RICE UNIVERSITY

Fixed-Polynomial Approximate Spectral Transformations for Preconditioning the Eigenvalue Problem

by

Heidi K. Thornquist

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree

Doctor of Philosophy

Approved, Thesis Committee:

Danny C. Sorensen, Chairman
Noah Harding Professor of Computational and Applied Mathematics

Matthias Heinkenschloss
Professor of Computational and Applied Mathematics

Mark Embree
Assistant Professor of Computational and Applied Mathematics

Matteo Pasquali
Associate Professor of Chemical and Biomolecular Engineering

Houston, Texas

May, 2006
Abstract

Fixed-Polynomial Approximate Spectral Transformations for Preconditioning the Eigenvalue Problem

by

Heidi K. Thornquist

Arnoldi’s method is often used to compute a few eigenvalues and eigenvectors of large, sparse matrices. When the eigenvalues of interest are not dominant or well-separated, this method may suffer from slow convergence. Spectral transformations are a common acceleration technique that address this issue by introducing a modified eigenvalue problem that is easier to solve. This modified problem accentuates the eigenvalues of interest, but requires the solution of a linear system, which is computationally expensive for large-scale problems. Furthermore, ensuring the precision of the computed eigenvalues with respect to the original eigenvalue problem requires restrictive accuracy requirements on the solution of each linear system for the modified eigenvalue problem.

This thesis shows how this expense can be reduced through a preconditioning scheme that uses a fixed-polynomial operator to approximate the spectral transformation. These operators are computed prior to being used in Arnoldi’s method and eliminate the accuracy requirements on the solution of each linear system for the modified eigenvalue problem. Three different constructions for a fixed-polynomial operator are derived from some common iterative methods for non-Hermitian linear
systems and the numerical behavior of these three operators are compared. There are significant computational benefits in precomputing an approximation to a spectral transformation, which become apparent in the discussion of implementation details as well as in the development of accuracy heuristics. Numerical experiments demonstrate that this preconditioning scheme is a competitive approach for solving large-scale eigenvalue problems. The results illustrate the effectiveness of this technique using several practical eigenvalue problems from science and engineering ranging from hundreds to more than a million unknowns.
Acknowledgements

First and foremost, I would like to thank my advisor, Danny Sorensen, who introduced me to the intriguing idea of approximating a spectral transformation with matrix polynomials. The discoveries presented in this thesis are largely due to his support and patience, for which I am extremely grateful.

I also wish to thank the rest of my committee for their interest in this thesis, especially Mark Embree. Our many discussions enabled the creation of a more complete and accurate document than would have been possible without his help.

The mere existence of this thesis has to be attributed to the wonderful mathematics teachers I have encountered over the years at Humboldt State University and Rice University. This unquestionably includes my undergraduate advisor, Charles Biles. I am forever indebted to him for his never-ending encouragement and guidance.

I am also appreciative for receiving the opportunity to finish this research while being gainfully employed at Sandia National Laboratories. The fortunate circumstance of working in an inspirational environment, surrounded by exceptional colleagues before earning my doctorate has not escaped my notice. I would especially like to thank David Womble, Mike Heroux, Rich Lehoucq, Andy Salinger, Karen Devine, and Bill Hart for their constant encouragement and friendship.

This challenging journey was enriched by the support of my friends and family. In particular, I would like to express my heart-felt gratitude to Carrie Jessen, Amy and Erin Wall, Chris Baker, Todd and Kristy Coffey, Shannon Scott, and Denis Ridzal for their friendship and emotional support. A special thanks goes to Daria Lawrence for keeping us graduate students in line and providing those much-needed hugs.

Lastly, and most importantly, I want to thank my parents, Geron and Marilyn Thornquist for their love, encouragement, and constant support. Without them, I would not have the solid foundation that allowed me to achieve this accomplishment.
## Contents

Abstract

Acknowledgements

List of Figures

List of Tables

1 Introduction

1.1 Notation and Pseudocode

1.2 The Eigenvalue Problem

1.3 The Power Method

1.4 The Rayleigh-Ritz Method

1.5 Krylov Subspace Methods

1.5.1 The Arnoldi Method

1.5.2 Spectral Transformations and Restarting

1.5.3 Eigenvector Purification

1.5.4 Implicitly Restarted Arnoldi Method

1.5.5 The Rational Krylov Method

1.5.6 Convergence of Krylov Subspace Methods

1.6 Subspace Iteration Methods

1.6.1 Jacobi-Davidson
### 2 IRA with a Fixed-Polynomial Operator

2.1 Iterative Methods for non-Hermitian Linear Systems

2.1.1 Generalized Minimum Residual Method

2.1.2 CGS Method

2.1.3 BiCGSTAB Method

2.1.4 Transpose-Free Quasi Minimum Residual Method

2.2 Fixed-Polynomial Operators

2.2.1 GMRES Fixed-Polynomial Operator

2.2.2 BiCGSTAB Fixed-Polynomial Operator

2.2.3 TFQMR Fixed-Polynomial Operator

2.3 Reconstructing Eigenpairs

2.4 Accuracy and Convergence Heuristics

2.5 Cost Comparison of Fixed-Polynomial Operators

### 3 Performance Studies for Several Practical Eigenproblems

3.1 Software and Experiment Parameters

3.2 Discretized 2D Laplacian Operator

3.3 Numerical Comparison of Fixed-Polynomial Operators using the NEP Collection

3.3.1 Brusselator Model

3.3.2 Crystal Structure Model

3.3.3 Olmstead Model

3.4 Numerical Comparison of the GMRES Fixed-Polynomial Operator and Jacobi-Davidson

3.4.1 Wind-Driven Barotropic Ocean Model

3.4.2 Semiconductor Laser Model
4 Performance Study for a Large-Scale Application: 8:1 Thermal Cavity Problem

4.1 Problem Definition .................................................. 139
4.2 Numerical Methods .................................................. 141
4.3 2D Numerical Results ............................................ 144
4.4 3D Numerical Results ............................................ 151

5 Conclusions ............................................................. 156

5.1 Future Directions .................................................. 158

Bibliography ............................................................. 160
List of Figures

1.1 The Power method ........................................... 6
1.2 The Rayleigh-Ritz procedure ................................. 8
1.3 The Harmonic Rayleigh-Ritz procedure ................... 9
1.4 The Arnoldi method ........................................... 12
1.5 Shift-invert and Möbius spectral transformations ........... 14
1.6 The Implicitly Restarted Arnoldi method ................. 18
1.7 The Rational Krylov method ................................. 20
1.8 General subspace iteration ................................... 26
1.9 The Shift-Invert Simultaneous Iteration method ........... 27
1.10 The Jacobi-Davidson method ............................... 30
1.11 The block preconditioned eigensolver ..................... 32
1.12 The Locally-Optimal Block Preconditioned Conjugate Gradient method 33

2.1 Right-preconditioned GMRES Algorithm .................. 46
2.2 Lanczos Biorthogonalization Algorithm .................. 49
2.3 Biconjugate Gradient Algorithm .......................... 50
2.4 Conjugate Gradient Squared Algorithm .................. 53
2.5 Right-preconditioned BiCGSTAB Algorithm ............. 55
2.6 Right-preconditioned TFQMR Algorithm ................ 64
2.7 Algorithm implementing IRA with fixed-polynomial operator . 67
2.8 Applying the GMRES fixed-polynomial operator .......... 69
2.9 Applying the BiCGSTAB fixed-polynomial operator ....... 72
2.10 Applying the TFQMR iteration polynomial (Algorithm 1) 74
2.11 Applying the TFQMR iteration polynomial (Algorithm 2) 75
2.12 Pseudospectrum of 2D convection-diffusion equation 87
2.13 Linear solver convergence curves for the SUPG discretized convection-diffusion equation with $N = 13$ 88
2.14 Contour plots of the GMRES, BiCGSTAB, and TFQMR residual polynomials over $\sigma(A)$ 90
2.15 Contour plot of the GMRES residual polynomial over $\sigma(A)$ for $k = 26$ 91
2.16 Contour plots of the GMRES, BiCGSTAB, and TFQMR residual polynomials over $\sigma(A)$ 93
2.17 Linear solver convergence curves for the SUPG discretized convection-diffusion equation ($N = 13$) right preconditioned by an incomplete LU factorization 94
2.18 Pseudospectrum of the 2D convection-diffusion equation right preconditioned with an incomplete LU factorization (drop tolerance = $10^{-1}$) 95
2.19 Contour plots of the GMRES, BiCGSTAB, and TFQMR residual polynomials over $\sigma(AM^{-1})$ where $M = luinc(A, 1e^{-1})$ 96
2.20 Pseudospectrum of the 2D convection-diffusion equation right preconditioned with an incomplete LU factorization (drop tolerance = $10^{-2}$) 97
2.21 Contour plots of the GMRES, BiCGSTAB, and TFQMR residual polynomials over $\sigma(AM^{-1})$ where $M = luinc(A, 1e^{-2})$ 98

