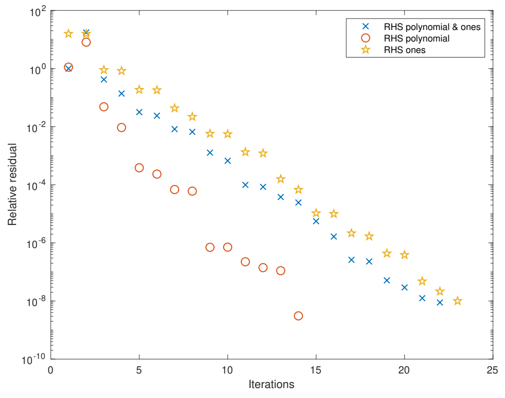
We studied the number of a priori poles and pole choices for a variety of problems and we can recommend the number and set of poles which should



(a) Uniform mesh convergence with IRKA poles and the ones and polynomial right-hand sides, as well as the rank-2 right-hand side formed from them.



(b) Graded mesh convergence with Zolotarev poles and the ones and alternating rank-1 right-hand sides, as well as the rank-2 right-hand side formed from them.



(c) Uniform mesh convergence with the adaptive poles approach and the ones and polynomial rank-1 right-hand sides, as well as the rank-2 right-hand side formed from them.

Figure 3.11: Convergence comparison for rank-1 right-hand sides and the corresponding rank-2 combinations.

be used in practice for discretised diffusion problems. Let us first summarise results for a priori pole choices. For problems with uniform mesh spacing, we recommend the use of between 12 and 16 IRKA poles as these can improve the convergence of the solver.

Additionally, we considered two types of nonuniform meshes, a graded one and a geometric one. It is clear from our results that the IRKA poles outperform the Zolotarev ones in the graded case. The ill-conditioning of the coefficient matrix A arising from geometric meshes cause IRKA poles to perform poorly when a strict tolerance is required. For the Zolotarev poles we can use an asymptotic approximation instead and so, in the case of ill-conditioned systems (condition number greater than 10^7, say), where a high accuracy is required, then we recommend the use of Zolotarev poles with asymptotic approximation.

Moreover, we considered how the time required to solve the Lyapunov equations changes as we increase the size of the problem. We have seen that the IRKA poles are optimal in terms of iteration counts in the cases of uniform and graded meshes, making them a favourable choice despite the higher computational cost to generate them. For the variable diffusion coefficient case, we have seen that the number of iterations increases slightly with n when the IRKA poles are used, but the large number of iterations required to solve the Lyapunov equation with the Zolotarev poles also increases greatly the solver time, making the IRKA poles favourable in this case, too.

For all our problems, we compared the convergence of the Lyapunov solver using a priori poles with the adaptive pole choice approach. For problems of moderate size, we have seen that in most cases where the IRKA poles perform better than Zolotarev, the convergence with adaptive pole choices is similar to this, meaning that the IRKA poles are, for these problems, comparable with the state-of-the-art adaptive approach. For large problems, the adaptive poles show quicker convergence, despite often needing more iterations that IRKA.

Furthermore, the ordering of the poles affects the speed of convergence. The usual ordering we have used in our results has been descending. However, there are some choices of right-hand side, such as the ones and polynomial cases, which benefit from an ascending pole ordering. This is a consequence of the angle between the right-hand side vector and the eigenvector corresponding to either the smallest or largest eigenvalue. If possible, we recommend to check this subspace angle and choose the pole ordering accordingly, and note that highly oscillatory right-hand sides may benefit from descending order.

Lastly, we looked at a number of higher rank right-hand sides given by combinations of the rank-1 right-hand sides. The results we obtained mimicked those of the rank-1 problems and our recommendations from above still hold in terms of number and sets of poles. One thing we recommend users note about the higher rank right-hand side problems is that the convergence is usually influenced by the “worse-off” of the right-hand side columns.

Chapter 4

Convection–diffusion problems

4.1 Introduction

The discretised convection–diffusion PDE can be written as the matrix equation

$$-\epsilon(AX + XA^T) + \Phi_1 BX\Psi_1 + \Phi_2 XB^T\Psi_2 = F,$$

as we derived in (2.14). Recall that A is symmetric positive definite, B is non-symmetric Toeplitz, and both are tridiagonal, while F is low rank, and $\Phi_{1,2}$ and $\Psi_{1,2}$ are diagonal matrices. This equation is formed by adding the symmetric Poisson operator from Section 2.2.1, which is a Lyapunov operator, to terms corresponding to the convective component of the PDE in (2.12). Depending on our choice of convection wind, without loss of generality, with the possibility to interchange $\Phi_1, \Phi_2, \Psi_1, \Psi_2$, and up to a constant factor, the matrix equation can be one of three types:

- a Lyapunov equation, i.e., $\Phi_1, \Phi_2, \Psi_1, \Psi_2 = I$;

- a Sylvester equation, i.e., $\Phi_1, \Psi_2 = I$, so that we have $-\epsilon(AX + XA^T) + \Phi_1 BX + XB^T\Psi_2 = F$ [64];
- a four-term matrix equation like (2.14), also known as a generalized Sylvester equation [49, 75].

In this chapter we consider each of these three types of matrix equations arising from the discretisation of convection–diffusion PDEs and discuss solution strategies, characteristics and limitations.

We begin this chapter by considering the Lyapunov equation, i.e., the first type above. We solve this using a projection into a rational Krylov subspace in a similar manner to the approach we used in Chapter 3. As the coefficient matrices involved can be more nonnormal than those from Lyapunov equations we have seen previously, we consider two upper bounds for the error of this approach, one using the eigenvector condition number, and one using the field of values. Note that a looser field of values bound is also presented in [21] in order to show the relationship between the ADI and rational Krylov methods. We then present suitable pole choices for the rational Krylov subspace and compare the Lyapunov solver convergence with these sets of poles.

Section 4.3 presents observations about Sylvester equations which arise from the discretisation of the convection–diffusion problem. We describe a projection-based approach for solving the Sylvester equation and provide insights into how we can use the Zolotarev and IRKA poles to form rational Krylov subspaces. We then present numerical results covering three main types of Sylvester equations that arise from convection–diffusion equations and make our recommendation of suitable poles.

Finally, we consider the four-term matrix equation and present a stationary iterative method for solving this. Such an approach is presented in [75] for generalized Lyapunov equations. We extend these results to generalized Sylvester equations, allowing us to solve the four-term matrix equation in (2.14). Note that the generalized Sylvester equation is considered in [49] for matrices that satisfy a commuting property, which is not the case for convection–diffusion coefficient matrices. We provide a criterion for the convergence of the stationary iteration. We then perform a series of numerical results to test our method, and focus on the limitations of such an approach for matrix equations arising from convection–diffusion PDEs.

4.2 Lyapunov equation

In this section we discuss the Lyapunov equation arising from the discretisation of convection–diffusion PDEs. As in the case of diffusion problems, we are interested in using projection methods to solve our matrix equations. When the convection wind is constant, i.e., $w = \alpha(1, 1)$, then our matrix equation is given by a Lyapunov equation given by

$$-\epsilon(AX + XA^T) + \alpha(BX + XB^T) = F,$$

which for simplicity we will write as

$$MX + XM^T = F, \quad M = -\epsilon A + \alpha B.$$

Note that here, the scalar α can be any non-zero number, and that in our numerical examples, we will use $\alpha = 1$. However, a more general wind direction, $w = (w_1, w_2)$ will lead to a Sylvester equation. The coefficient matrix M is a nonsymmetric tridiagonal real-valued Toeplitz matrix discretised on a uniform mesh and so, depending on our choice of size n, the diffusion coefficient ϵ, and the scalar α, the eigenvalues of the matrix M can be real or complex. Recall that we can compute the exact eigenvalues for such a matrix using the formula presented in (2.1),

$$\lambda_k = a - 2\sqrt{bc}\cos\left(\frac{k\pi}{n+1}\right), \quad k = 1, 2, \ldots, n,$$

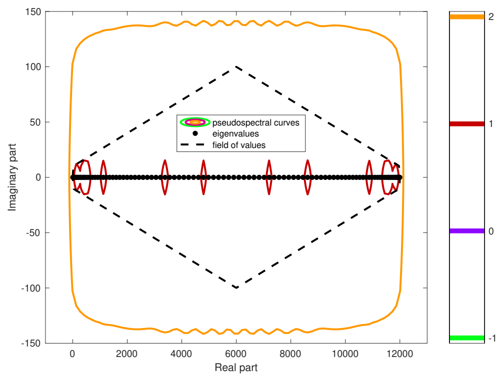
where a is the entry on the main diagonal and b, c are the entries on the sub- and super-diagonals of M. Then, using this, it is clear that for the eigenvalues to be complex, we need the product $bc < 0$. For our convection–diffusion problem, this product is given, in terms of n, α and ϵ, by $n^4\epsilon^2 - \alpha^2 n^2/4$. It is easy to show that this is negative when $n\epsilon < \alpha/2$. We require (2.1) in this section because we will encounter matrices which are far from normal and for which, as a result, we cannot compute the eigenvalues accurately using the built-in `MATLAB` functions.

When finite differences are used to discretise the convection-diffusion PDE care must be taken to avoid unstable methods that cause spurious oscillations in the solution [38]. As our problem is on a uniform grid, with mesh size $h \leq 1/n$, this can be quantified using the mesh Péclet number

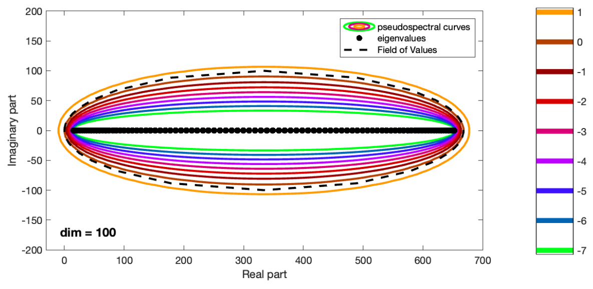
$$P_e = \frac{h|w|}{2\epsilon},$$

where $|w| = \|w\|_\infty = \max(|w_1|, |w_2|)$ is a measure of how convective a grid point is. Note that, in particular, non-physical oscillations will occur in the solution if the Péclet number is greater than one.

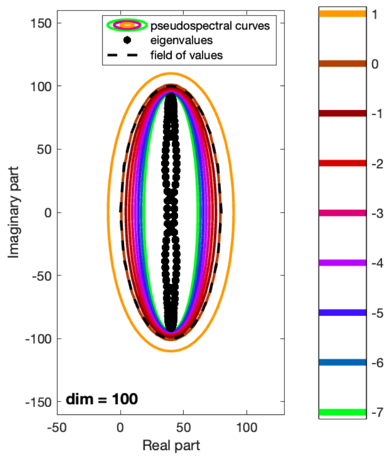
The size n of the matrix, the diffusion parameter ϵ and the scalar α play an



(a) $\epsilon = 0.3$ corresponding to M with real eigenvalues, eigenvector condition number 5.2857, and mesh Péclet number 0.0236.



(b) $\epsilon = 0.0167$ corresponding to M with real eigenvalues, eigenvector condition number 2.36×10^{13}, and mesh Péclet number 0.4234.



(c) $\epsilon = 0.002$ corresponding to M with complex eigenvalues, eigenvector condition number 3.62×10^{16}, and mesh Péclet number 3.5355.

Figure 4.1: Eigenvalues, field of values and pseudospectrum for convection–diffusion coefficient matrix of size $n = 100$, with $w = (1, 1)$.

important role in the distribution of the eigenvalues and the conditioning of the eigenvector matrices. When ϵ is large (close to 1), then M has real spectrum, and it is also well conditioned. However, when ϵ is small, but bigger than $\frac{\alpha}{2n}$, then M is far from normal, and so, it can influence the solvability of the matrix equation. When ϵ drops below $\frac{\alpha}{2n}$, the problem becomes even harder to solve since the spectrum of M includes complex eigenvalues, and the condition number of the eigenvector matrix is large. These three different cases are shown below for three values of ϵ, a fixed value of n, and $\alpha = 1$. We plot in Figure 4.1 not just the spectrum of M, but also the field of values and pseudospectrum using `eigtool` [94]. Note that by computing the exact eigenvalues using (2.1), we have seen that we obtain complex eigenvalues if $2n\epsilon < \alpha$. Furthermore, from the definition of the Péclet number, we can also see that this is greater than one if $2n\epsilon < |w|$. Therefore, using the two inequalities, we can state that we have complex eigenvalues, as well as $P_e > 1$ for cases where $|w| = \alpha$. We see such an example below, in Figure 4.1(c). This means that this problem is not only more difficult to solve, but also susceptible to spurious oscillations in the solution.

We can see from Figure 4.1 (with $\alpha = 1$) that, as expected, the choice of ϵ affects the distribution of the eigenvalues. We are, in practice, interested in cases where ϵ is small so that the coefficient matrix (with real spectrum) is dominated more by the convection part of the PDE. This means that we will find ourselves in cases similar to Figure 4.1(b), where our matrices are far from normal. We will see in subsequent sections what this means in terms of pole choices.

4.2.1 Projection

Since we are interested in the convection-dominated case which leads to a nonnormal matrix M with real spectrum, we cannot use the approach described in Section 3.2.3 as presented. The changes required do not apply to the 2-sided approximate solution derivation, but to the way in which we select the poles and construct the rational function in (3.19), as we now are required to take into account the possibility of using complex poles.