3.1 Matrix-vector products needed to compute smallest five eigenvalues of large-scale, discretized 2D Laplacian operator 106
3.2 Ritz residual for the smallest five approximate eigenvalues of large-scale, discretized 2D Laplacian operator; maximum dimension of the Krylov subspace required by GMRES($\infty$) 107
3.3 Relative error for the smallest five approximate eigenvalues of small-scale, discretized 2D Laplacian operator; 500 matrix-vector products .................................................. 108
3.4 Relative error for the smallest five approximate eigenvalues of small-scale, discretized 2D Laplacian operator; 1000 matrix-vector products .................................................. 109
3.5 Spectrum of reaction-diffusion Brusselator model ....................... 111
3.6 Convergence curves for the Brusselator model with $N = 2048$ ........ 112
3.7 Parameter study of polynomial degree versus Krylov subspace dimension for computing the rightmost eigenpair of the Brusselator model . 114
3.8 Spectrum of crystal growth diffusion model .............................. 116
3.9 Convergence curves for the crystal growth diffusion model with $N = 2500118$
3.10 Parameter study of polynomial degree versus Krylov subspace dimension for computing the rightmost eigenpair of the crystal growth model 119
3.11 Spectrum of Olmstead model .............................................. 121
3.12 Convergence curves for the Olmstead model with $N = 2000$ .......... 122
3.13 Parameter study of polynomial degree versus Krylov subspace dimension for computing the rightmost eigenpair of the Olmstead model . 123
3.14 Domain (a) and wind stress forcing (b) of wind-driven barotropic ocean model .................................................. 126
3.15 Steady-state streamfunctions depicting double-gyre structure;
(a) $Ro = .0001$, (b) $Ro = .0016$, (c) $Ro = .0036$ ...................... 127
3.16 Sparsity pattern and spectrum of linearized ocean model; dim = 903 . 128
3.17 Spectra of restricted-degree, fixed-polynomial operators .............. 129
3.18 Relative error for the rightmost eigenvalues of wind-driven barotropic ocean model; $M = luinc(A_\sigma, 10^{-3})$; 1000 matrix-vector products . . 130
3.19 Relative error for the rightmost eigenvalues of wind-driven barotropic ocean model; $M = luinc(A_\sigma, 10^{-4})$; 500 matrix-vector products . . . 131
3.20 Sample spectrum of complex, non-hermitian matrix from sequence indicating eigenvalues of interest ........................................ 134
3.21 Tracking eigenvalues of interest throughout matrix sequence . . . 135
3.22 Relative errors for eigenvalues of interest, quality of preconditioner and cost of IRA with GMRES fixed-polynomial operator ............... 136

4.1 Geometry of the 2D thermal cavity ........................................ 140
4.2 Approximate rightmost part of the spectrum of discretized 2D thermal cavity problem ....................................................... 145
4.3 Parameter study of polynomial degree versus Krylov subspace dimension for computing the rightmost eigenpair of the 2D thermal cavity problem \( N = 43508 \) .......................................................... 147
4.4 Relative linear solver time (wall-clock) for computing the rightmost eigenpair of the 2D thermal cavity problem \( N = 43,508 \) .......... 148
4.5 Approximate rightmost part of the spectrum of discretized 3D thermal cavity problem ............................................................... 151
4.6 Parameter study of polynomial degree versus Krylov subspace dimension for computing the rightmost eigenpair of the 3D thermal cavity problem \( N = 1,096,865 \) .......................................................... 153
4.7 Relative linear solver time (wall-clock) for computing the rightmost eigenpair of the 3D thermal cavity problem \( N = 1,096,865 \) .... 154
List of Tables

2.1 Summary of Fixed-Polynomial Operator Comparisons . . . . . . . . . . 86
2.2 Summary of Residual Polynomial Comparisons . . . . . . . . . . . . . 99
3.1 Experiment parameters for Jacobi-Davidson (JDQR) . . . . . . . . . . 103
3.2 Experiment parameters for LOBPCG . . . . . . . . . . . . . . . . . . 103
3.3 Experiment parameters for IRA with Fixed-Polynomial Operator . . 104
3.4 Rightmost five eigenvalues of reaction-diffusion Brusselator model . . 112
3.5 Rightmost five eigenvalues of crystal growth diffusion model . . . . . 117
3.6 Rightmost five eigenvalues of Olmstead model . . . . . . . . . . . . . 120
4.1 Cayley transform parameters and Rayleigh number for 2D thermal
cavity problem . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 145
4.2 Specification for a single QED compute node used to obtain results for
thermal cavity problem . . . . . . . . . . . . . . . . . . . . . . . . . . . 146
4.3 Average wall-clock times for the linear system solve of the 2D thermal
cavity problem \((N = 43,508)\) for a given iteration polynomial degree 150
4.4 Cayley transform parameters and Rayleigh number for 3D thermal
cavity problem . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 152
4.5 Average wall-clock times for the linear system solve of the 3D thermal
cavity problem \((N = 1,096,865)\) for a given iteration polynomial degree155
Chapter 1

Introduction

Arnoldi’s method [2, 67] is often used to compute the eigenvalues and eigenvectors of large, sparse matrices. Various techniques can be employed to improve the convergence of this method, some resulting in considerable computational expense. This thesis introduces an efficient preconditioning scheme that greatly accelerates convergence to the rightmost eigenvalues of non-Hermitian matrices. Furthermore, this scheme can easily be implemented with existing Arnoldi-based software, extending the domain of application.

Before laying out the motivation for this preconditioning scheme, some notation and definitions will be discussed in Section 1.1. Following this, the large-scale eigenvalue problem will be introduced in Section 1.2. The foundation of most current methods for solving this problem is the power method, which will be presented in Section 1.3. Generalizations of the power method to larger subspaces require extraction techniques like the Rayleigh-Ritz method discussed in Section 1.4. Arnoldi’s method is examined in Section 1.5 in addition to some of the acceleration and computational techniques it is commonly paired with. This will set the stage for the Implicitly Restarted Arnoldi [85] and rational Krylov [63] methods.

The discussion of acceleration techniques for Arnoldi’s method will show the need for a spectral transformation to improve the convergence of Arnoldi-based methods.
to the rightmost eigenvalues. These spectral transformations typically require the solution of a linear system (linear solve) for each basis vector of the Krylov subspace. Utilizing iterative methods for these linear solves requires a high degree of accuracy to ensure that the computed subspace is numerically close to a Krylov subspace. For large-scale problems such accuracy may be necessary, but computationally expensive. Alternatively, we propose a preconditioning scheme that uses a fixed-polynomial operator to approximate a spectral transformation, thus removing the computational expense of a high-accuracy linear solve. The fixed-polynomial can then be used to produce a modified problem with approximately the same solution as the original. Most importantly, this modified problem is one that the Krylov subspace method can solve rapidly.

1.1 Notation and Pseudocode

The following notation will be used throughout this thesis.

- \( \mathbb{R} \) and \( \mathbb{C} \) will denote the real and complex numbers, respectively.
- Capital and lower case Latin letters denote matrices and vectors, respectively. Lower case Greek letters will denote scalars.
- The transpose of a matrix \( A \) is denoted by \( A^T \) and the conjugate-transpose is denoted by \( A^H \); likewise for vectors.
- Unless otherwise specified, \( \| \cdot \| \) is the Euclidean norm of a vector and the induced two-norm of a matrix.
- The \( j \)th canonical basis vector is denoted by \( e_j \).
- A diagonal matrix whose diagonal elements are \( \alpha_i, i = 1, \cdots, n \) is denoted by \( \text{diag}(\alpha_1, \alpha_2, \cdots, \alpha_n) \).
• Given a matrix $A$, $\mathcal{R}(A)$ and $\mathcal{N}(A)$ denote the range and null space of $A$, respectively.

• Given a scalar $\alpha \in \mathbb{C}$, $\Re(\alpha)$ and $\Im(\alpha)$ are the real and imaginary parts of $\alpha$, respectively.

Pseudocode will be provided throughout the thesis when it is appropriate and illustrative of the current discussion. For briefness of presentation, the pseudocode may use Matlab® syntax and intrinsic functions. Any other functions referenced in the pseudocode will be defined in the discussion.

1.2 The Eigenvalue Problem

The algebraic eigenvalue problem

$$Ax = x\lambda$$  \hfill (1.1)

is a fundamental problem in scientific computing. It appears in disciplines like economics, dynamical systems, control theory and mechanics. The computation of solutions $(\lambda, x)$ to equation (1.1) is important for performing structural and stability analysis on mathematical models in these disciplines.

Large eigenvalue problems commonly arise from a finite dimensional approximation of a continuous model

$$\mathcal{L}u = u\lambda,$$  \hfill (1.2)

where $\mathcal{L}$ is a linear differential operator. Often, $\mathcal{L}$ is the linearization of a nonlinear operator about a steady state. If the continuous model (1.2) is discretized by finite differences, the result is a standard eigenvalue problem (1.1). Discretization by a finite element method will lead to a generalized eigenvalue problem of the form

$$Ax = Bx\lambda.$$  \hfill (1.3)
Usually $A$ is referred to as the *stiffness* matrix and $B$ the *mass* matrix. Sometimes the matrices $A$ and $B$ in the generalized eigenvalue problem (1.3) are referred to as the matrix pencil $(A, B)$.

This thesis will address both the standard and generalized eigenvalue problem. The scalar, $\lambda$, and vector, $x$, that satisfy (1.1) or (1.3) are called the *eigenvalue* and *(right) eigenvector*, respectively. Often they are also referred to as an *eigenpair* $(\lambda, x)$. Furthermore, an eigenvalue will also have a corresponding nonzero *left eigenvector* $y$ that satisfies

$$
y^H A = \lambda y^H B,
$$

where $B = I$ if the original eigenvalue problem is in standard form (1.1).

Standard and generalized eigenvalue problems are classified by the structure of the matrix $A$ or the matrix pencil $(A, B)$, respectively. These classifications guide the design of algorithms through theoretical insight. If $A$ is Hermitian, $A = A^H$, then (1.1) is referred to as a Hermitian eigenvalue problem. Furthermore, if $A$ and $B$ are both Hermitian, then (1.3) is referred to as a generalized Hermitian eigenvalue problem. The eigenvalues are always real for Hermitian eigenvalue problems and both their left and right eigenvectors are the same. This also holds true for the generalized case when $B$ is positive definite. However, only the standard Hermitian eigenvalue problem is guaranteed to have an orthonormal basis of eigenvectors.

When $A$ is non-Hermitian, $A \neq A^H$, then (1.1) is referred to as a non-Hermitian eigenvalue problem. Likewise, if $A$ and/or $B$ is non-Hermitian, then (1.3) is referred to as a generalized non-Hermitian eigenvalue problem. In both cases, the eigenvalues may be real, complex, or appear as complex conjugate pairs and the left and right eigenvectors are usually not the same. Even worse, a complete set of independent eigenvectors may not exist, complicating the theory and algorithms for non-Hermitian eigenvalue problems [5]. When a matrix or matrix pencil does not have an orthonormal basis of eigenvectors, it will be referred to as *non-normal*.

Numerical methods that solve the standard (1.1) and generalized (1.3) eigenvalue
problems without direct factorizations or similarity transformations are important since the discretization of the operator $L$ results in matrices that are likely to be very large and sparse. Thus, matrix-vector products can cost much less than $n^2$ floating point operations. The focus of this thesis will be on methods that require only matrix-vector products, also called \textit{iterative methods}.

Many iterative methods for finding solutions to (1.1) compute approximations to the eigenpairs in a lower-dimensional subspace $S_k$. The classic power method, presented in Section 1.3, computes solutions to (1.1) where the dimension of the subspace $S_k$ is one. The behavior of this single-vector iteration is the foundation for understanding the current methods for solving eigenvalue problems. Generalizing the power method to a subspace $S_k$ with dimension larger than one results in two families of methods: Krylov subspace methods (Section 1.5) and subspace iteration methods (Section 1.6). These families differ in the choice of the subspace $S_k$, but both use the Rayleigh-Ritz procedure discussed in Section 1.4 to extract approximations to the desired eigenvalues and eigenvectors of $A$ from $S_k$.

\section{1.3 The Power Method}

The power method is a simple and efficient way to compute a single eigenvalue and its corresponding eigenvector of the standard eigenvalue problem (1.1). The method, described in Figure 1.1, only requires the application of a fixed operator $A$ at step 1.1. This vector is then normalized such that $\|v\|_\infty = 1$, where $i_{\text{max}}(w)$ returns the index of $w$ corresponding to the entry of largest magnitude. It is well known that the power method tends to converge to the eigenvalue of $A$ with the largest magnitude and its corresponding eigenvector [95]. However, with the use of a spectral transformation, this method can be made to converge to any specified eigenvalue in the spectrum of $A$.

The convergence analysis for this method is straightforward when $A$ is diagonal-
Input: \((A,v)\)
Output: An eigenvalue, \(\lambda\), of \(A\) and its corresponding eigenvector \(v\).

1. for \(k = 1,2,\cdots\), until convergence
   1.1. \(w = Av\)
   1.2. \(j = \text{i} \_ \text{max}(w)\)
   1.3. \(\lambda = w(j)\)
   1.4. \(v \leftarrow w/\lambda\)
2. end

Figure 1.1: The Power method

izable and has \(n\) distinct eigenvalues which can be ordered such that

\[ |\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n| . \]

Generally, the convergence rate of \(v\) to the eigenvector \(q_1\) corresponding to \(\lambda_1\) is linear and proportional to \(|\lambda_2/\lambda_1|\). One can show that there is a positive constant \(c\) such that

\[ \|v^{(k)} - q_1\| \leq c \left| \frac{\lambda_2}{\lambda_1} \right|^k \|v^{(1)} - q_1\|. \]

Thus, convergence may be slow if \(\lambda_1\) is not well separated from the rest of the spectrum. However, if \(v\) is orthogonal to \(q_1\) and \(A\) is normal, the power method (in exact arithmetic) will converge to the eigenvector \(q_2\) corresponding to \(\lambda_2\) at a rate proportional to \(|\lambda_3/\lambda_2|\). This realization motivates the idea of deflation, by which another eigenpair of \(A\) is exposed.