Recall, the rational approximation to the solution of the Lyapunov equation was given by (3.18),

$$\operatorname{vec}(X_k) = r_{k-1}(M \otimes I, I \otimes M)\operatorname{vec}(bb^T),$$

with the rational function given by $r_{k-1}(x,y) = \frac{P(x,y)}{q_{k-1}(x)q_{k-1}(y)}$, where $P(x,y)$ is a polynomial of degree at most $k-1$ in both x and y, and $q_{k-1}(x)$ is a single variable polynomial of degree at most $k-1$, with the poles $\mathcal{S}$ as roots, so that $r_{k-1}(x,y)$ interpolates $f(x,y) = (x+y)^{-1}$ at the Ritz values $\sigma(M_k)$, where M_k is the projected coefficient matrix, $M_k = V^TMV$. In the case when we have a complex spectrum, this rational function interpolation is still valid, but our chosen poles are complex. As in [21], we will choose poles in conjugate pairs, so that $\mathcal{S} = \{s_1, \ldots, s_k\} = \{\overline{s_1}, \ldots, \overline{s_k}\} = \overline{\mathcal{S}}$. When the poles and the Ritz values coincide, the rational function can then be written as

$$r_{k-1}(x,y) = \frac{1}{x+y}\Big(1 - \prod_{i=1}^{k-1} \frac{(x-\overline{s}_i)(y-s_i)}{(x+s_i)(y+\overline{s}_i)}\Big),$$

with x, y corresponding to $M \otimes I$ and $I \otimes M$, respectively, and with $\frac{1}{x}$ interpreted as the inverse x^{-1}. Therefore, the error is given by

$$\begin{aligned} E(x,y) &= \frac{1}{x+y} \prod_{i=1}^{k-1} \frac{(x-\overline{s}_i)(y-s_i)}{(x+s_i)(y+\overline{s}_i)} \operatorname{vec}(bb^T) \\ &= \frac{1}{x+y} \tilde{R}(x,y) \operatorname{vec}(bb^T), \end{aligned} \tag{4.1}$$

with

$$\tilde{R}(x,y) = \prod_{i=1}^{k-1} \frac{(x-\overline{s}_i)(y-s_i)}{(x+s_i)(y+\overline{s}_i)}.$$

The error is large in norm when $\|\tilde{R}(x,y)\|_2$ is large. Note that $\tilde{R}(x,y) = \prod_{i=1}^{k-1} \frac{(x-\overline{s}_i)}{(x+s_i)} \prod_{i=1}^{k-1} \frac{(y-s_i)}{(y+\overline{s}_i)}$. Let [21]

$$\tilde{r}(z) = \prod_{i=1}^{k-1} \frac{z-\overline{s}_i}{z+s_i} = \overline{\prod_{i=1}^{k-1} \frac{\overline{z}-s_i}{\overline{z}+\overline{s}_i}} = \overline{\tilde{r}(\overline{z})}. \tag{4.2}$$

Then we have that

$$\tilde{R}(x,y) = \prod_{i=1}^{k-1} \frac{(x-\overline{s}_i)}{(x+s_i)} \prod_{i=1}^{k-1} \frac{(y-s_i)}{(y+\overline{s}_i)} = \tilde{r}(x)\overline{\tilde{r}(\overline{y})}, \tag{4.3}$$

and the minimax problem corresponding to (3.20) is

$$\min_{s_i \in \mathcal{S}} \max_{z \in \sigma(A)} \left| \tilde{r}(z)\overline{\tilde{r}(\overline{z})} \right|.$$

Assuming that ϵ and n are chosen so that the spectrum is real, then as in Section 3.2.3, this can be relaxed so that the error is maximised over the spectral interval, giving

$$\min_{s_i \in \mathcal{S}} \max_{z \in [\rho_{\min}, \rho_{\max}]} \left| \tilde{r}(z)\overline{\tilde{r}(\overline{z})} \right|. \tag{4.4}$$

This relaxed minimisation problem corresponds to the third Zolotarev problem.

We are interested in bounding the error described above. Since for convection–diffusion problems we can have both real and complex eigenvalues, we consider two bounds for the rational error: one which bounds the error using eigenvectors, and one which bounds it using the field of values. Note that the eigenvector bound holds only for matrices with real eigenvalues, while the field of values bound holds for matrices with both real and complex eigenvalues.

4.2.2 Eigenvector bound

In this subsection we consider an eigenvector bound for the error from (4.1). In order to derive this bound, we require that the coefficient matrix has real and positive eigenvalues, which is equivalent to satisfying the relationship $\epsilon > \frac{1}{2n}$.

Theorem 4.2.1. *Let $A = Z\Lambda Z^{-1}$ be a diagonalisable matrix, with real and positive eigenvalues, where Z is the matrix of eigenvectors and Λ is the diagonal matrix of eigenvalues, let $\mathcal{S} = \{s_1, \ldots, s_k\}$ be an a priori chosen set of poles, and let $\kappa(Z) = \|Z\|_2\|Z^{-1}\|_2$ be the 2-norm eigenvector condition number. Then*

$$\|E(A\otimes I, I\otimes A)\|_2 \le \frac{1}{2\lambda_{\min}}\kappa(Z)^2 \min_{s_i\in\mathcal{S}} \max_{z\in\sigma(A)} |\tilde{r}(x)\overline{\tilde{r}(\overline{z})}|\|b\|_2^2.$$

Proof. Substituting $A = Z\Lambda Z^{-1}$ into the Kronecker product form of the error expression (4.1), we have

$$E(A\otimes I, I\otimes A) = (Z\otimes Z)(\Lambda\otimes I + I\otimes\Lambda)^{-1}\tilde{R}(\Lambda\otimes I, I\otimes\Lambda)(Z\otimes Z)^{-1}\operatorname{vec}(bb^T).$$

We want to bound the norm of $E(A \otimes I, I \otimes A)$. With $\|\cdot\| = \|\cdot\|_2$, we have that

$$\begin{aligned} \|E(A \otimes I, I \otimes A)\| &= \|(Z \otimes Z)(\Lambda \otimes I + I \otimes \Lambda)^{-1}\tilde{R}(\Lambda \otimes I, I \otimes \Lambda)(Z \otimes Z)^{-1} \operatorname{vec}(bb^T)\| \\ &\leq \|Z \otimes Z\|\|(\Lambda \otimes I + I \otimes \Lambda)^{-1}\|\|\tilde{R}(\Lambda \otimes I, I \otimes \Lambda)\|\|(Z \otimes Z)^{-1}\|\|bb^T\| \\ &= \kappa(Z)^2\|(\Lambda \otimes I + I \otimes \Lambda)^{-1}\|\|\tilde{R}(\Lambda \otimes I, I \otimes \Lambda)\|\|bb^T\|. \end{aligned} \tag{4.5}$$

Note that $\|(\Lambda \otimes I + I \otimes \Lambda)^{-1}\|_2 = \frac{1}{\min_i \tau_i}$, where τ_i is a singular value of $\Lambda \otimes I + I \otimes \Lambda$. As this matrix is diagonal and contains sums of the real and positive eigenvalues of A, then, in this case, $\min_i \tau_i = 2\lambda_{\min}$ and so

$$\|(\Lambda \otimes I + I \otimes \Lambda)^{-1}\| = \frac{1}{2\lambda_{\min}}. \tag{4.6}$$

Given that

$$\|\tilde{R}(\Lambda \otimes I, I \otimes \Lambda)\| = \min_{s_i \in \mathcal{S}} \max_{z \in \sigma(A)} |\tilde{r}(z)\overline{\tilde{r}(\overline{z})}|, \tag{4.7}$$

the bound for the error follows from (4.5), (4.6), and (4.7). □

It is clear that for highly nonnormal matrices, as in Figure 4.1(b), the eigenvector bound is dominated by the condition number of the eigenvector matrix. Furthermore, note that even if we were to compute the eigenvalues and eigenvectors, which is very expensive, this bound might not be descriptive if $\kappa(Z) \gg 1$.

4.2.3 Field of values bound

In this section we present a field of values bound. The field of values is not as sensitive to perturbations as the eigenvectors and eigenvalues for nonnormal matrices [29], making it more attractive in cases when the matrix is far from normal.

Theorem 4.2.2. *Let $W(A)$ be the field of values of A, let $c_1 = (2\,\mathrm{dist}(0, W(A)))^{-1}$, and let c_2 be the Crouzeix constant [16]. Then*

$$\|E(A \otimes I, I \otimes A)\|_2 \leq c_1 c_2^2 \max_{z \in W(A)} |\tilde{r}(z)|^2 \|b\|_2^2.$$

Proof. From (4.1), with $\|\cdot\| = \|\cdot\|_2$,

$$\begin{aligned}\|E(A \otimes I, I \otimes A)\| &= \|(A \otimes I + I \otimes A)^{-1} \tilde{R}(A \otimes I, I \otimes A)\,\mathrm{vec}(bb^T)\| \\ &\leq \|(A \otimes I + I \otimes A)^{-1}\| \|\tilde{R}(A \otimes I, I \otimes A)\,\mathrm{vec}(bb^T)\|.\end{aligned}$$

We look to bound the two terms on the right-hand side separately. We begin with $\|(A \otimes I + I \otimes A)^{-1}\|$ and, for convenience, derive the bound in [21], where the authors first show that $W(A \otimes I) = W(A) = W(I \otimes A)$ by explicitly writing the Rayleigh quotients for each term. Then, $W(A \otimes I + I \otimes A) \subseteq W(A \otimes I) + W(I \otimes A) = W(A) + W(A)$, and so,

$$\mathrm{dist}(0, W(A \otimes I + I \otimes A)) \geq \mathrm{dist}(0, W(A) + W(A)) = 2\,\mathrm{dist}(0, W(A)),$$

so that

$$\|A \otimes I + I \otimes A\| \leq c_1. \tag{4.8}$$

Next, we look to bound $\|\tilde{R}(A \otimes I, I \otimes A)\,\mathrm{vec}(bb^T)\|$. We have shown in (4.3) that $\tilde{R}(A \otimes I, I \otimes A) = \tilde{r}(A \otimes I)\overline{\tilde{r}(\overline{I \otimes A})}$ so, using (2.6) and noting that A is real, we have

$$\begin{aligned}\|\tilde{r}(A \otimes I)\overline{\tilde{r}(\overline{I \otimes A})}\,\mathrm{vec}(bb^T)\| &= \|\tilde{r}(A) \otimes \overline{\tilde{r}(A)}\,\mathrm{vec}(bb^T)\| \\ &= \|\mathrm{vec}([\overline{\tilde{r}(A)}b][\tilde{r}(A)b]^T)\|.\end{aligned}$$

We use the results by Crouzeix from [16], i.e., for a matrix function $f(A)$ which is analytic, we have

$$\|f(A)\| \leq c_2 \max_{z \in W(A)} |f(z)|.$$

The most recently derived value of c_2 is $1+\sqrt{2}$, as presented in [17]. We then have that

$$\|\tilde{r}(A)b\| \leq c_2 \max_{z\in W(A)} |\tilde{r}(z)|\|b\|,$$

so that

$$\|\tilde{r}(A\otimes I)\overline{\tilde{r}(\overline{I\otimes A})}\,\mathrm{vec}(bb^T)\| \leq c_2^2 \max_{z\in W(A)} |\tilde{r}(z)|^2\|b\|^2. \tag{4.9}$$

The bound follows by combining (4.8) and (4.9). □

This bound has contributions from both c_1 and the Crouzeix constant, and they can provide a more realistic picture of the accuracy of the rational Krylov method, especially when the coefficient matrices are far from normal.

4.2.4 Poles

In the previous sections, we bounded the error using both an eigenvector and a field of values approach. Both these bounds use the rational function $\tilde{r}(z) = \prod_{i=1}^{k-1} \frac{z-\bar{s}_i}{z+s_i}$ from (4.2), which depends on the poles s_i. In this section we consider appropriate pole choices for the convection–diffusion Lyapunov equation,

$$MX + XM^T = F, \quad M = -\epsilon A + B,$$

with A and B the Poisson and convection matrices described in Section 2.2. We will look at the Zolotarev and IRKA pole choices, also presented in Section 3.5 for symmetric matrices, and discuss the changes we are required to make in order to use those sets of poles to solve the nonsymmetric Lyapunov equation above. We will then use those poles in practice and assess their suitability.

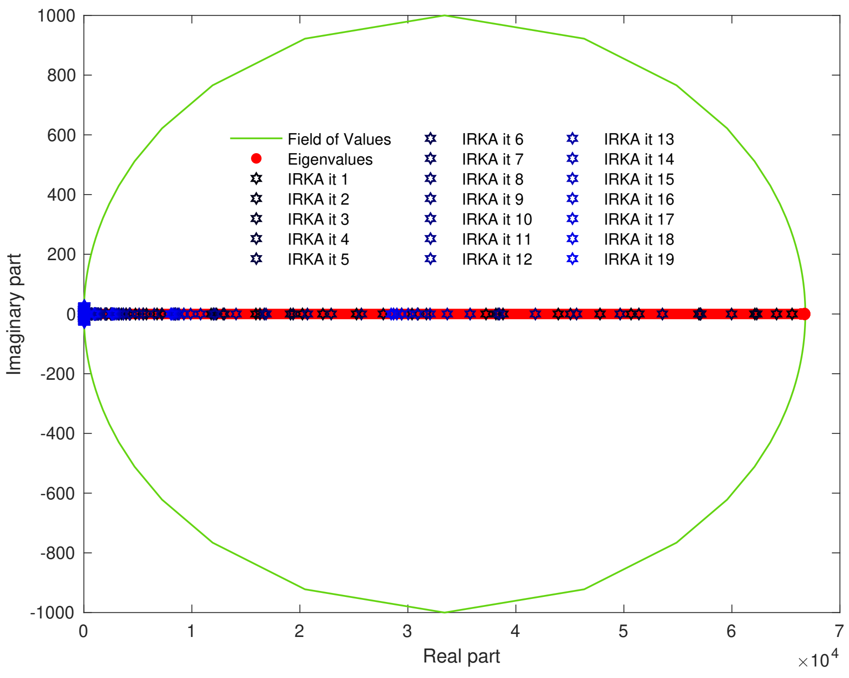
Zolotarev

Recall that we are interested in solving Lyapunov equations arising from convection–diffusion PDEs which are moderately convective (as in Figure 4.1(b)), but which still have coefficient matrices with real spectrum. For such matrices, we have seen that the rational approximation of the solution still leads to a relaxed minimax problem (4.4), corresponding to the third Zolotarev problem. This means that the solution to this minimax problem is a set of poles which can be computed using elliptic integrals, using the code by Sabino from [74]. Furthermore, the examples presented in [27] suggest that for coefficient matrices with real spectra, a set of real poles is appropriate. This motivates the use of the solution to the real Zolotarev problem, as described in Section 3.5.1 for the diffusion problem.

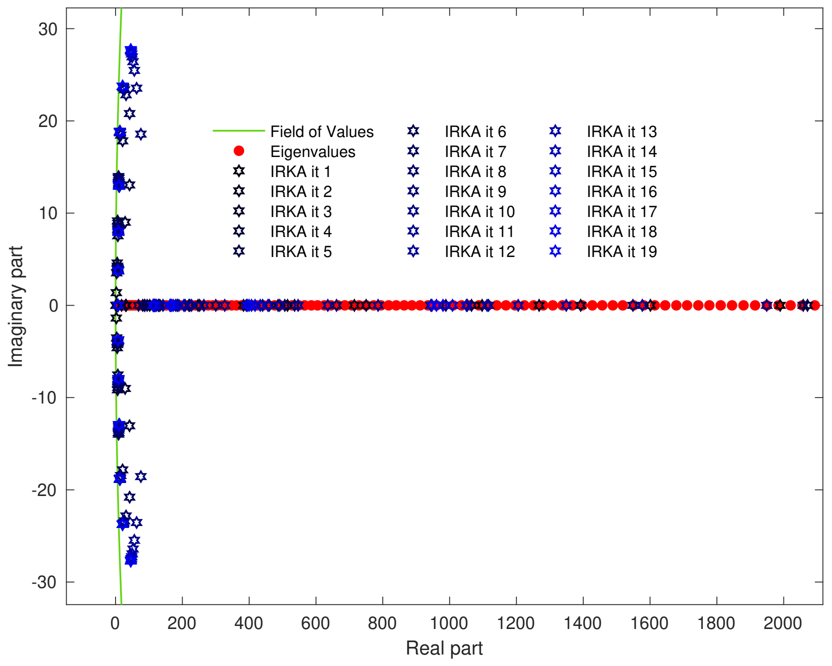
IRKA

The convection–diffusion problems we are interested in yield real-valued nonnormal coefficient matrices. Even though these matrices have real spectra, the nonnormality of the matrix and the large eigenvector condition number suggest that Zolotarev poles, which only use information about the spectral interval might not be optimal in this case. Accordingly, we are interested in a pole-generation approach which takes into account more information about the problem at hand, such as the Ritz values and the right-hand side. This suggests that the IRKA approach from Algorithm 5 might prove useful for this type of Lyapunov equation.

When we presented the two upper bounds in Theorems 4.2.1 and 4.2.2, we stated that for the nonnormal matrices that arise in the discretisation of convection-



(a) IRKA pole distribution at each iteration.



(b) Closer image of leftmost IRKA poles at each iteration.

Figure 4.2: IRKA pole distribution at each IRKA iteration, eigenvalues and the field of values for a convection–diffusion problem of size $n = 1000$, with $\epsilon = 0.0167$, and right-hand side vector of ones.

dominated PDEs, the field of values bound may be more appropriate than the eigenvector bound, which depends on the large eigenvector condition number. Moreover, we can show that the poles resulting from the IRKA algorithm lie in the field of values of the coefficient matrix M, indicating that the IRKA poles may be better suited to generate the appropriate rational Krylov subspace in which to seek a solution to the convection–diffusion Lyapunov equation.

Recall from Definition 2.1.5 that the field of values is the set of all scalars $\{x^*Mx : x \in \mathbb{R}^n, x^*x = 1\}$. We will now show that the poles that we get from Algorithm 5 lie in the field of values. Let $M_k = V^TMV$, let q_i be a normalised eigenvector of M_k, i.e., $\|q_i\|_2 = 1$, and let ρ_i be an eigenvalue of M_k. Furthermore, recall that the IRKA poles are Ritz values, ρ_i. Then

$$\rho_i = q_i^T M_k q_i = (Vq_i)^T M (Vq_i) = x_i^T M x_i,$$

with $x_i = Vq_i$ and $\|x_i\|_2 = \|Vq_i\|_2 = \|q_i\|_2 = 1$. Therefore,

$$\rho_i = x_i^T M x_i \in W(M). \tag{4.10}$$

This shows that the poles that we get from Algorithm 5 lie in the field of values.

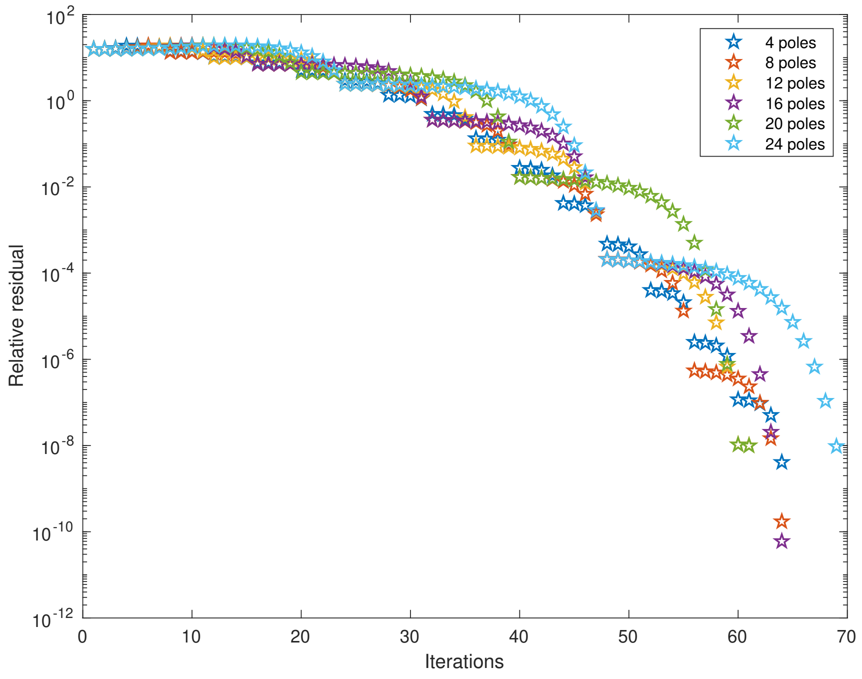
To see where the IRKA poles lie in practice, we can plot them at each iteration on the same graph as the field of values. This is presented in Figure 4.2 for a convection–diffusion problem of size $n = 1000$, with $\epsilon = 0.0167$ and a right-hand side vector of ones.

We can see from Figure 4.2 that the IRKA poles for this nonnormal matrix are complex and lie in the field of values. Furthermore, we notice that the imaginary parts of the complex poles are relatively small. This is because they are near the boundary of the field of values, so given that they have small real part, they cannot have large imaginary part. Moreover, the IRKA poles are clearly covering the entire spectrum of the matrix, hence providing a promising approach. In the next section we will see how well the IRKA poles perform in practice.

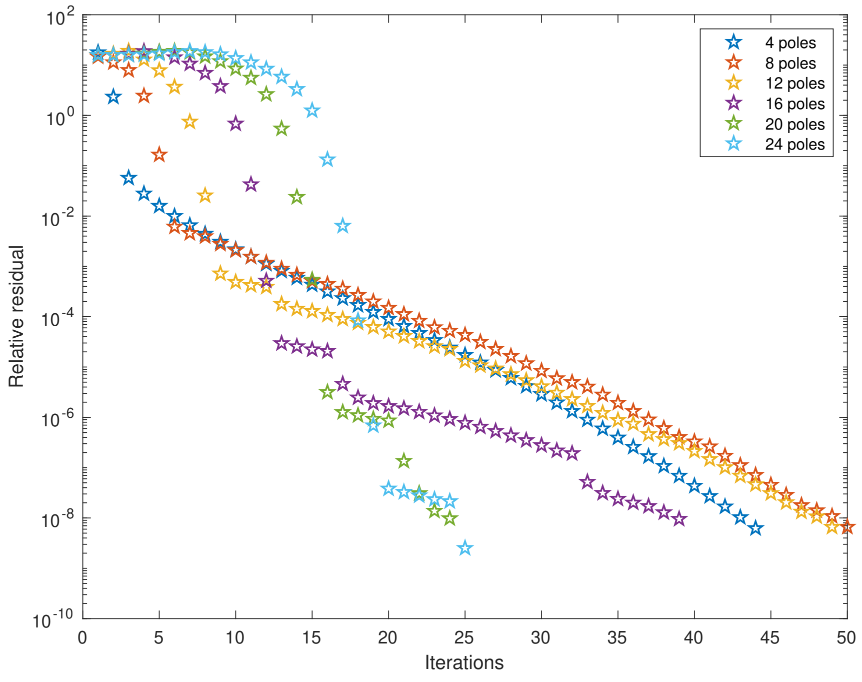
4.2.5 Numerical results

In this section we present numerial results focusing on iteration counts and computation times for Lyapunov equations arising from the discretisation of convection–diffusion problems. These will be given by $MX + XM^T = F$, with $M = -\epsilon A + B$, where A and B are as in (2.10) and (2.13), respectively. We consider three small values of ϵ which will place us in similar scenarios to Figure 4.1(b), where we have real spectrum but fairly nonnormal matrices. We will compare the IRKA poles (which are complex, as in Figure 4.2) and Zolotarev poles (which are real since the spectrum of M is real). We compare our results for problem size $n = 1000$ and a right-hand side choice of ones at relative residual tolerances given by $\tau = 10^{-4}$ and $\tau = 10^{-8}$. Note that due to the nonnormality of the matrices M, the spectral interval cannot be easily computed using built-in `MATLAB` functions so, as the matrices are tridiagonal Toeplitz, then we use the exact computation using (2.1). In Figure 4.3 we show the convergence based on the number of poles used for a test problem with $\epsilon = 0.0167$.

We can see clearly from Figure 4.3 that a large number of IRKA poles can give better convergence than the Zolotarev poles, which, despite using different numbers of poles, tend to require similar number of iterations for convergence of the solver. In order to perform a computational time comparison we choose to look



(a) Convergence for different numbers of Zolotarev poles.



(b) Convergence for different numbers of IRKA poles.

Figure 4.3: Convergence for different numbers of Zolotarev and IRKA poles for $\epsilon = 0.0167$.

at 20 poles of each kind. We present our computation times, as well as iteration numbers in Table 4.1.

ϵ	Poles	Time poles	$\tau = 10^{-4}$		Total time	$\tau = 10^{-8}$		Total time
			Iter.	Time		Iter.	Time	
0.0333	Zolotarev	0.0216	39	0.4935	0.5151	53	0.6059	0.6275
	IRKA	0.1601	16	0.2786	0.4387	22	0.3182	0.4783
0.0167	Zolotarev	0.0248	58	0.7099	0.7347	61	0.7104	0.7352
	IRKA	0.2130	15	0.2155	0.4285	24	0.3810	0.5940
0.0083	Zolotarev	0.0373	79	1.1176	1.1549	97	1.4718	1.5091
	IRKA	0.2940	16	0.3228	0.6168	37	0.6846	0.9786

Table 4.1: Summary of pole choices for Lyapunov equations with nonnormal coefficient matrices. Total time columns represents the pole time and solver time added together at $\tau = 10^{-4}$ and $\tau = 10^{-8}$, respectively.

We can see in Table 4.1 that the complex IRKA poles perform much better than the real Zolotarev ones in terms of both iteration numbers and convergence time, at both 10^{-4} and 10^{-8} residual tolerances. Even though the time to generate the IRKA poles is higher than that required to generate the Zolotarev poles, this extra work is insignificant when we consider the overall time required to solve the Lyapunov equations. Furthermore, as ϵ decreases, we can see that both the Zolotarev and IRKA poles require more iterations to converge at $\tau = 10^{-8}$, while at $\tau = 10^{-4}$, only the Zolotarev poles need more iterations, the IRKA poles requiring a nearly constant number of iterations, making them optimal in this sense. This makes the complex IRKA poles favourable over the Zolotarev ones for problems of this type.

4.3 Sylvester equation

When describing the types of matrix equations that arise from the discretisation of convection–diffusion PDEs we said that when the convection wind is either $w = (w_1, w_2) = (\phi_1(x), \psi_2(y))$ or $w = (w_1, w_2) = (\psi_1(y), \phi_2(x))$, we obtain a Sylvester equation of the form

$$-\epsilon(AX + XA^T) + \Phi_1 BX + XB^T\Psi_2 = F. \tag{4.11}$$

In this section, we discuss the projection method, algorithm, pole choices and numerical results for solving such Sylvester equations.

Similarly to the Lyapunov equation in Section 4.2, we combine the matrices on the left-hand side of X into a matrix M and the matrices on the right-hand side into a matrix N, so that our Sylvester equation is

$$MX + XN = F, \tag{4.12}$$

with $M = -\epsilon A + \Phi_1 B$ and $N = -\epsilon A^T + B^T\Psi_2$.

4.3.1 Projection

Our approach to solving the Sylvester equation in (4.11) uses projections into rational Krylov subspaces. As we have two different matrices which are very likely to have different characteristics, we will use two different bases and a Petrov–Galerkin orthogonality condition to obtain a reduced problem. Projection methods for the Sylvester equation are discussed in the literature in, e.g., [67, 78]. For simplicity of exposition, we use a symmetric, rank-1 right-hand side given by $F = bb^T$. Note that other rank-1 and higher-rank options are possible here.