Convergence to any eigenvalue of \(A\) near a shift \(\sigma\) can be achieved through the use of a shift-invert spectral transformation, \((A - \sigma I)^{-1}\). This is called the inverse power method and is useful for computing interior eigenvalues of \(A\), but it still obtains only linear convergence. There is a natural extension of this method through the variation of the shift at each step. By choosing \(\sigma\) to be the Rayleigh quotient of \(v\), \(v^H Av\), the Rayleigh Quotient Iteration (RQI) is obtained. Convergence is generally quadratic for this method, but increases to cubic if \(A\) is Hermitian.
The main drawback of the power method is that it only computes one eigenpair at a time. Deflation using the spectral projector of $\lambda_1$,

$$P_1 = q_1 \hat{q}_1^H,$$

where $\hat{q}_1$ is a left eigenvector of $\lambda_1$ normalized so that $\hat{q}_1^H q_1 = 1$, can be used to continue the computation. By restricting $A$ to the complementary invariant subspace of $\lambda_1$, $(I - q_1 \hat{q}_1^H)A(I - q_1 \hat{q}_1^H)$, convergence to $q_2$ can be achieved. This mechanism allows for the computation of more than one eigenpair, but it can be numerically sensitive when $A$ is non-normal.

There are two generalizations of the power method for computing multiple eigenpairs: Krylov subspace methods and subspace iteration methods. Krylov subspace methods consider a sequence of vectors generated by the power method, while subspace iteration methods perform the power method on a block of vectors. Either of these generalizations require a technique to extract approximate eigenpairs from the subspace $S_k$. The Rayleigh-Ritz procedure is the frequently chosen technique.

### 1.4 The Rayleigh-Ritz Method

A common issue for any method that generates a subspace $S_k$ from which approximate eigenpairs of $A$ are computed is the extraction of these approximations from $S_k$. The Rayleigh-Ritz procedure, shown in Figure 1.2, is often chosen to perform this task. The procedure projects $A$ onto the subspace $S_k$ using the orthonormal basis for the subspace, stored in the columns of $V$. The resulting matrix

$$H = V^H AV$$

(1.5)

called the Rayleigh quotient. The eigenvalue $\mu_j$ of $H$ is called a Ritz value and the eigenvector $y_j$ of $H$ is used to compute the Ritz vector, $x_j = V y_j$. The Ritz pair $(\mu_j, x_j)$ is an approximate eigenpair of $A$ and has an associated Ritz estimate

$$\|Ax_j - \mu_j x_j\|.$$ (1.6)
Input: \((A, V \in \mathbb{C}^{n \times k})\), where \(V^H V = I\) and \(S_k = \mathcal{R}(V)\)

Output: Approximate eigenpairs \((\mu_j, x_j)\) of \(A\).

1. \(W = AV\)
2. \(H = V^H W\)
3. Compute the \(k\) eigenpairs \((\mu_j, y_j)\) of \(H\)
4. Compute the \(k\) Ritz pairs \((\mu_j, x_j = V y_j)\) of \(A\) and corresponding residuals \(r_j = Ax_j - x_j \mu_j\)

Figure 1.2: The Rayleigh-Ritz procedure

The approximate eigenpairs computed by the Rayleigh-Ritz method have been extensively analyzed for a general matrix \(A\) [55, 56, 36]. It is well known that for Hermitian matrices the Ritz values and Ritz vectors obtained with this procedure are the optimal approximations given only the information contained in \(V\) [61]. Furthermore, the Ritz estimate can be used to provide a rigorous bound on the accuracy of the eigenvalues of \(H\) as estimates for the eigenvalues of \(A\). For non-Hermitian matrices a small Ritz estimate does not imply that the Ritz value is equally close to an eigenvalue of \(A\) due to possible non-normality.

In practice it has been noticed that the Rayleigh-Ritz procedure usually computes better estimates for the exterior eigenvalues of a general matrix \(A\) [56]. Even for a Hermitian matrix, the Rayleigh-Ritz procedure may give poor approximations for interior eigenvalues. This is because Ritz vectors are generally a combination of the eigenvectors corresponding to the eigenvalues in the neighborhood of its Ritz value. The Rayleigh-Ritz procedure is optimal for exterior eigenvalues [61], so the neighborhood of influence for Ritz vectors is smaller when a Ritz value is near the exterior of the spectrum. However, interior Ritz values have a larger neighborhood of influence resulting in Ritz vectors that are composed of eigenvector directions from all over the spectrum. As a result, interior Ritz values can sometimes be considered “spurious” because their value is not meaningful.
Input: \((A, \sigma, P \in \mathbb{C}^{n \times k})\) and \(\mathcal{S}_k = \mathcal{R}(P)\)
Output: Approximate eigenpairs \((\mu_j, x_j)\) of \(A\).

1. \(Q = (A - \sigma I)P\)
2. \(H = Q^H P\)
3. \(M = Q^H Q\)
4. Compute the \(k\) eigenpairs \((\theta_j, y_j)\) of \(Hy_j = \theta_j My_j\)
5. Compute the \(k\) harmonic Ritz pairs \((\mu_j = \sigma + \frac{1}{\theta_j}, x_j = Qy_j)\) of \(A\) and corresponding residuals \(r_j = Ax_j - x_j \mu_j\)

Figure 1.3: The Harmonic Rayleigh-Ritz procedure

The lack of confidence in interior Ritz values is problematic since there is an increasing number of applications interested in the interior eigenvalues of \(A\). For these applications it would be more desirable to convert the interior eigenvalues to exterior eigenvalues. As with the power method, using a shift-invert spectral transformation \((A - \sigma I)^{-1}\) in the Rayleigh-Ritz procedure extracts better approximations for the eigenvalues of \(A\) around a shift \(\sigma\). Pairing the inverted operator \((A - \sigma I)^{-1}\) with the Rayleigh-Ritz procedure is effective [75] but impractical. However, the Rayleigh-Ritz procedure applied to \((A - \sigma I)^{-1}\) can be reformulated to avoid the application of the inverted operator, resulting in the harmonic Rayleigh-Ritz procedure [56] in Figure 1.3.

Consider a generalized form of the Rayleigh-Ritz procedure where the basis \(Q\) of the subspace \(\mathcal{S}_k\) is not necessarily orthonormal. The procedure would compute solutions to the eigenproblem

\[
Q^H AQy_j = \mu_j Q^H Qy_j,
\]

where \((\mu_j, Qy_j)\) are the Ritz pairs of \(A\). The generalized Rayleigh-Ritz procedure for an inverted operator would compute solutions to the eigenproblem

\[
Q^H (A - \sigma I)^{-1} Qy_j = \theta_j Q^H Qy_j;
\]
where \((\mu_j = \sigma + \frac{1}{\theta_j}, Qy_j)\) are the harmonic Ritz pairs of \(A\). To eliminate the inverted operator, perform the Rayleigh-Ritz procedure on \(P = (A - \sigma I)Q\). Substituting \(P\) into (1.8) gives an eigenproblem
\[
Q^H (A - \sigma I)^H Qy = \theta Q^H (A - \sigma I)^H (A - \sigma I)Qy
\] (1.9)
that has the same harmonic Ritz pairs as (1.8) but avoids the application of \((A-\sigma I)^{-1}\).

The Rayleigh-Ritz or harmonic Rayleigh-Ritz procedure underlies most iterative methods for solving eigenvalue problems that are based on subspace projection. More specifically, this procedure is utilized in all the methods presented in this thesis. We will now discuss the first generalization of the power method, Krylov subspace methods.

### 1.5 Krylov Subspace Methods

A single-vector iteration like the power method does not take advantage of the information provided by the successive vectors in the sequence
\[
v, Av, A^2v, A^3v, \ldots
\]
This sequence may be rich with information along eigenvector directions that correspond to the dominant eigenvalues of \(A\). A natural consequence is to consider the \(k\)-dimensional Krylov subspace
\[
\mathcal{K}_k(A, v) = \text{span}\{v, Av, A^2v, \ldots, A^{k-1}v\}.
\] (1.10)
Methods that use this subspace, \(S_k = \mathcal{K}_k(A, v)\), to compute estimates for eigenpairs of \(A\) are called Krylov subspace methods.

#### 1.5.1 The Arnoldi Method

Computationally, Krylov subspace methods often use an orthonormal basis for the subspace (1.10). The construction of this basis is performed by the Arnoldi method.
[2] and results in the *k-step Arnoldi factorization* of $A$

\[ AV_k = V_k H_k + f e_k^T, \quad V_k^H V_k = I_k, \quad V_k^H f = 0, \quad \text{(1.11)} \]

where the columns of $V_k$ form an orthonormal basis for $\mathcal{K}_k(A,v)$. Figure 1.4 illustrates the generation of $V_k \in \mathbb{C}^{n \times k}$ and the matrix

\[ H_k = V_k^H AV_k, \quad \text{(1.12)} \]

which is the Rayleigh-Ritz projection of $A$ onto the subspace $\mathcal{K}_k(A,v_0)$. Generally, $H_k \in \mathbb{C}^{k \times k}$ is an upper Hessenberg matrix, an upper-triangular matrix with elements along the first subdiagonal. If $A$ is Hermitian then $H_k \in \mathbb{R}^{k \times k}$ is a symmetric, tridiagonal matrix and the factorization (1.11) is called the *k-step Lanczos factorization* of $A$ [40].

An equivalent representation for the Arnoldi factorization (1.11) is

\[ AV_k = V_k H_k + \beta v_{k+1} e_k^T = V_{k+1} \bar{H}_k \quad \text{(1.13)} \]

where $\beta = \| f \|$, $v_{k+1} = f / \beta$ and $\bar{H}_k$ is the augmented Hessenberg matrix, $\bar{H}_k = \begin{bmatrix} H_k \\ \beta e_k^T \end{bmatrix} \in \mathbb{C}^{k+1 \times k}$. If $\beta = 0$ for some $k$, then either representation of the Arnoldi factorization indicates that an invariant subspace has been computed

\[ AV_k = V_k H_k. \]

This situation is considered a *lucky breakdown* of the Arnoldi process because it implies that the eigenvalues of $H_k$ are a subset of the eigenvalues of $A$. All the discussions in this thesis will assume that $H_k$ is an *unreduced* Hessenberg matrix, meaning that no lucky breakdown has occurred up to step $k$.

Extracting the eigenvalue and eigenvector estimates for $A$ from the Arnoldi factorization follows the Rayleigh-Ritz procedure (Section 1.4). Using the eigenpairs $(\theta, y)$ of $H_k$, we obtain the Ritz value $\theta$ and Ritz vector $x = V_k y$. The Arnoldi factorization (1.11) gives us an equivalent representation for the Ritz estimate of an approximate eigenpair $(\theta, x)$

\[ \| Ax - x\theta \| = \| (AV_k - V_k H_k)y \| = |\beta e_k^T y|. \quad \text{(1.14)} \]
Input: \((A, v_0, k)\)

Output: \((V_k, H_k, f)\) where \(AV_k = V_kH_k + fe_k^T\), \(V_k^HV_k = I_k\), and \(V_k^Hf = 0\).

1. \(v_1 = v_0/\|v_0\|\)
2. \(w = Av_1, \alpha = v_1^Hw\)
3. \(H_1 = [\alpha], V_1 = [v_1], f = w - v_1\alpha\)
4. for \(j = 1, 2, \ldots, k - 1\)
   4.1. \(\beta = \|f\|, v_{j+1} = f/\beta\)
   4.2. \(\tilde{H}_j \leftarrow \begin{bmatrix} H_j \\ \beta e_j^T \end{bmatrix}, V_{j+1} = [V_j \ v_{j+1}]\)
   4.3. \(w = Av_{j+1}, h = V_{j+1}^Hw\)
   4.4. \(H_{j+1} \leftarrow [\tilde{H}_j \ h], f = w - V_{j+1}h\)
5. end

Figure 1.4: The Arnoldi method

The estimate \(|\beta e_k^Ty|\) is an inexpensive by-product of the Arnoldi factorization because it avoids the potentially expensive application of \(A\) to obtain a residual norm.

There are a couple of implementation details to notice about the Arnoldi method given in Figure 1.4. First, the method does not require explicit storage of the matrix \(A\), only matrix-vector products at steps 2 and 4.3. Second, the Arnoldi method uses the classical Gram-Schmidt (CGS) algorithm to build the orthogonal Krylov basis in steps 4.3 and 4.4. For eigenvalue calculations an orthogonal basis is essential and CGS is notoriously unstable. Replacing classical Gram-Schmidt with modified Gram-Schmidt (MGS) does not always ensure orthogonality. Furthermore, MGS is not as computationally efficient as CGS. The best solution to this problem is to reorthogonalize the Krylov basis using the Daniel, Gragg, Kaufman, and Stewart (DGKS) correction [15] whenever necessary.

It is impossible to know in advance how large the Krylov subspace must be before
the eigenvalues of $H_k$ are good approximations to the eigenvalues of interest. For large-scale problems, storing a sufficient number of basis vectors may not be possible. Incorporating an acceleration scheme with the Arnoldi method helps keep $k$ small, while still converging to the selected eigenvalues. Two techniques briefly discussed in the next section are spectral transformations and restarting. These are essential for practical implementations, like the Implicitly Restarted Arnoldi (IRA)[85] and rational Krylov [63] methods. More information on acceleration techniques for the Arnoldi method can be found in Yang’s thesis [97].

1.5.2 Spectral Transformations and Restarting

Spectral transformations improve the convergence of the Arnoldi method through the substitution of $A$ with $\psi(A)$, where $\psi(\lambda)$ is some simple function. This transformation serves to map interior or clustered eigenvalues of $A$ to dominant, well-separated eigenvalues of $\psi(A)$. The most common of these is the shift-invert spectral transformation

$$\psi_{SI}(\lambda) = \frac{1}{\lambda - \sigma},$$

which emphasizes the eigenvalues of $A$ around the shift $\sigma$ (Figure 1.5). The Arnoldi method applied to this transformation, called shift-invert Arnoldi, is a generalization of the inverse power method. If the shift $\sigma$ is varied at each step, the shift-invert Arnoldi method becomes a generalization of RQI called rational Krylov [63].

Another interesting spectral transformation is the Möbius transformation

$$\psi_{M}(\lambda) = \frac{\lambda - \mu}{\lambda - \sigma}.$$  

This transformation maps the eigenvalues of $A$ close to $\sigma$ far from the unit circle, and those close to $\mu$ to eigenvalues of $\psi_{M}(A)$ with small modulus (Figure 1.5). In exact arithmetic there is no advantage to using the Möbius transformation since it can be written as a linear translation of $\psi_{SI}(A)$

$$\psi_{M}(A) = I + (\sigma - \mu)\psi_{SI}(A)$$
Figure 1.5: Spectral transformations: shift-invert (top) and Möbius (bottom)

and Arnoldi’s method is translation invariant. However, it can be beneficial to use $\psi_M(A)$ when the application of the spectral transformation is approximated.

Restarting accelerates convergence of the Arnoldi method by modifying the starting vector $v$ using eigenvector information obtained in the previous Krylov subspace. Simple restarting replaces the starting vector with a single Ritz vector [69]. Polynomial restarting generalizes this by updating the starting vector $v$ with

$$v \leftarrow \psi(A)v,$$

where $\psi(\lambda)$ is constructed to filter out unwanted eigenvector components. The easiest
method uses a linear combination of Ritz vectors

\[ v^+ = \sum_{j=1}^{k} q_j \gamma_j, \]  

(1.17)

with some appropriate choice of weights \( \gamma_j \) [67, 70].

Another approach for building \( \psi \) is to specify its roots as points in a region containing unwanted eigenvalues. A straightforward choice would be to select a set of eigenvalues from the current \( H_m \) and use them as the roots of the polynomial \( \psi \). This method of selection, suggested by Sorensen [85], is called exact shifts. If \( K \) is a compact set containing the unwanted eigenvalues then Leja points [46] or Fejér points (cite) from \( K \) can also be used as roots of the polynomial \( \psi \). Given a real-valued, positive function \( w(z) \) defined on \( K \), the sequence of Leja points \( z_j \) satisfies

\[
\begin{align*}
    w(z_0)|z_0| &= \max_{z \in K} w(z)|z| \\
    w(z_j) \prod_{k=0}^{j-1} |z_j - z_k| &= \max_{z \in K} w(z) \prod_{k=0}^{j-1} |z - z_k|,
\end{align*}
\]

for \( j = 1, 2, \ldots \). These points are uniformly distributed in \( K \) with respect to \( w(z) \) and are typically expensive to compute exactly. Fast Leja points are a more practical alternative, providing an inexpensive approximation to Leja points [3]. Fejér points are also uniformly distributed in \( K \) and require a conformal mapping \( \varphi(w) \) on \( K \). The set of points defined by

\[
    z_j^{(m)} = \varphi(e^{2\pi ij/(m+1)}),
\]

for \( j = 0, \ldots, m \) are called Fejér points. By construction Leja points are recursively defined making them more practical than Fejér points, which have to be recomputed if the number of points \( m \) changes.