Suppose we have two orthonormal bases for approximation spaces of, for simplicity of exposition, equal dimensions k, with $k \ll n$. Then, we can collect all the basis vectors generated with coefficient matrix M into a matrix V of size $n \times k$, and all the basis vectors generated with coefficient matrix N into a matrix W of size $n \times k$. Then, we seek an approximate solution to the Sylvester equation (4.12) of the form $X_k = VY_kW^T \approx X$, where Y_k is found by insisting the residual $R = MX + XN - F$ satisfies the Petrov–Galerkin orthogonality condition, given by

$$V^TRW = 0. \tag{4.13}$$

By replacing X by X_k in the residual and applying (4.13), we obtain the projected Sylvester equation

$$M_kY_k + Y_kN_k = b_Vb_W^T, \tag{4.14}$$

where $M_k = V^TMV$, $N_k = W^TNW$, $b_V = V^Tb$, and $b_W = W^Tb$. Since both V and W have orthonormal columns, $V^TV = W^TW = I$. The reduced problem is small enough that it can be efficiently solved using the Bartels–Stewart `MATLAB` function `lyap`. Having obtained Y_k, we then compute low rank factors of the solution X_k by using an SVD of Y_k. We describe this projection method in Algorithm 6.

Notice that in step 1 of Algorithm 6 we use as input two sets of poles which we use to generate the two bases, together with the matrices M and N, respectively. We mention here that, in practice, the two sets of poles can be the same, with $\mathcal{S}_1 = \mathcal{S}_2$, so that the only influence in the basis is given by one of the coefficient matrices. This approach of using the same set of poles for both bases is useful

Algorithm 6 Sylvester equation projection

```
1: INPUT: $M, N, b, \mathcal{S}_1 = \{s_1, s_2, \dots\}, \mathcal{S}_2 = \{\tilde{s}_1, \tilde{s}_2, \dots\}$,
           maximum iterations $maxit$, tolerance $tol$.
2: Set $V = b/\|b\|, W = b/\|b\|, i = 0$.
3: while not converged and $i < maxit$ do
4:     Increase $i = i + 1$
           and expand bases using the poles: $V = getbasis(M, s_i, V)$,
                                             $W = getbasis(N, \tilde{s}_i, W)$.
5:     Project the matrices $M_k = V^T M V$,
           $N_k = W^T N W$, and the right-hand side terms
           $b_V = V^T b, b_W = W^T b$.
6:     Solve $M_k Y_k + Y_k N_k = b_V b_W^T$.
7:     Compute the factors $\tilde{X}_1, \tilde{X}_2$ from $Y_k$ so that $X_k = \tilde{X}_1 \tilde{X}_2^T$.
8:     Check convergence.
9: end while
10: OUTPUT: solution factors $\tilde{X}_1, \tilde{X}_2$, final residual.
```

when one of the coefficient matrices carries more influence than the other matrix. An example of this appears in the context of stochastic Galerkin approximation for a generalised Sylvester equation [71], where the authors multiply all terms in the equation by the inverse of the Cholesky factor of the chosen dominant SPD coefficient matrix. In our numerical tests, we will not use such a transformation to obtain a dominant matrix, but in some cases we will check if one of the coefficient matrices is more important by only using the poles associated with it. We will present numerical results with both $\mathcal{S}_1 = \mathcal{S}_2$ and $\mathcal{S}_1 \neq \mathcal{S}_2$, and compare the performance of the solver with both. Note that the basis generation in step 4 uses a modified Gram–Schmidt process.

4.3.2 Poles

In this section we discuss some of the pole choices that can be used for $\mathcal{S}_1$ and $\mathcal{S}_2$ in Algorithm 6. As in the case of Lyapunov equations, we focus on matrices with real spectra, so it makes sense to consider poles which are the solution to the Zolotarev problem on real intervals, as well as IRKA poles. We describe how we generate and use those poles in the following sections, and then present some numerical results in order to understand how the pole choice affects the convergence of the solver.

Zolotarev

In [71], the authors suggest that for a Sylvester equation, a set of poles given by the solution to the third Zolotarev problem could still be effective. In that paper, as described above, the authors transform the coefficient matrices so they all contain information about a chosen SPD matrix, the spectrum of which is used to find the Zolotarev poles. We do not perform such a transformation, and so, we will be computing two sets of Zolotarev poles, one using the spectral interval of M and one using the spectral interval of N. We then use each set of poles to generate the corresponding basis. In our numerical results, we will solve Sylvester equations with a single set of poles corresponding to M ($\mathcal{S}_1 = \mathcal{S}_2$), with a single set corresponding to N ($\mathcal{S}_1 = \mathcal{S}_2$), and with both sets at the same time ($\mathcal{S}_1 \neq \mathcal{S}_2$).

IRKA

As in the case of Lyapunov equations, we consider IRKA poles as we are interested in addressing the influence of the right-hand side on the poles. In this section we present the algorithm for generating IRKA poles for the Sylvester equation. The algorithm has been adapted from [7, Alg. 1] and is presented in Algorithm 7 for rank-1 symmetric right-hand side b.

Note that the IRKA algorithm for computing two sets of poles for the Sylvester equation outputs poles which are related to one another. This is because of the way in which we update the poles: for matrix M we use the eigenvalues of the

Algorithm 7 IRKA poles for Sylvester equation

```
1: INPUT: coefficient matrices $M, N$, right-hand side factor $b$,
          initial poles $\mathcal{S}_1 = \{s_1, \ldots, s_k\}, \mathcal{S}_2 = \{\tilde{s}_1, \ldots, \tilde{s}_k\}$, tolerance $tol$.
2: while relative change in $s_i > tol$ and $\tilde{s}_i > tol$ do
3:     Compute orthonormal $V, W$ so that
           $\text{span}\{V\} = \text{span}_{i=1,\ldots,k}\{(s_i I + M)^{-1} b\}$,
           $\text{span}\{W\} = \text{span}_{i=1,\ldots,k}\{(\tilde{s}_i I + N)^{-1} b\}$.
4:     Project $M_k = V^T M V, N_k = W^T N W$.
5:     Compute eigenvalues of $M_k : \Lambda$, and $N_k : \Sigma$.
6:     Update $s_i = \text{diag}(\Sigma), \tilde{s}_i = \text{diag}(\Lambda)$.
7: end while
8: OUTPUT: new poles $s_i, \tilde{s}_i$.
```

projected matrix N_k, and similarly, for the poles corresponding to matrix N we use the eigenvalues of the projected matrix M_k. This is then updated in step 6 of Algorithm 7. The convergence of Algorithm 6 will greatly benefit from these sets of IRKA poles, as we will show in the numerical results. We will compare the solver performance with IRKA poles generated with M only, N only, and the approach in Algorithm 7 which uses information about both coefficient matrices.

4.3.3 Numerical results

In this section we present some numerical results which compare pole choices for the Sylvester equation obtained by discretising the convection–diffusion equations. We will show results for three convection winds of the form $w(\phi_1(x), 0)$, $w(\phi_1(x), 1)$ and $w(\phi_1(x), \psi_2(y))$ and three different values of ϵ: 0.0333, 0.0167, and 0.0083 [63]. In our numerical results, we will use the stopping criterion given by $\frac{\|R\|_F}{\|bb^T\|_F} < \tau$, where $R = MX_k + X_k N - bb^T$, for the two tolerances $\tau = 10^{-4}$ and $\tau = 10^{-8}$.

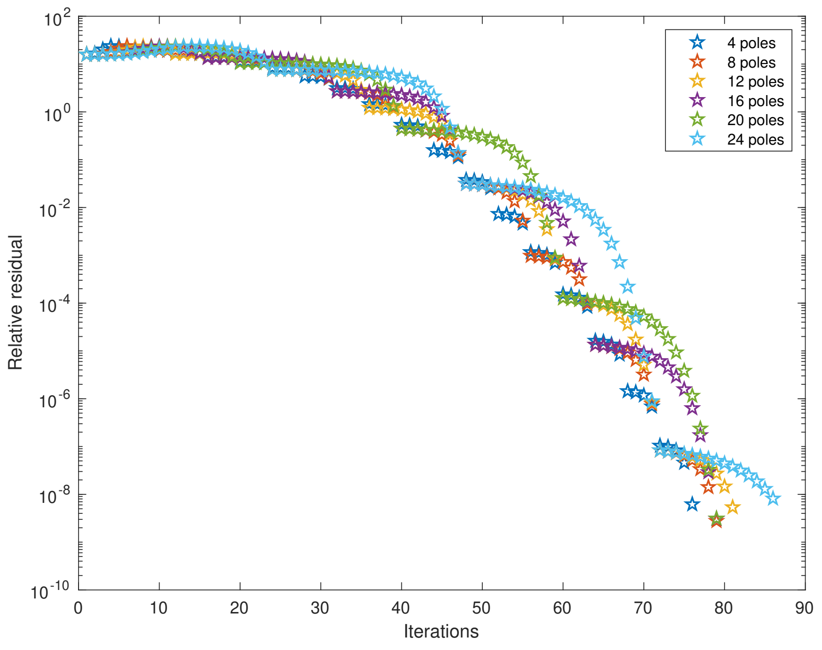
Example 4.1.

In this example, we show results for the convection–diffusion problem with wind $w = (1 + (x + 1)^2/4, 0)$. This leads to a Sylvester equation $MX + XN = F$ with $M = -\epsilon A + \Phi_1 B$, $N = -\epsilon A$, where $\Phi_1 = \text{diag}(\phi_1(x_i))$, $\phi_1(x) = 1 + (x + 1)^2$, and x_i are linearly spaced values between 0 and 1. The right-hand side $F = bb^T$ is rank-1, with b a vector of ones.

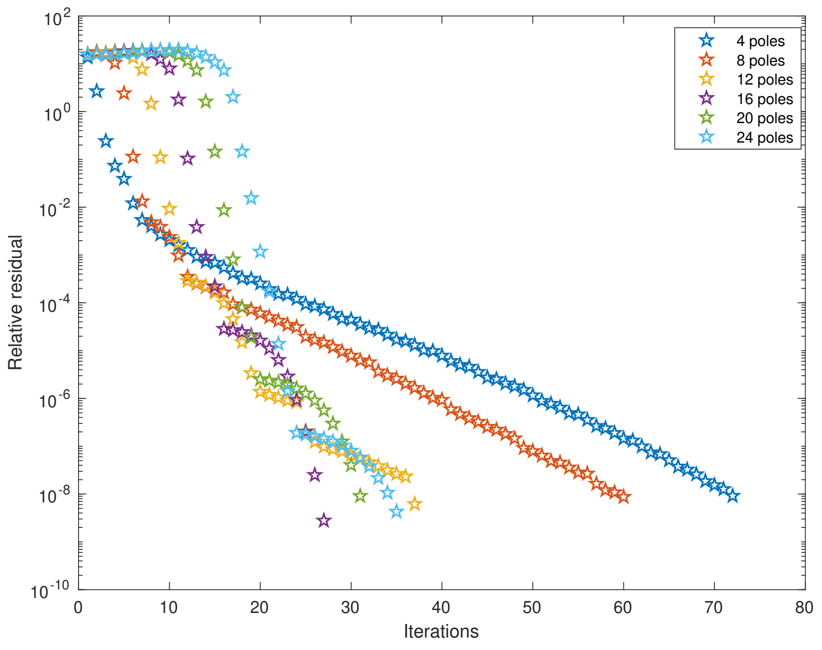
As in the Lyapunov equation case, we begin by comparing the convergence for different numbers of Zolotarev and IRKA poles. Note that for both, we use two sets of poles, one generated with M and one with N. These results are presented in Figure 4.4.

From the convergence plots in Figure 4.4, we can clearly see that the number of Zolotarev poles does not strongly influence the number of iterations required by the solver to generate a solution, however, in the case of IRKA poles, it is clear that 16 poles outperform the other choices. Therefore, in our remaining numerical tests for this problem, we will use 16 poles to compare the iterations and times required to solve the Sylvester equation.

We notice in Table 4.2 that both the Zolotarev poles and the IRKA poles generated with N show similar solver convergence times and iteration counts for $\epsilon = 0.0333$ and $\epsilon = 0.0167$, but that the IRKA poles allow for quicker convergence than the Zolotarev poles when generated with only M or both matrices M and N. However, for $\epsilon = 0.0083$, the solver using IRKA poles requires fewer iterations and time to converge than Zolotarev at both $\tau = 10^{-4}$ and $\tau = 10^{-8}$, indicating that for highly convective Sylvester equations of this type, the IRKA poles might be favourable.



(a) Convergence for different numbers of Zolotarev poles.



(b) Convergence for different numbers of IRKA poles.

Figure 4.4: Convergence for different numbers of Zolotarev and IRKA poles for $\epsilon = 0.0167$, for the Sylvester equation $MX + XN = F$ with $M = -\epsilon A + \Phi_1 B$, $N = -\epsilon A$, and $w = (1 + (x+1)^2/4, 0)$.