Other possibilities for constructing \( \psi \) require knowledge about the region containing the wanted eigenvalues. A Chebyshev polynomial could be constructed [68] if the containment region is a line segment or an ellipse. This idea was based upon the acceleration scheme of Manteuffel [48]. Faber polynomials [20] can be constructed
for polygonal containment regions [32]. This is an adaptation of the hybrid Arnoldi-Faber iterative method proposed by Starke and Varga [86] using a Schwarz-Christoffel conformal mapping [90].

1.5.3 Eigenvector Purification

Generalized eigenvalue problems of the form

$$Ax = Bx\lambda,$$  \hspace{1cm} (1.18)

where $A \in \mathbb{R}^{n \times n}$ and $B \in \mathbb{R}^{n \times n}$ is positive semidefinite, arise in many physical situations. The most common way to solve problem (1.18) is to transform it back to a standard eigenvalue problem using a spectral transformation, like the shift-invert. Then the original problem becomes

$$Sx = x\theta,$$

where $S = (A - \sigma B)^{-1}B$ and $\theta = \frac{1}{\lambda - \sigma}$. This technique works well with both rational Krylov and restarted Arnoldi methods. However, this transformation can cause numerical problems when $B$ is singular.

If $B$ is singular then it is possible for Arnoldi’s method to find approximations to the zero eigenvalues of $S$. These eigenvalues usually correspond to infinite eigenvalues of the generalized eigenvalue problem (1.18), which are often considered uninteresting. However, the eigenvectors corresponding to these uninteresting eigenvalues typically corrupt the invariant subspace corresponding to the nonzero eigenvalues of $S$. The first way to prevent this is to ensure the starting vector, $v_1$, is in $\mathcal{R}(S)$ by replacing $v_1$ with the normalized vector $Sv_1/\|Sv_1\|$. However, the final approximate eigenvector, $x$, may still have components from the zero eigenvalues of $S$ due to cumulative rounding errors from the computation. These can again be purged by replacing $x$ by $Sx$ and normalizing it, which incurs an additional matrix-vector product.

Purification of the computed eigenvectors is an efficient solution for purging these undesirable components and is equivalent to performing a formal step of the power
method with $S$ [18, 59]. Ericsson and Ruhe [18] proposed a purification step for the $B$-orthogonal Lanczos method applied to $S$ when $A$ is symmetric. Given a $k$-step Lanczos factorization for $S$, where $H_k y = y \theta$ and $x = V_k y$

$$Sx = (A - \sigma B)^{-1} B x = V_k H_k y + f e_k^T y = x \theta + f e_k^T y.$$  \hspace{0.5cm} (1.19)

So by replacing the computed eigenvector $x$ with $x + f e_k^T y$ and renormalizing it, any undesirable components have been purged without additional matrix-vector products. Meerbergen and Spence [52] extended this result for the nonsymmetric defective eigenvalue problem, proving that the purification step (1.19) is especially effective when paired with the implicit restarting techniques discussed in the next section.

### 1.5.4 Implicitly Restarted Arnoldi Method

The Implicitly Restarted Arnoldi (IRA) method [85], described in Figure 1.6, provides a numerically stable way to extract the desired eigenvalues and eigenvectors from a high-dimensional Krylov subspace. It implements polynomial restarting by applying a sequence of $p$ implicit QR updates to an $m$-step Arnoldi factorization, where $m = k + p$. These updates apply $p$ exact shifts implicitly to reduce the factorization back to order $k$.

A shift $\mu \in \mathbb{R}$ is applied implicitly using the QR factorization of $H_m - \mu I$,

$$H_m - \mu I = Q_m R_m$$

where $Q_m$ is unitary and $R_m$ is upper triangular. Multiplying the Arnoldi factorization on the right by $Q_m$ results in

$$A(V_m Q_m) = (V_m Q_m) Q_m^H H_m Q_m + f_m e_m^T Q_m.$$  \hspace{0.5cm} (1.20)

Since $H_m$ is Hessenberg, $Q_m$ will also have Hessenberg structure and $e_m^T Q_m$ will have $m - 1$ nonzero entries. Thus, the first $m - 1$ columns of (1.20) are a new Arnoldi factorization. After $p$ shifts have been applied, the new $k$-step Arnoldi factorization
Input: \((A, v_0, m, k)\) where \(p = m - k\)

Output: \((V_k, H_k, f)\) where \(AV_k = V_k H_k + f e_k^T, V_k^H V_k = I_k\), and \(V_k^H f = 0\).

1. Compute an \(m\)-step Arnoldi factorization: \(AV_m = V_m H_m + f e_m^T\)

2. for \(\text{iter} = 1, 2, \ldots, \text{until convergence}\)
   2.1. Compute \(\sigma(H_m)\) and select \(p\) shifts \(\mu_1, \mu_2, \ldots, \mu_p\)
   2.2. \(Q = I_m\)
   2.3. for \(j = 1, 2, \ldots, p\)
       2.3.1. \([Q_j, R_j] = \text{qr}(H_m - \mu_j I)\)
       2.3.2. \(H_m \leftarrow Q_j^H H_m Q_j, Q \leftarrow QQ_j\)
   2.4. end
   2.5. \(\beta_k = H_m(k + 1, k), \sigma_k = Q(m, k)\)
   2.6. \(f \leftarrow v_{k+1} \beta_k + f \sigma_k\)
   2.7. \(V_k \leftarrow V_m Q(:, 1 : k), H_k \leftarrow H_m(1 : k, 1 : k)\)
   2.8. Apply \(p\) additional steps of the Arnoldi process to \(AV_k = V_k H_k + f e_k^T\) to obtain a new \(m\)-step Arnoldi factorization \(AV_m = V_m H_m + f e_m^T\)

3. end

Figure 1.6: The Implicitly Restarted Arnoldi method

is the first \(k\) columns of

\[ A(V_m Q) - (V_m Q) Q^T H_m Q = f e_m^T Q, \]

where \(Q = \prod_{j=1}^p Q_j\). This can be expanded, through \(p\) new steps of the Arnoldi process, back to an \(m\)-step factorization.

Accelerating a Krylov subspace method with a spectral transformation requires a linear system solve for each basis vector of the Krylov subspace. It is preferable to use a direct method for these solves. However, for large problems it is often necessary to use an iterative solver. This also has its drawbacks since each iterative solve must be highly accurate to ensure that a Krylov subspace is being constructed. Usually methods that incorporate an iterative solver with a Krylov subspace method are called
19

i

The inexact Implicitly Restarted Arnoldi method accelerated by a Möbius transformation is effective in practice. The Möbius transformation has properties that make it preferable over shift-invert for linear stability analysis in computational fluid dynamics (CFD) problems [27, 28, 29]. This method has been employed in the stability analysis of a Chemical Vapor Deposition (CVD) reactor simulation with over 4 million variables [45]. More recently, techniques based on these findings have allowed researchers to perform stability analyses on systems with 16 million variables [13].

1.5.5 The Rational Krylov Method

The rational Krylov method [63, 64, 65] is a generalization of shift-invert Arnoldi that allows for the shift to be varied at each step. It also allows the iteration to continue with a linear combination of the basis vectors already computed instead of just the last one. This method builds an orthogonal basis for the \( k \)-dimensional rational Krylov subspace

\[
\text{span}\{w_1, w_2, \cdots, w_k\}, \quad \text{where} \quad w_{j+1} = (A - \sigma_j B)^{-1} B w_j.
\]

The use of several shifts \( \sigma_j \) can improve convergence, but may result in more computationally expensive linear solves.

The rational Krylov iteration illustrated in Figure 1.7 produces a generalized shift-invert Arnoldi factorization. By eliminating \( w \) in the orthogonalization step 2.3, it can be seen that the relationship at step \( j \) is

\[
(A - \sigma_j B)^{-1} B V_j t_j = V_{j+1} \hat{h}_j
\]

where \( \hat{h}_j = \begin{bmatrix} h_j \\ \beta \end{bmatrix} \). Let \( \hat{t}_j = \begin{bmatrix} t_j \\ 0 \end{bmatrix} \), then \( V_{j+1} \hat{t}_j = V_j t_j \) and (1.21) can be rewritten as

\[
A V_{j+1} \hat{h}_j = B V_{j+1}(\sigma_j \hat{h}_j + \hat{t}_j).
\]

Collecting these relations over a sequence of iterations, \( j = 1, 2, \cdots, k \), results in

\[
A V_{k+1} H_{k+1,k} = B V_{k+1} K_{k+1,k},
\]
\textbf{Input:} \((A, B, v_0)\) where \(\|v_0\| = 1\)

1. \(V_1 = v_0\)

2. \textbf{for} \(j = 1, 2, \ldots, \text{until convergence} \)
   
   2.1. Select a shift \(\sigma_j\) and continuation vector \(r = V_j t_j\)
       where \(\|t_j\| = 1\)
   
   2.2. \(w = (A - \sigma_j B)^{-1} Br\)
   
   2.3. \(w \leftarrow w - V_j h_j\), where \(h_j = V_j^H w\)
   
   2.4. \(\beta = \|w\|, v = \frac{w}{\beta}\)
   
   2.5. \(V_{j+1} = [V_j \ v]\)
   
   2.6. \(\hat{h}_j = \left[\begin{array}{c} h_j \\ \beta \end{array}\right], \hat{k}_j = \sigma_j \hat{h}_j + \left[\begin{array}{c} 0 \\ v_j \end{array}\right]\)
   
   2.7. \textbf{if} \(j > 1\) \textbf{then}
       
       \(H_{j+1,j} \leftarrow \left[\begin{array}{c} H_{j,j-1} \hat{h}_j \\ 0 \end{array}\right]\) and \(K_{j+1,j} \leftarrow \left[\begin{array}{c} K_{j,j-1} \hat{k}_j \\ 0 \end{array}\right]\)
       
   \textbf{else}
       
       \(H_{j+1,j} = \hat{h}_j\) and \(K_{j+1,j} = \hat{k}_j\)

   2.8. \textbf{Compute approximate solution and test for convergence}

3. \textbf{end}

Figure 1.7: The Rational Krylov method

where \(H_{k+1,k}\) and \(K_{k+1,k}\) are upper Hessenberg matrices, and

\[K_{k+1,k} = H_{k+1,k} \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_k) + T_{k+1,k}\]

The columns of the upper triangular matrix \(T_{k+1,k}\) contain the continuation vectors from step 2.1. If the shift is held constant, \(\sigma_j = \sigma\), and the continuation vector is the latest vector, \(t_j = e_j\), for all iterations \(j = 1, 2, \ldots, k\), the resulting factorization is the shift-invert Arnoldi

\[(A - \sigma B)^{-1} BV_k = V_{k+1} H_{k+1,k}\]

Extracting approximate eigenvalues and eigenvectors from the rational Krylov subspace can be done in a variety of ways [66]. If the approximate eigenvector is
assumed to have the form \( x \equiv V_{k+1}H_{k+1,k}y \), then \((\theta, x)\) is a Ritz pair for the matrix pencil \((A, B)\) with respect to the subspace \( \mathcal{R}(V_{k+1}H_{k+1,k}) \) if and only if

\[
H_{k+1,k}^H K_{k+1,k} y = \theta H_{k+1,k}^H H_{k+1,k} y. \tag{1.22}
\]

The accuracy of a Ritz pair \((\theta, x)\) is estimated by \( \|Ax - Bx\theta\| \), which is approximately \( \|(K_{k+1,k} - \theta H_{k+1,k})y\| \). Calculating the Ritz values is only one approach to approximating the eigenvalues and eigenvectors from the rational Krylov subspace. This theory can also be extended to compute harmonic Ritz values.

While the rational Krylov method can be very effective, there are many practical implementation issues that need to be addressed. Choosing and deciding when to use a new shift \( \sigma_j \) is subjective. If direct factorizations are used to solve the linear system in step 2.2, then it might be necessary to use the same shift for multiple iterations to reduce the computational cost. If the rational Krylov subspace becomes large, then the cost of orthogonalization becomes computationally expensive. A strategy for locking and purging converged Schur vectors from the rational Krylov subspace has been proposed along with an implicit restarting scheme [66]. Both of these ideas were based on techniques developed for the Arnoldi method [43, 85].

The variation of the shift in the rational Krylov method is very important. If the shift is held constant, the resulting method is the shift-invert Arnoldi method. However, if the shift is varied at every iteration and a direct factorization is used to solve the linear system (2.2), then this method can be computationally expensive. One approach to dealing with this issue is to use the same shift for multiple iterations. Another would be to solve the linear system (2.2) using an iterative method. The later approach, called inexact rational Krylov, is more effective in dealing with large eigenvalue problems.

As opposed to the shift-invert Arnoldi method, the rational Krylov method is not as demanding on the accuracy of the iterative solver. This is because the rational Krylov method does not rely on the correct representation of an underlying Krylov basis. An inexact rational Krylov method accelerated by a Möbius transformation
(IC-RKS) has been proposed by Lehoucq and Meerbergen [42]. This method has shown that a Möbius transformation is more robust than a shift-invert transformation when the linear systems are solved inexactly within the rational Krylov method.

1.5.6 Convergence of Krylov Subspace Methods

Krylov subspace methods are a popular choice for computing a few eigenpairs of large-scale eigenproblems. Since these methods obtain approximations from the Krylov subspace

$$K_k(A, v) = \text{span}\{v, Av, A^2v, \ldots, A^{k-1}v\}$$

there is a question as to how large k must be to compute the eigenpairs of interest to the desired precision. Suppose $A \in \mathbb{R}^{n \times n}$ is an operator that requires $O(n)$ flops to perform the operation $Av$, where $v \in \mathbb{R}^n$. Then the storage of the basis for a $k$-dimensional Krylov subspace is $O(nk)$. Computing the $k^{th}$ basis vector of this sequence requires $O(nk)$ flops, while the computation of the eigensystem of $H_k$ is $O(k^3)$ flops. Clearly the cost of building and storing the Krylov subspace is increasingly more expensive the larger $k$ gets.

Krylov subspace methods are guaranteed to converge in a finite number of steps in exact arithmetic. Unfortunately, the dimension of the Krylov subspace may not be much less than the dimension of the $A$ when the eigenpairs of interest are computed to the desired precision. This is the motivation behind utilizing acceleration techniques like spectral transformations and restarting. However, as useful as they are in practice, understanding the effect these acceleration techniques have on the convergence of Krylov subspace methods is complicated.

The convergence of Krylov subspace methods is currently an active field of research. Recent convergence results utilize the angle between $K_k(A, v)$ and the invariant subspace of $A$ containing the eigenvectors of interest. In particular, these results bound the quantity

$$\delta(W, V) = \sup_{w \in W} \inf_{v \in V} \frac{\|w - v\|}{\|w\|} = \sin(\vartheta_{\text{max}}),$$

(1.24)
also known as the *gap*, to measure the proximity between subspaces $\mathcal{W}$ and $\mathcal{V}$ in $\mathbb{C}^n$. If the dimensions of $\mathcal{V}$ and $\mathcal{W}$ to are allowed to differ, (1.24) is called the *one-sided gap*. The scalar $\vartheta_{\text{max}}$ is the largest canonical angle between $\mathcal{W}$ and a subspace $\hat{\mathcal{V}} \subseteq \mathcal{V}$. When $\dim \mathcal{W} \leq \dim \mathcal{V}$ then $\hat{\mathcal{V}}$ is a subspace of $\mathcal{V}$ “closest” to $\mathcal{W}$ with the same dimension as $\mathcal{W}$. Otherwise, $\delta(\mathcal{W}, \mathcal{V}) = 1$ if the dimension of $\mathcal{V}$ is smaller than the dimension of $\mathcal{W}$.