ϵ	Poles	Matrices	Time poles	$\tau = 10^{-4}$		Total time	$\tau = 10^{-8}$		Total time
				Iter.	Time		Iter.	Time	
0.0333	Zolotarev	M only	0.0146	48	1.2421	1.2567	62	1.6258	1.6404
		N only	0.0156	16	0.4138	0.4294	28	0.7012	0.7168
		both M, N	0.0166	48	1.2442	1.2608	62	1.7357	1.7523
	IRKA	M only	0.1579	14	0.3234	0.4813	33	0.8417	0.9996
		N only	0.0239	16	0.4087	0.4326	28	0.6436	0.6675
		both M, N	0.1127	16	0.4589	0.5716	27	0.6517	0.7644
0.0167	Zolotarev	M only	0.0164	64	1.7474	1.7638	79	2.3067	2.3231
		N only	0.0154	16	0.4279	0.4433	28	0.6732	0.6886
		both M, N	0.0169	64	1.8382	1.8551	79	2.2578	2.2747
	IRKA	M only	0.1975	13	0.3550	0.5525	40	1.1027	1.3002
		N only	0.0248	16	0.3724	0.3972	28	0.6590	0.6838
		both M, N	0.1464	16	0.4599	0.6063	27	0.6980	0.8444
0.0083	Zolotarev	M only	0.0178	73	2.2540	2.2718	88	2.9324	2.9502
		N only	0.0158	20	0.6045	0.6203	29	0.8261	0.8419
		both M, N	0.0183	73	2.6287	2.6470	88	2.9127	2.9310
	IRKA	M only	0.2546	13	0.3851	0.6397	43	1.3726	1.6272
		N only	0.0227	16	0.4004	0.4231	28	0.7493	0.7664
		both M, N	0.1654	16	0.4582	0.6236	28	0.8538	1.0192

Table 4.2: Iterations and CPU time for Sylvester equations obtained with wind $w = (w_1, w_2) = (1 + (x+1)^2/4, 0)$. Total time columns represents the pole time and solver time added together for $\tau = 10^{-4}$ and $\tau = 10^{-8}$, respectively.

From the numerical results in Table 4.2, we note that having poles related to only one matrix is typically preferable to having poles associated with each of the two coefficient matrices. What is particularly interesting here is that the coefficient matrix that allows us to obtain superior numerical results is the scaled Poisson matrix, N. This leads us to believe that in this particular case, there exists a matrix which 'dominates' the Sylvester equation, i.e., A. This makes sense here, as the Poisson matrix A appears in both M and N.

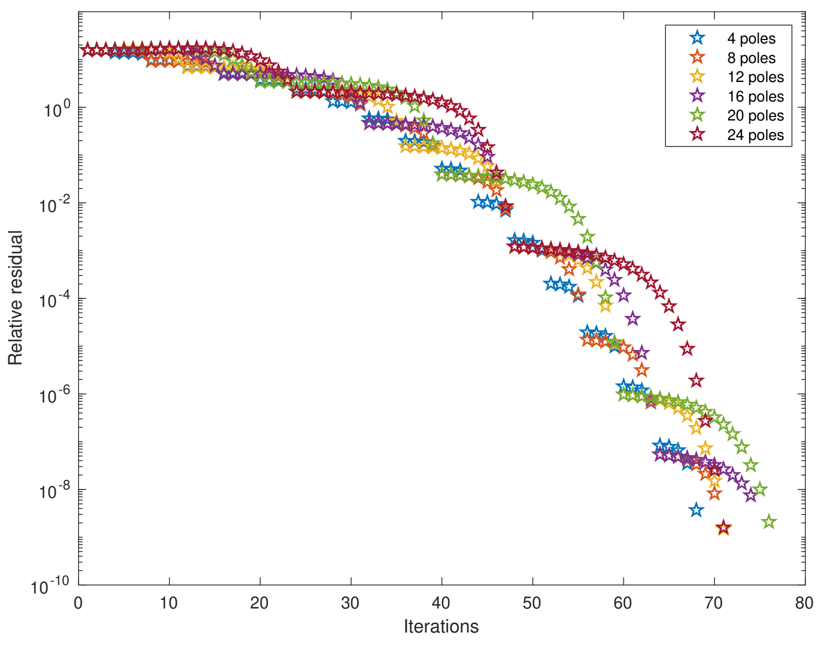
Example 4.2.

In this example, we show results for the convection–difusion problem with wind given by $w = (1 - x^2, 1)$, i.e., a wind dictated by a function of x in one direction, and constant wind in the other. Our Sylvester equation then takes the form $MX + XN = F$, with $M = -\epsilon A + \Phi_1 B$, and $N = -\epsilon A + B^T$, where, as before, $\Phi_1 = \text{diag}(\phi_1(x_i))$, $\phi_1(x) = 1 - x^2$ and x_i are linearly-spaced values in the interval $(0, 1)$. The right-hand side $F = bb^T$ is rank-1 symmetric, with b given by a vector of ones.

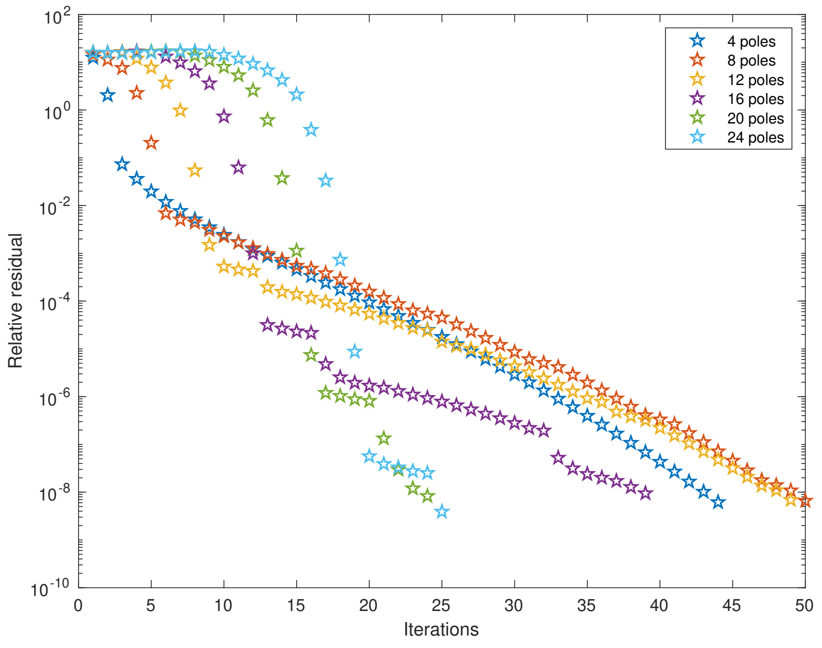
In Figure 4.5, we again compare the convergence for different numbers of poles for both the Zolotarev and IRKA approaches. Note that for both, we use two sets of poles, one generated with M and one with N. We see that the number of Zolotarev poles does not strongly influence the number of iterations required by the solver to generate a solution, however, 20 IRKA poles show better convergence than other numbers of poles. Therefore, for this problem we will use 20 poles to compare the iterations and times required to solve the Sylvester equation.

ϵ	Poles	Matrices	Time poles	$\tau = 10^{-4}$		Total time	$\tau = 10^{-8}$		Total time
				Iter.	Time		Iter.	Time	
0.0333	Zolotarev	M only	0.0207	36	0.7377	0.7584	39	0.7764	0.7971
		N only	0.0233	40	0.8116	0.8349	56	1.1089	1.1322
		both M, N	0.0239	40	0.7563	0.7802	56	1.1018	1.1257
	IRKA	M only	0.2616	16	0.4876	0.7492	18	0.5313	0.7929
		N only	0.2983	16	0.4549	0.7532	18	0.4945	0.7940
		both M, N	0.2848	16	0.3835	0.6683	22	0.4717	0.7565
0.0167	Zolotarev	M only	0.0173	38	0.7162	0.7335	55	1.0775	1.0948
		N only	0.0201	59	1.2061	1.2262	76	1.6309	1.6510
		both M, N	0.0224	59	1.2207	1.2431	76	1.6751	1.6975
	IRKA	M only	0.3939	16	0.4721	0.8660	17	0.4774	0.8713
		N only	0.3619	16	0.4642	0.8261	18	0.4842	0.8461
		both M, N	0.3954	16	0.3921	0.7875	24	0.5334	0.9288
0.0083	Zolotarev	M only	0.0209	48	1.0102	1.0311	59	1.1664	1.1873
		N only	0.0146	71	1.5189	1.5335	80	2.0628	2.0774
		both M, N	0.0189	71	1.6311	1.6500	80	1.8719	1.8908
	IRKA	M only	0.5355	15	0.4391	0.9746	22	0.6426	1.1781
		N only	0.4760	16	0.4648	0.9408	21	0.5884	1.0644
		both M, N	0.5020	16	0.4554	0.9574	40	1.0985	1.6005

Table 4.3: Iterations and CPU times for Sylvester equations obtained with wind $w = (w_1, w_2) = (1-x^2, 1)$. Total time columns represents the pole time and solver time added together for $\tau = 10^{-4}$ and $\tau = 10^{-8}$, respectively.



(a) Convergence for different numbers of Zolotarev poles.



(b) Convergence for different numbers of IRKA poles.

Figure 4.5: Convergence for different numbers of Zolotarev and IRKA poles for $\epsilon = 0.0167$, for the Sylvester equation $MX + XN = F$ with $M = -\epsilon A + \Phi_1 B$, $N = -\epsilon A + B^T$, and $w = (1 - x^2, 1)$.

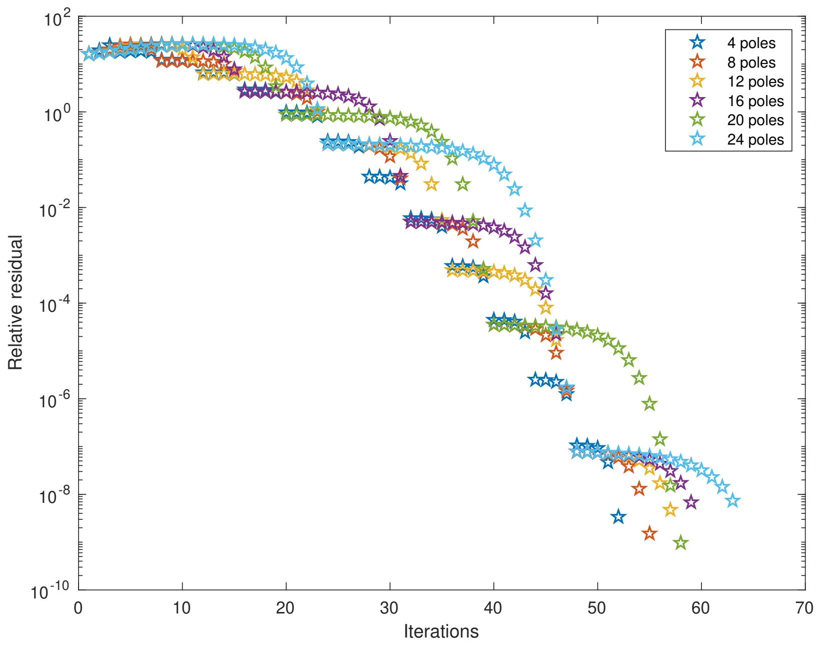
We can see from Table 4.3 that despite the increased cost of generating the IRKA poles, they perform better in the Sylvester solver than Zolotarev poles both at residual tolerance levels of $\tau = 10^{-4}$ and $\tau = 10^{-8}$. In the case of $\epsilon = 0.0333$, having the poles generated with each of the two coefficient matrices proves quicker than having only one set of poles corresponding to only one coefficient matrix. In the remaining setups, this is not the case, with lower iteration numbers and CPU times corresponding to the poles generated using the matrix N. This, as before, may be because N is a combination of the matrices A and B which both appear in M as well, but slightly modified.

Another observation we make from Table 4.3 is related to the effect of ϵ. To be precise, we can see that as ϵ decreases and our problem becomes more convection dominated, the time and number of iterations required to solve the Sylvester equation grow.

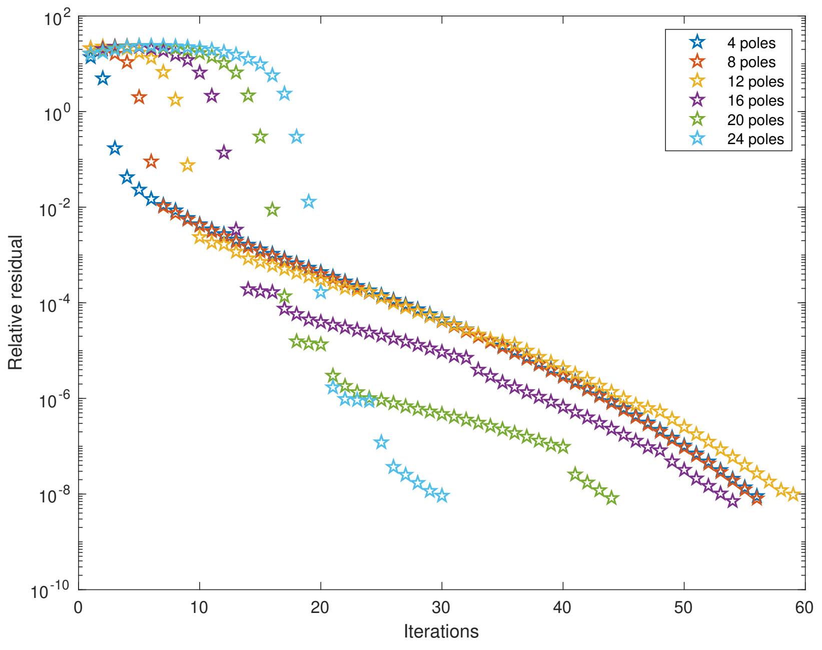
Example 4.3.

In this example, we look at the matrix equation arising from the convection–diffusion problem with wind $w = (1 - (2x+1)^2, y)$. This leads to the Sylvester equation $MX + XN = F$, where $M = -\epsilon A + \Phi_1 B$ and $N = -\epsilon A + B^T \Psi_2$, with $\Phi_1 = \text{diag}(\phi_1(x_i))$, $\Psi_2 = \text{diag}(\psi_2(y_i))$, $\phi_1(x) = 1 - (2x+1)^2$, $\phi_2(y) = y$ and x_i, y_i are linearly-spaced values in $(0, 1)$. As before, the right-hand side is given by $F = bb^T$, where b is a vector of ones.

We begin by comparing the convergence of our solver for different numbers of Zolotarev and IRKA poles. We do this by testing our solver with poles generated using both matrices M and N. Our convergence curves are presented in Figure 4.6. From this we see that, as in our other examples, the Zolotarev poles



(a) Convergence for different numbers of Zolotarev poles.



(b) Convergence for different numbers of IRKA poles.

Figure 4.6: Convergence for different numbers of Zolotarev and IRKA poles for $\epsilon = 0.0167$, for the Sylvester equation $MX + XN = F$ with $M = -\epsilon A + \Phi_1 B$, $N = -\epsilon A + B^T \Psi_1$, and $w = (1 - (2x + 1)^2, y)$.

do not show big differences in the number of iterations required for convergence of the solver, but the IRKA poles do, with 24 poles achieving the smallest iteration count. As a result, we will use 24 poles in the numerical tests we perform for this problem. These results are generated, as before, for three values of ϵ, namely, 0.0333, 0.0167 and 0.0083, and we compare the convergence results for poles generated with only M, only N and with both coefficient matrices.

ϵ	Poles	Matrices	Time poles	$\tau = 10^{-4}$		Total time	$\tau = 10^{-8}$		Total time
				Iter.	Time		Iter.	Time	
0.0333	Zolotarev	M only	0.0252	45	2.0943	2.1195	48	2.2854	2.3106
		N only	0.0249	23	0.9943	1.0192	41	1.7578	1.7827
		both M, N	0.0215	45	2.0195	2.0410	48	2.0671	2.0886
	IRKA	M only	0.6667	20	1.0900	1.7567	25	1.2977	1.9644
		N only	0.2309	19	0.9512	1.1821	20	0.9764	1.2073
		both M, N	0.4030	20	0.9436	1.3466	26	1.1972	1.6002
0.0167	Zolotarev	M only	0.0141	46	2.0874	2.1015	63	2.9741	2.9882
		N only	0.0151	23	1.1511	1.1662	42	1.9910	2.0061
		both M, N	0.0176	46	2.2410	2.2586	63	3.1595	3.1771
	IRKA	M only	0.4379	21	1.0924	1.5303	54	3.0506	3.4885
		N only	0.1068	18	0.8483	0.9551	29	1.3779	1.4847
		both M, N	0.6055	21	1.0234	1.6289	30	1.4615	2.0670
0.0083	Zolotarev	M only	0.1756	23	1.2320	1.4076	39	2.1223	2.2979
		N only	0.0157	23	0.9785	0.9942	42	1.8706	1.8863
		both M, N	0.0192	23	1.0994	1.1186	42	2.0181	2.0373
	IRKA	M only	0.7909	41	2.3239	3.1148	44	2.5830	3.3739
		N only	0.2637	33	1.5640	1.8277	36	1.6267	1.8904
		both M, N	1.0207	41	2.1744	3.1951	43	2.2421	3.2628

Table 4.4: Iterations and CPU times for Sylvester equations obtained with wind $w = (w_1, w_2) = (1 - (2x+1)^2, x)$. Total time columns represents the pole time and solver time added together for $\tau = 10^{-4}$ and $\tau = 10^{-8}$, respectively.

We can see from Table 4.4 that, as in Examples 4.1 and 4.2, the times required to generate the IRKA poles are higher than those for the Zolotarev poles, and that in the cases of $\epsilon = 0.0333$ and $\epsilon = 0.0167$ the IRKA poles require fewer solver iterations to converge. This means that our best solver times occur for the IRKA poles generated with the matrix N. For $\epsilon = 0.0083$, our most convective problem, we can see that both IRKA and Zolotarev poles require similar solver times to converge, but that the IRKA poles lead to fewer solver iterations. This means that the IRKA poles could still be considered superior.

We can see that in Examples 4.1–4.3, using two sets of poles is not as efficient in terms of both iterations and times as using the poles generated with the matrix N only. Based on the results we have seen in these three examples, we can state that using IRKA poles with N only shows the best iteration counts and convergence times. We notice that the matrix N has usually been chosen so that the functions associated with the convection wind are easier to compute. Recall that in Example 4.1, this was 0, in Example 4.2, this was a constant, 1, and that in Example 4.3, it was y. Therefore, we are in a position to make a recommendation as to which poles to use for Sylvester equations arising from convection–diffusion problems. That is, the IRKA poles, generated with one matrix only (the one associated with the convection wind function which resembles a constant), seems to result in faster convergence rates in most cases.

4.4 Generalised Sylvester equation

We have seen that discretising the convection–diffusion PDE can lead to three different matrix equations. In Section 4.2 we presented the resulting Lyapunov equation, while in Section 4.3 we discussed the Sylvester equation. In this section, we discuss solvers for (2.14), the last possible type of matrix equation arising from the discretisation of the convection–diffusion PDE, which was presented in Section 2.2.2. The equation (2.14) is a particular case of the following matrix equation:

$$M_0X + XM_0 + M_1XN_1 + M_2XN_2 = F, \tag{4.15}$$

where, in our problem, $M_0 = -\epsilon A$ is SPD, $M_1 = \Phi_1 B$, $M_2 = \Phi_2$, $N_1 = \Psi_1$ and $N_2 = B^T \Psi_2$. We call (4.15) a generalised Sylvester equation and recall that it is a special case of the more general multi-term matrix equation [49]

$$AX + XB + \sum_{i=1}^{m} M_i X N_i = F.$$

We introduce two linear operators, namely $\mathcal{M} : \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$, and $\mathcal{N} : \mathbb{R}^{n\times n} \to \mathbb{R}^{n\times n}$ such that

$$\begin{aligned} \mathcal{M}(X) &:= M_0 X + X M_0, \\ \mathcal{N}(X) &:= -(M_1 X N_1 + M_2 X N_2), \end{aligned} \tag{4.16}$$

with corresponding Kronecker forms given, respectively, by

$$\begin{aligned} M &= I \otimes M_0 + M_0^T \otimes I, \\ N &= -(N_1^T \otimes M_1 + N_2^T \otimes M_2). \end{aligned} \tag{4.17}$$

Then, we can write (4.15) as

$$\mathcal{M}(X) - \mathcal{N}(X) = F. \tag{4.18}$$

Note that in the context of PDE problems, matrix M_0 is large, sparse and nonsingular, and the operator $\mathcal{M}$ is assumed to be invertible [49].

4.4.1 Stationary iterative method

We will now use a stationary iterative method for solving (4.15). Shank, Simoncini and Szyld present such a stationary approach for the generalised Lyapunov equation [75], but the method cannot be applied to our matrix equation as is, since in our problem $\mathcal{N}$ is nonsymmetric. However, we can show that many of the properties exploited in [75] still arise in our problem.

As for linear systems, a fixed point (stationary) iterative method requires a stationary splitting. In our matrix equation case, we use (4.18) to give the splitting

$$\mathcal{M}(X) = \mathcal{N}(X) + F. \tag{4.19}$$

It is straightforward to show that $\mathcal{M}$ and $\mathcal{N}$ are linear and furthermore, that $\mathcal{M}(X) = Y$ is equivalent to

$$M \operatorname{vec}(X) = (M_0 \otimes I + I \otimes M_0) \operatorname{vec}(X) = \operatorname{vec}(Y).$$

The fixed point iteration is then defined as

$$X_{k+1} = \mathcal{T}(X_k), \text{ where } \mathcal{T}(X) := \mathcal{M}^{-1}\mathcal{N}(X) + \mathcal{M}^{-1}F. \tag{4.20}$$

We want to determine when this fixed point iteration is asymptotically convergent. To do this, let $\mathbb{R}^0 = \mathbb{R}^{n\times n} - \{0\}$ and let $\mathcal{B}$ be the real Banach space of $n \times n$ matrices, equipped with the matrix operator norm, defined by

$$\|\mathcal{M}\| = \sqrt{\sup_{X\in\mathbb{R}^0} \frac{\langle \mathcal{M}(X), \mathcal{M}(X)\rangle_F}{\langle X, X\rangle_F}}. \tag{4.21}$$

Lemma 4.4.1 ([30])**.** *For the mapping* $\mathcal{M}$ *from* (4.16)*, and its corresponding Kronecker form* M *from* (4.17)*, we have*

$$\|\mathcal{M}\| = \|M\|_2.$$

Proof. Using the definition of $\|\mathcal{M}\|$ from (4.21), we have that

$$\begin{aligned}\|\mathcal{M}\|^2 &= \sup_{X\in\mathbb{R}^0} \frac{\langle \mathcal{M}(X), \mathcal{M}(X)\rangle_F}{\langle X, X\rangle_F} \\ &= \sup_{X\in\mathbb{R}^0} \frac{\|\mathrm{vec}(\mathcal{M}(X))\|_2^2}{\|\mathrm{vec}(X)\|_2^2} \\ &= \sup_{X\in\mathbb{R}^0} \frac{\|M\,\mathrm{vec}(X)\|_2^2}{\|\mathrm{vec}(X)\|_2^2} \\ &= \|M\|_2^2.\end{aligned}$$

□

Similarly, we can show that the operator norm corresponding to the operator $\mathcal{N}$ is equivalent to the 2-norm of the corresponding Kronecker formulation, i.e., $\|\mathcal{N}\| = \|N\|_2$. We can now describe the convergence of the fixed point iteration from (4.20).

Lemma 4.4.2. *Let* $\mathcal{M}, \mathcal{N}$ *be as in* (4.16) *and* M, N *as in* (4.17)*. Then, the fixed point iteration from* (4.20) *converges for any initial guess* X_0 *if*

$$\|\mathcal{M}^{-1}\mathcal{N}\| = \|M^{-1}N\|_2 < 1. \tag{4.22}$$

Proof. In the Banach space, $\mathcal{B}$, for all $X, Y \in \mathbb{R}^{n\times n}$ we have that

$$\|\mathcal{T}(X) - \mathcal{T}(Y)\| = \|\mathcal{M}^{-1}(\mathcal{N}(X-Y))\| \le \|\mathcal{M}^{-1}\mathcal{N}\|\|X-Y\|, \tag{4.23}$$

so that

$$\|X_{k+1}-X_k\| = \|\mathcal{T}(X_k)-\mathcal{T}(X_{k-1})\| \le \|\mathcal{M}^{-1}\mathcal{N}\|\|X_k-X_{k-1}\| \le \|\mathcal{M}^{-1}\mathcal{N}\|^k\|X_1-X_0\|. \tag{4.24}$$

Then, by the Banach fixed point theorem, the iteration must converge for any initial approximation X_0 if $\|\mathcal{M}^{-1}\mathcal{N}\| < 1$.

We now show that $\|\mathcal{M}^{-1}\mathcal{N}\| = \|M^{-1}N\|_2$. Consider the splitting in (4.19). Since $\mathcal{M}$ is an invertible operator and since M is an invertible matrix, the solution of this matrix equation at iteration $k+1$ is given by

$$X_{k+1} = \mathcal{M}^{-1}\mathcal{N}(X_k) + \mathcal{M}^{-1}F,$$

which is equivalent to

$$\mathcal{M}(X_{k+1}) = \mathcal{N}(X_k) + F.$$

Vectorising this gives

$$\operatorname{vec}(\mathcal{M}(X_{k+1})) = \operatorname{vec}(\mathcal{N}(X_k)) + \operatorname{vec}(F),$$

which is equivalent to

$$M \operatorname{vec}(X_{k+1}) = N \operatorname{vec}(X_k) + \operatorname{vec}(F).$$

Finally, we have the vectorised form of the solution given by

$$\operatorname{vec}(X_{k+1}) = M^{-1}N \operatorname{vec}(X_k) + M^{-1} \operatorname{vec}(F),$$

and so $\mathrm{vec}(\mathcal{M}^{-1}\mathcal{N}(X)) = M^{-1}N\,\mathrm{vec}(X)$. We can now obtain the norm of $\mathcal{M}^{-1}\mathcal{N}(X)$ as follows

$$\begin{aligned}\|\mathcal{M}^{-1}\mathcal{N}\|^2 &= \sup_{X\in\mathbb{R}^0} \frac{\langle \mathcal{M}^{-1}\mathcal{N}(X), \mathcal{M}^{-1}\mathcal{N}(X)\rangle_F}{\langle X, X\rangle_F} \\ &= \sup_{X\in\mathbb{R}^0} \frac{\langle \mathrm{vec}(\mathcal{M}^{-1}\mathcal{N}(X)), \mathrm{vec}(\mathcal{M}^{-1}\mathcal{N}(X))\rangle_2}{\langle \mathrm{vec}(X), \mathrm{vec}(X)\rangle_2} \\ &= \sup_{X\in\mathbb{R}^0} \frac{\|\mathrm{vec}(\mathcal{M}^{-1}\mathcal{N}(X))\|_2^2}{\|\mathrm{vec}(X)\|_2^2} \\ &= \sup_{X\in\mathbb{R}^0} \frac{\|M^{-1}N\,\mathrm{vec}(X)\|_2^2}{\|\mathrm{vec}(X)\|_2^2} \\ &= \|M^{-1}N\|_2^2.\end{aligned}$$

Hence, we are guaranteed to converge if $\|\mathcal{M}^{-1}\mathcal{N}\| = \|M^{-1}N\|_2 < 1$. $\square$

In order to ensure our method is memory-efficient, we consider low-rank approximations to X of the form YZ^T, with Y and Z of dimensions $n \times r, r \ll n$. Using a factorisation of the right-hand side $F = C_1C_2^T$, as in [75], we propose the fixed point stationary algorithm for the generalised Sylvester equation in Algorithm 8.

Algorithm 8 Stationary Iterations for generalised Sylvester equations

```
1: Input: Problem data $A, B, M_j, N_j, C_1, C_2$ stopping tolerance
2: Output: $Y_k, Z_k$ so that $X_k = Y_kZ_k^T$ is an approximate solution of the
   generalised Sylvester equation.
3: Solve $AX_1 + X_1B + C_1C_2^T = 0$ for $X_1 = Y_1Z_1^T$
4: for $k = 2, 3, \ldots$ do
5:     Set $C_{1k} = [M_1Y_{k-1}, \ldots, M_mY_{k-1}, C_1]$ and
           $C_{2k} = [N_1^TZ_{k-1}, \ldots, N_m^TZ_{k-1}, C_2]$
6:     Solve $AX_k + X_kB + C_{1k}C_{2k}^T = 0$ for $X_k = Y_kZ_k^T$
7:     If sufficiently accurate, stop
8: end for
```

In steps 3 and 6 of Algorithm 8 we are required to solve Sylvester equations. This can be done using a projection approach or directly, such as with the Bartels–Stewart method. Note that for our purposes, the latter is sufficient. We have tested Algorithm 8 for our matrix equation (ME), as well as a Kronecker formulation linear system (LS) equivalent of our fixed point iteration algorithm, i.e., we solve $M\,\mathrm{vec}(X_{k+1}) = N\,\mathrm{vec}(X_k) + \mathrm{vec}(F)$ iteratively. We have used a right-hand side given by the function $f(x, y) = \sin(\pi x)\cos(\pi y)$, and have used the convection wind given by $w = (2y(1 - x^2), -2x(1 - y^2))$. The maximum number of iterations was set to 300 and the relative residual convergence tolerance was set to $\tau = 10^{-8}$. The results are presented in Table 4.5.

n	ϵ	it (LS)	time (LS)	it (ME)	time (ME)	$\|M^{-1}N\|_2$
4^2	1	10	0.0059	10	0.0084	0.1645
	0.5	14	0.0037	14	0.0062	0.3291
	0.2	44	0.0036	44	0.0083	0.8227
	0.1	diverges	-	diverges	-	1.6455
16^2	1	10	0.0133	10	0.0089	0.1739
	0.5	15	0.0224	14	0.0072	0.3478
	0.2	46	0.0616	46	0.0103	0.8694
	0.1	diverges	-	diverges	-	1.7316
64^2	1	11	3.9619	11	0.0218	0.1732
	0.5	16	5.7369	16	0.0025	0.3463
	0.2	45	16.071	45	0.0103	0.8658
	0.1	diverges	-	diverges	-	1.7316
256^2	1	-	-	11	0.1976	-
	0.2	-	-	46	0.4634	-
	0.1	-	-	diverges	-	-

Table 4.5: Convergence of the stationary iterative algorithm for the generalised Sylvester equation (ME) and for the equivalent Kronecker product formulation (LS).

In Table 4.5 we can see that the convergence of the fixed point iteration in terms of number of iterations is the same for both the linear system and matrix equation. This is what we expected from Lemma 4.4.2. Note that for moderate to large n, the stationary method for the Kronecker formulation takes longer than for the matrix equation. This again highlights that solvers for matrix equations formulations may be much more efficient than equivalent linear system solvers. Recall that the stationary iterative schemes are only guaranteed to converge when

$\|M^{-1}N\|_2 < 1$. When $\epsilon = 0.1$, $\|M^{-1}N\|_2 > 1$ for all considered n, so it is not surprising that the methods diverge. Note that the problems corresponding to $\|M^{-1}N\|_2 > 1$ result in oscillatory solutions, indicating that the chosen finite difference method is not stable for these problems. Furthermore, note that the number of iterations increase as $\|M^{-1}N\|_2$ increases, as expected from (4.24). Hence, we are interested in developing more powerful solvers in the future.

4.5 Conclusions

In this chapter we looked at solving matrix equations arising from discretised convection–diffusion PDEs. We showed that these matrix equations are of three main types. To be precise, we can have two-term equations in the form of Lyapunov and Sylvester equations, and we can have four-term matrix equations, depending on the choice of convection wind.

We first considered convection–diffusion problems with constant winds, discretisations of which lead to Lyapunov equations. We showed that our matrix equations can be ill-conditioned. As a result, we used projection methods into rational Krylov subspaces to solve Lyapunov equations and derived two upper bounds for the convergence of the Lyapunov solver. As in the case of Lyapunov equations from Chapter 3, we considered the best pole choices for generating these rational Krylov subspaces, and performed numerical tests which showed the superiority of the IRKA poles for Lyapunov equations arising from convection–diffusion PDEs.

We next considered discretised convection–diffusion PDEs which take the form of Sylvester equations. These Sylvester equations result from linear and quadratic convection wind functions in only one direction, but note that they could be any functions. We presented a rational Krylov subspace solver for such matrix equations and considered Zolotarev and IRKA pole choices. In our numerical tests, we found that having one set of IRKA poles generated using the matrix corresponding to the simpler of the two wind functions led to the best convergence results. While there is a cost associated with computing the IRKA poles, the reduced iteration count required, as compared to the Zolotarev poles, more than compensated for this.

Lastly, we considered convection–diffusion PDEs which led to four-term matrix equations. These arise when the convection wind components are separable functions of both x and y. We considered a stationary iterative method and provided a criterion for the convergence of this method. Our numerical results, however, showed that as we decrease ϵ, making our PDEs more convective, it is difficult to solve these problems. We believe that a projection method into a rational Krylov subspace might prove to be a better fit for this type of problems, and advise further research in this direction.

Chapter 5

Conclusions and future work

In this thesis we obtained novel methods for efficiently solving Lyapunov and Sylvester equations arising from the discretisation of PDEs. Chapter 1 motivates the work presented in this thesis by briefly discussing the advances of numerical methods for PDEs in the literature. Chapter 2 presents some preliminary results, necessary for the analysis presented in later chapters. We also present in this chapter the discretisation of our model problems and discuss solving approaches via linear systems and matrix equations, as well as presenting a comparison of the two. We focused on two model problems: in Chapter 3 we considered diffusion PDEs, which gave Lyapunov equations, and in Chapter 4 we looked at convection–diffusion PDEs, which led to three types of matrix equations: Lyapunov equations, Sylvester equations, and four-term generalised Sylvester equations. We considered solutions to all these problems in this thesis.

In Chapter 3 we used 1- and 2-sided projections into rational Krylov subspaces to solve Lyapunov equations arising from the discretisation of diffusion PDEs. We derived explicit rational functions which describe the rational Krylov solution of the Lyapunov equation, and compared the convergence results for the two projection approaches, determining that the 2-sided projection is more suitable for the problems we solve. We then focused on the numerical study of the attainable accuracy of the 2-sided approach and compared our convergence curves with two theoretical bounds, finding that the bound in (3.22) from [4] is more representative of our convergence than the asymptotic bound from [21, Theorem 4.9].

The second part of Chapter 3 was dedicated to the study of suitable pole choices for the rational Krylov subspace. We discussed the well-known Zolotarev poles, and their logarithmic approximations, the state-of-the-art adaptive approach from [23], as well as poles derived from the IRKA approach from the field of model order reduction of dynamical systems. In Section 3.6 we presented iteration counts and convergence times for a variety of diffusion problems obtained using uniform meshes, variable diffusion coefficients, and graded and geometric meshes. We have found that for uniform meshes, variable diffusion coefficients and the graded mesh, 12 or 16 IRKA poles perform better than other a priori pole options, while for the nonuniform mesh spacings, due to the ill-conditioning of the matrices, the Zolotarev poles with asymptotic approximation are more reliable. For all problems, we compared the convergence of a priori poles with the adaptive pole choices approach and found that, in the cases where the IRKA poles outperform the Zolotarev ones, they also show similar convergence to the adaptive approach. Moreover, we considered the influence of the ordering of the poles and found this is influenced by the choice of the right-hand side, with the highly oscillatory alternating right-hand side benefiting more from a descending pole ordering than other, smoothly varying right-hand side choices.

Furthermore, we considered the scalability of our rational approach as we increase the size of the problems. We found that, as n increases, the IRKA poles can be optimal in terms of iteration counts. However, the cost of generating the IRKA poles drives the overall CPU time up. The Zolotarev poles, despite being cheap to compute, can lead to long convergence times due to the increased number of iterations required. In all cases, we have seen that the adaptive approach can be faster than the a priori pole choices.

We finished Chapter 3 by looking at higher rank right-hand sides given by combinations of the rank-1 right-hand side choices we considered previously. We found that these results mimic those of the "worse-off" rank-1 right-hand sides for all our problems and pole choices.

Chapter 4 focused on matrix equations arising from the discretisation of convection–diffusion PDEs. The convection wind leads to three different classes of matrix equation: Lyapunov, Sylvester, and four-term generalised Sylvester equations. We considered each of these and presented solving strategies and limitations.

In Section 4.2 we focused on Lyapunov equations. We considered a rational Krylov projection method to solve them, and presented two upper bounds for the relative error, one using the eigenvector condition number, and one using the field of values. As for the Lyapunov equations from Chapter 3, we performed numerical tests with Zolotarev and IRKA poles and, as before, we have shown the superiority of the IRKA poles for Lyapunov equations arising from the discretisation of convection–diffusion PDEs.

In Section 4.3 we considered Sylvester equations which arise from the discretisation of convection–diffusion PDEs. We focused on adapting the rational

Krylov subspace approach used for Lyapunov equations, as well as the IRKA method for generating poles. From our empirical study, we have found that the IRKA poles generated from the coefficient matrix corresponding to the simpler of the two convection wind functions, despite being more expensive to compute than the Zolotarev poles, can result in lower iteration counts and total solver times for these Sylvester equations.

We finished this thesis by looking at a stationary iterative approach for solving a four-term generalised Sylvester equation. We have given a criterion for the convergence of this method and performed numerical experiments. We have found that, as we decrease the value of ϵ, hence making the problem more convective, the stationary iterative method diverges. We have also noticed that the solutions of the problems where $\|M^{-1}N\|_2 > 1$ present oscillations, indicating that the chosen finite difference method is not suitable for such problems.

Future work

In this thesis, we showed that using a priori computed IRKA poles can be advantageous for matrix equations arising from the discretisation of diffusion and convection–diffusion PDEs. We have seen, however, that the cost of generating these poles can be higher than other pole options. We have made some advances on how to reduce the cost of IRKA poles by exploring the differences in the poles when the tolarence on the relative change in poles is both stricter and looser, finding that a looser tolerance does not affect the poles, leading to our favourable results. Furthermore, we considered an approximation to the IRKA poles which has not been mentioned in this thesis since it was not as accurate as the current implementation. This focused on using scaled IRKA poles obtained for problems of smaller dimensions. Further work on how the IRKA poles are computed in practice can certainly be considered in order to reduce the cost. Approximating the IRKA poles has not been discussed in this thesis, and this might also prove to be less cost intensive than the current implementation of the pole generating algorithm.

We have seen that poles which include information about the right-hand side choice (IRKA and adaptive) seem to perform better in tests than the Zolotarev poles, which are generated using only the spectral interval. A natural continuation of these observations would be to establish them theoretically. Understanding this could lead to better pole choices for both rational Krylov projection methods and the ADI approach.

We have only touched on generalised Sylvester equations. To be precise, we only described a stationary iterative method which diverges as the problem becomes more convective. It would be interesting to develop a rational Krylov approach, such as the ones described for Lyapunov and Sylvester equations, to solve such matrix equations. Projection approaches and pole choices are two important areas that need to be considered in order to make such an approach work for generalised Sylvester equations.

Finally, we only looked at diffusion and convection–diffusion PDEs in two spatial dimensions, with separable convection winds. Our results can be extended from 2D to 3D (or higher dimensions) by considering tensors, which would form an excellent basis for future work.

References

[1] A. C. Antoulas. *Approximation of large-scale dynamical systems.* Society for Industial and Applied Mathematics, Philadelphia, PA, USA, 2005.

[2] J. Baker, M. Embree, and J. Sabino. Fast singular value decay for Lyapunov solutions with nonnormal coefficients. *SIAM J. Matrix Anal. Appl.*, 36(2):656–668, 2015.

[3] R. H. Bartels and G. W. Stewart. Solution of the matrix equation $AX + XB = C$ [F4]. *Comm. ACM*, 15(9):820–826, 1972.

[4] B. Beckermann. An error analysis for rational Galerkin projection applied to the Sylvester equation. *SIAM J. Numer. Anal.*, 49(6):2430–2450, 2011.

[5] P. Benner and T. Breiten. Low rank methods for a class of generalized Lyapunov equations and related issues. *Numer. Math.*, 124(3):441–470, 2013.

[6] P. Benner and T. Breiten. On optimality of approximate low rank solutions of large-scale matrix equations. *Systems Control Lett.*, 67:55–64, 2014.

[7] P. Benner and T. Breiten. Rational interpolation methods for symmetric Sylvester equations. *Electron. Trans. Numer. Anal.*, 42:147–164, 2014.

[8] P. Benner and T. Damm. Lyapunov equations, energy functionals, and model order reduction of bilinear and stochastic systems. *SIAM J. Control Optim.*, 49(2):686–711, 2011.

[9] P. Benner, P. Kürschner, and J. Saak. Self-generating and efficient shift parameters in ADI methods for large Lyapunov and Sylvester equations. *Electron. Trans. Numer. Anal.*, 43:142–162, 2014.

[10] P. Benner and J. Saak. Numerical solution of large and sparse continuous time algebraic matrix Riccati and Lyapunov equations: a state of the art survey. *GAMM-Mitt.*, 36(1):32–52, 2013.

[11] M. Benzi. Preconditioning techniques for large linear systems: A survey. *J. Comput. Phys.*, 182(2):418 – 477, 2002.

[12] D. A. Bini, B. Iannazzo, and B. Meini. *Numerical solution of algebraic Riccati equations*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2011.

[13] J. Boyle, M. Mihajlović, and J. Scott. HSL_MI20: An efficient AMG preconditioner for finite element problems in 3D. *Internat. J. Numer. Methods Engrg.*, 82(1):64–98, 2010.

[14] W. Briggs, V. Henson, and S. McCormick. *A Multigrid Tutorial: Second Edition*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

[15] B. C. Carlson and J. L. Gustafson. Asymptotic expansion of the first elliptic integral. *SIAM J. Math. Anal.*, 16(5):1072–1092, 1985.

[16] M. Crouzeix. Numerical range and functional calculus in Hilbert space. *J. Funct. Anal.*, 244(2):668–690, 2007.

[17] M. Crouzeix and C. Palencia. The numerical range is a $(1+\sqrt{2})$-spectral set. *SIAM J. Matrix Anal. Appl.*, 38(2):649–655, 2017.

[18] T. Damm. Direct methods and ADI-preconditioned Krylov subspace methods for generalized Lyapunov equations. *Numer. Linear Algebra Appl.*, 15(9):853–871, 2008.

[19] T. A. Davis. *Direct methods for sparse linear systems*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2006.

[20] V. Druskin and L. Knizhnerman. Extended Krylov subspaces: approximation of the matrix square root and related functions. *SIAM J. Matrix Anal. Appl.*, 19(3):755–771, 1998.

[21] V. Druskin, L. Knizhnerman, and V. Simoncini. Analysis of the rational Krylov subspace and ADI methods for solving the Lyapunov equation. *SIAM J. Numer. Anal.*, 49(5):1875–1898, 2011.

[22] V. Druskin, L. Knizhnerman, and M. Zaslavsky. Solution of large scale evolutionary problems using rational Krylov subspaces with optimized shifts. *SIAM J. Sci. Comput.*, 31(5):3760–3780, 2009.

[23] V. Druskin and V. Simoncini. Adaptive rational Krylov subspaces for large-scale dynamical systems. *Systems Control Lett.*, 60(8):546–560, 2011.

[24] V. Druskin, V. Simoncini, and M. Zaslavsky. Adaptive tangential interpolation in rational Krylov subspaces for MIMO dynamical systems. *SIAM J. Matrix Anal. Appl.*, 35(2):476–498, 2014.

[25] I. Duff, A. Erisman, and J. Reid. *Direct Methods for Sparse Matrices.* Oxford University Press, Oxford, 2017.

[26] N. S. Ellner and E. L. Wachspress. New ADI model problem applications. In *Proceedings of 1986 ACM Fall Joint Computer Conference, Dallas, TX, US*, pages 528–534, Los Alamitos, CA, USA, 1986.

[27] N. S. Ellner and E. L. Wachspress. Alternating direction implicit iteration for systems with complex spectra. *SIAM J. Numer. Anal.*, 28(3):859–870, 1991.

[28] H. Elman, D. Silvester, and A. Wathen. *Finite Elements and Fast Iterative Solvers : with Applications in Incompressible Fluid Dynamics: with Applications in Incompressible Fluid Dynamics.* Oxford University Press, Oxford, 2005.

[29] M. Embree. How descriptive are GMRES convergence bounds? Technical report, Oxford University Computing Laboratory, 1999.

[30] J.-E. Feng, J. Lam, G. Yang, and Z. Li. On a conjecture about the norm of Lyapunov mappings. *Linear Algebra Appl.*, 465:88–103, 2015.

[31] G. M. Flagg and S. Gugercin. On the ADI method for the Sylvester equation and the optimal-H_2 points. *Appl. Numer. Math.*, 64:50–58, 2013.

[32] M. A. Freitag and D. L. Green. A low-rank approach to the solution of weak constraint variational data assimilation problems. *J. Comput. Phys.*, 357:263–281, 2018.

[33] R. W. Freund and N. M. Nachtigal. QMR: a quasi-minimal residual method for non-hermitian linear systems. *Numer. Math.*, 60(1):315–339, 1991.

[34] Z. Gajic and M. T. J. Qureshi. *Lyapunov matrix equation in system stability and control.* Academic Press, San Diego, CA, USA, 1995.

[35] T. Gergelits, K.-A. Mardal, B. F. Nielsen, and Z. Strakos. Laplacian preconditioning of elliptic PDEs: localization of the eigenvalues of the discretized operator. *SIAM J. Numer. Anal.*, 57(3):1369–1394, 2019.

[36] G. Golub, S. Nash, and C. Van Loan. A Hessenberg-Schur method for the problem $AX + XB = C$. *IEEE Trans. Automat. Control*, 24(6):909–913, 1979.

[37] L. Grasedyck. Existence of a low rank or $\mathcal{H}$-matrix approximant to the solution of a Sylvester equation. *Numer. Linear Algebra Appl.*, 11(4):371–389, 2004.

[38] D. Griffiths, J. Dold, and D. Silvester. *Essential Partial Differential Equations: Analytical and Computational Aspects.* Springer International Publishing, 2015.

[39] S. Gugercin, A. C. Antoulas, and C. Beattie. H_2 model reduction for large-scale linear dynamical systems. *SIAM J. Matrix Anal. Appl.*, 30(2):609–638, 2008.

[40] S. Güttel. *Rational Krylov Methods for Operator Functions.* PhD thesis, Institut für Numerische Mathematik und Optimierung, Technische Universität Bergakademie Freiberg, 2010.

[41] S. Güttel. Rational Krylov approximation of matrix functions: Numerical methods and optimal pole selection. *GAMM-Mitt.*, 36(1):8–31, 2013.

[42] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *J. Res. Nat. Bur. Standards*, 49(6):409, 1952.

[43] N. J. Higham. Perturbation theory and backward error for AX- XB= C. *BIT Numerical Mathematics*, 33(1):124–136, 1993.

[44] N. J. Higham. *Functions of matrices: theory and computation.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2008.

[45] D. Y. Hu and L. Reichel. Krylov-subspace methods for the Sylvester equation. *Linear Algebra Appl.*, 172:283–313, 1992.

[46] I. Ipsen and C. Meyer. The idea behind Krylov methods. *The Amer. Math. Monthly*, 105, 11 1997.

[47] M.-P. Istace and J.-P. Thiran. On the third and fourth Zolotarev problems in the complex plane. *SIAM J. Numer. Anal.*, 32(1):249–259, 1995.

[48] I. M. Jaimoukha and E. M. Kasenally. Krylov subspace methods for solving large Lyapunov equations. *SIAM J. Numer. Anal.*, 31(1):227–251, 1994.

[49] E. Jarlebring, G. Mele, D. Palitta, and E. Ringh. Krylov methods for low-rank commuting generalized Sylvester equations. *Numer. Linear Algebra Appl.*, 25(6):2176, 2018.

[50] K. Jbilou. Low rank approximate solutions to large Sylvester matrix equations. *Appl. Math. Comput.*, 177(1):365–376, 2006.

[51] K. Jbilou and A. Riquet. Projection methods for large Lyapunov matrix equations. *Linear Algebra Appl.*, 415(2-3):344–358, 2006.

[52] C. R. Johnson. Normality and the numerical range. *Linear Algebra Appl.*, 15(1):89–94, 1976.

[53] L. Knizhnerman and V. Simoncini. Convergence analysis of the extended Krylov subspace method for the Lyapunov equation. *Numer. Math.*, 118(3):567–586, 2011.

[54] D. Kressner and P. Sirković. Truncated low-rank methods for solving general linear matrix equations. *Numer. Linear Algebra Appl.*, 22(3):564–583, 2015.

[55] D. Kulkarni, D. Schmidt, and S.-K. Tsui. Eigenvalues of tridiagonal pseudo-Toeplitz matrices. *Linear Algebra Appl.*, 297:63–80, 1999.

[56] P. Lancaster and M. Tismenetsky. *The theory of matrices: with applications*. Academic Press, San Diego, CA, USA, 2nd edition, 1985.

[57] J.-R. Li and J. White. Low rank solution of Lyapunov equations. *SIAM J. Matrix Anal. Appl.*, 24(1):260–280, 2002.

[58] A. Lu and E. L. Wachspress. Solution of Lyapunov equations by Alternating Direction Implicit iteration. *Comput. Math. Appl.*, 21(9):43–58, 1991.

[59] S. Massei and L. Robol. Rational Krylov for Stieltjes matrix functions: convergence and pole selection. *BIT*, 61(1):237–273, 2021.

[60] N. Nachtigal, S. Reddy, and L. Trefethen. How fast are nonsymmetric matrix iterations? *SIAM J. Matrix Anal. Appl.*, 13:778–795, 1992.

[61] I. V. Oseledets. Lower bounds for separable approximations of the Hilbert kernel. *Sb. Math.*, 198(3):425, 2007.

[62] C. C. Paige and M. A. Saunders. Solution of sparse indefinite systems of linear equations. *SIAM J. Numer. Anal.*, 12(4):617–629, 1975.

[63] D. Palitta. *Numerical Solution of Large-Scale Linear Matrix Equations.* PhD thesis, Alma Mater Studiorum Università di Bologna, 2018.

[64] D. Palitta and V. Simoncini. Matrix-equation-based strategies for convection–diffusion equations. *BIT*, 56(2):751–776, 2016.

[65] D. Palitta and V. Simoncini. Computationally enhanced projection methods for symmetric Sylvester and Lyapunov matrix equations. *J. Comput. Appl. Math.*, 330:648–659, 2018.

[66] D. Palitta and V. Simoncini. Numerical methods for large-scale Lyapunov equations with symmetric banded data. *SIAM J. Sci. Comput.*, 40(5):A3581–A3608, 2018.

[67] D. Palitta and V. Simoncini. Optimality properties of Galerkin and Petrov–Galerkin methods for linear matrix equations. *Vietnam J. Math.*, 48(4):791–807, 2020.

[68] D. W. Peaceman and H. H. Rachford, Jr. The numerical solution of parabolic and elliptic differential equations. *J. Soc. Indust. Appl. Math.*, 3(1):28–41, 1955.

[69] T. Penzl. Eigenvalue decay bounds for solutions of Lyapunov equations: the symmetric case. *Systems Control Lett.*, 40(2):139–144, 2000.

[70] B. Postlethwaite. Graded mesh. Version 1.5.0.0, [Accessed Online Feb. 2021].

[71] C. E. Powell, D. J. Silvester, and V. Simoncini. An efficient reduced basis solver for stochastic Galerkin matrix equations. *SIAM J. Sci. Comput.*, 39(1), 2017.

[72] Y. Saad. *Iterative Methods for Sparse Linear Systems: Second Edition.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2003.

[73] Y. Saad and M. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM J. Sci. and Stat. Comput.*, 7(3):856–869, 1986.

[74] J. Sabino. *Solution of large-scale Lyapunov equations via the Block Modified Smith Methods.* PhD thesis, Rice University, 2006.

[75] S. D. Shank, V. Simoncini, and D. B. Szyld. Efficient low-rank solution of generalized Lyapunov equations. *Numer. Math.*, 134(2):327–342, 2016.

[76] V. Simoncini. A new iterative method for solving large-scale Lyapunov matrix equations. *SIAM J. Sci. Comput.*, 29:1268–1288, 01 2007.

[77] V. Simoncini. Computational methods for linear matrix equations. *SIAM Rev.*, 58:377–441, 01 2016.

[78] V. Simoncini and V. Druskin. Convergence analysis of projection methods for the numerical solution of large Lyapunov equations. *SIAM J. Numer. Anal.*, 47(2):828–843, 2009.

[79] G. L. Sleijpen and D. R. Fokkema. BiCGstab (ℓ) for linear equations involving unsymmetric matrices with complex spectrum. *Electron. Trans. Numer. Anal.*, 1(11):2000, 1993.

[80] P. Sonneveld and M. B. Van Gijzen. IDR(s): A family of simple and fast algorithms for solving large nonsymmetric systems of linear equations. *SIAM J. Sci. Comput.*, 31(2):1035–1062, 2009.

[81] M. Stoll and T. Breiten. A low-rank in time approach to PDE-constrained optimization. *SIAM J. Sci. Comput.*, 37(1):B1–B29, 2015.

[82] T. Stykel and V. Simoncini. Krylov subspace methods for projected Lyapunov equations. *Appl. Numer. Math.*, 62(1):35–50, 2012.

[83] O. Tatebe. The multigrid preconditioned conjugate gradient method. In *Proceedings of Sixth Copper Mountain Conference on Multigrid Methods*, pages 621–634, 1993.

[84] L. Trefethen and D. Bau. *Numerical Linear Algebra.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1997.

[85] L. N. Trefethen and M. Embree. *Spectra and pseudospectra.* Princeton University Press, Princeton, 2005.

[86] I. T. Vaduva, P. A. Knight, and J. Pestana. An explicit rational approximation and efficient pole choices for Lyapunov equations arising from diffusion PDEs. *Submitted for publication.*

[87] H. A. Van der Vorst. BiCGStab: A fast and smoothly converging variant of BiCG for the solution of nonsymmetric linear systems. *SIAM J. Sci. and Stat. Comput.*, 13(2):631–644, 1992.

[88] C. F. Van Loan and G. Golub. *Matrix computations.* The Johns Hopkins University Press, Baltimore, MD, USA, 1996.

[89] B. Vandereycken and S. Vandewalle. A Riemannian optimization approach for computing low-rank solutions of Lyapunov equations. *SIAM J. Matrix Anal. Appl.*, 31(5):2553–2579, 2010.

[90] E. Wachspress. *The ADI model problem.* Springer, East Windsor, NJ, USA, 2013.

[91] E. L. Wachspress. The ADI minimax problem for complex spectra. In *Iterative methods for large linear systems*, pages 251–271. Elsevier, 1990.

[92] E. L. Wachspress. Trail to a Lyapunov equation solver. *Comput. Math. Appl.*, 55(8):1653–1659, 2008.

[93] T. Wolf, H. K. Panzer, and B. Lohmann. Model order reduction by approximate balanced truncation: A unifying framework. *at-Automatisierungstechnik*, 61(8):545–556, 2013.

[94] T. G. Wright. Eigtool: a graphical tool for nonsymmetric eigenproblems. *Oxford University Computing Laboratory*, 15(2):1, 2002.

[95] E. Zolotarev. Application of elliptic functions to questions of functions deviating least and most from zero. *Zap. Imp. Akad. Nauk. St. Petersburg*, 30(5):1–59, 1877.