Recent convergence theory for Krylov subspace methods offer bounds for the gap $\delta(\mathcal{X}_g, \mathcal{K}_k(A, v))$ where $\mathcal{X}_g$ is the desired invariant subspace of $A$. The first result for bounding the gap between $\mathcal{K}_k(A, v)$ and the desired invariant subspace was presented by Saad for a single vector ($\dim(\mathcal{X}_g) = 1$) of a matrix with simple eigenvalues [67]. This bound was generalized by Jia to handle matrices with defective eigenvalues [35].

Beattie, Embree, and Rossi extended the previous convergence theory using functional analysis, pseudospectra, and potential theory [7]. This extension uses the *maximal reachable invariant subspace* which is defined as

$$\mathcal{U}_g := \mathcal{K}(A, P_g v) \subseteq \mathcal{X}_g,$$

where $P_g$ is a spectral projection onto the desired invariant subspace $\mathcal{X}_g$. The subspace $\mathcal{U}_g$ takes into account if any desired eigenvalues are derogatory or the starting vector $v$ is devoid of components from $\mathcal{X}_g$. The convergence theory in [7] provides a bound for the gap between the maximal reachable invariant subspace and a polynomial restarted Krylov subspace, $\delta(\mathcal{U}_g, \mathcal{K}_k(A, \Phi(A)v))$, where $m = \dim(\mathcal{U}_g)$, $\ell \geq m$, and $\Phi$ is the product of all the restart polynomials. Unlike previous bounds, this bound does not impose a restriction on the diagonalizability of $A$.

Beattie, Embree, and Sorensen simplified the gap bound in [7] through a more elementary derivation to obtain a new bound

$$\delta(\mathcal{U}_g, \mathcal{K}_\ell(A, \Phi(A)v)) \leq \left( \max_{\psi \in \Pi_{m-1}} \frac{\|\psi(A)P_b v\|}{\|\psi(A)P_g v\|} \right) \kappa(\Omega_b) \max_{z \in \Omega_b} |1 - \Psi(z)\alpha_g(z)|,$$

(1.25)

where $\ell = 2m$, $P_b$ is a spectral projection onto the undesired invariant subspace $\mathcal{X}_b$,
and $\Omega_b$ is any compact subset of the complex plane that contains all the undesired eigenvalues of $A$. The first term of the bound,

$$\max_{\psi \in \Pi_{m-1}} \frac{\|\psi(A)P_b v\|}{\|\psi(A)P_g v\|},$$

describes the bias in the starting vector toward $U_g$. The second term of the bound, $\kappa(\Omega_b)$, measures the non-normality of $A$ with respect to $U_b := K(A, P_b v)$. The last term of the bound, $\max_{z \in \Omega_b} |1 - \Psi(z)\alpha_g(z)|$, is derived from the roots of $\Phi$. These can be freely chosen with the IRA method, enabling the selection of interpolation points for $\Psi$ that minimize this last term. The gap bound presented in [8] provides insight into the effects of ill-conditioning and instability of eigenvalues, revealing that instability of the undesired eigenvalues impacts convergence.

Convergence theory for inexact Krylov subspace methods has also been recently investigated [9, 10, 11, 79, 80]. Inexact Krylov subspace methods are defined as Krylov subspace methods where the operator-vector product $Av$ cannot be applied exactly, as is the case when $A$ represents an iterative linear solver. Most convergence theory for Krylov subspace methods is based on the numerically exact construction of the subspace

$$K_k(A, v) = \text{span}\{v, Av, A^2v, \cdots, A^{k-1}v\}$$

or the factorization

$$AV_k = V_k H_k + f e_k^T.$$  \hfill (1.27)

An inexact operator-vector product applies the operator $A_j = (A + E_j)$, where $E_j$ is an error matrix that is different for each $j$. Using $A_j$ results in the construction of the subspace

$$K_k(A_j, v) = \text{span}\{v, A_1v, A_2A_1v, \cdots, (\prod_{i=1}^{k-1} A_{k-i})v\}$$

or the factorization

$$[A_1v_1, A_2v_2, \cdots, A_kv_k] = V_k H_k + f e_k^T$$

$$AV_k + [E_1v_1, E_2v_2, \cdots, E_kv_k] = V_k H_k + f e_k^T.$$  \hfill (1.29)
The collective perturbation matrix $[E_1v_1, E_2v_2, \cdots, E_kv_k] \in \mathbb{R}^{n \times k}$ in this factorization prevents any straightforward extension of the previously discussed gap bounds to inexact Krylov subspace methods.

Addressing the effect of $E_j$ on the overall convergence of the iterative method to the eigenvalues of interest makes inexact Krylov subspace methods more analytically complex. Current convergence theory seeks to provide a bound on $\|E_jv_j\|$ that guarantees a residual norm tolerance of the computed eigenpair. Empirical results have shown that the bound on $\|E_jv_j\|$ can be relaxed as the construction of the Krylov subspace progresses without affecting the ability of the eigensolver to converge to the eigenpairs of interest [9, 10, 11]. Simoncini has provided a bound for $\|E_jv_j\|$ that supports this experimental evidence and is inversely proportional to the current residual norm [79]. This bound is sensitive to the conditioning of $A$ and does not address restarting the Krylov subspace.

The convergence theory discussed so far pertains to Krylov subspace methods, sometimes addressing restarting techniques. Lehoucq provides another approach for obtaining convergence theory for the Implicitly Restarted Arnoldi method by relating it to subspace iteration [41]. This is accomplished by exploiting the well-known connection between the QR algorithm [61, 87, 93] and non-stationary subspace iteration to utilize the geometric convergence theory developed by Watkins and Elsner [94].

## 1.6 Subspace Iteration Methods

*Subspace iteration* is another generalization of the power method that computes approximate eigenvalues from the block power sequence

$$S_k, AS_k, A^2S_k, A^3S_k, \cdots \quad (1.30)$$

where $S_k$ is a $k$-dimensional subspace of $\mathbb{R}^n$. Estimates of the approximate eigenvalues are extracted from each subspace $A^iS_k$ through the Rayleigh-Ritz procedure (see Section 1.4). Bauer [6] introduced a variation of subspace iteration called *Treppeniteration*, which operates on a block of independent vectors that span the subspace $A^iS_k$. 
Input: \((A, W \in \mathbb{C}^{n \times k}, \text{tol})\)
Output: \(k\) approximate eigenpairs \((\theta_j, x_j)\) of \(A\).

1. \([V, R] = \text{factor}(W); H = 0\)
2. while \((\|AV - VH\| > \text{tol}\|H\|)\),
   2.1. \(W = AV\)
   2.2. \([V_+, R] = \text{factor}(W)\)
   2.3. \(H = V^H_+ AV_+ / V^H_+ V_+\)
   2.4. \(V \leftarrow V_+\)
3. end
4. Compute \(k\) Ritz pairs \((\theta_j, x_j)\) of \(A\) using \(H\).

Figure 1.8: General subspace iteration

Initially, it was suggested that these vectors be computed through Gaussian transformations, or the LR factorization. For stability reasons, the LR factorization should be replaced by the QR factorization resulting in Simultaneous Iteration methods. A general formulation of subspace iteration can be seen in Figure 1.8.

Most current eigensolvers that are based on subspace iteration use an orthogonal set of vectors \(V\), and can also be considered simultaneous iteration methods. Naturally, as a generalization of the power method, simultaneous iteration methods tend to find the dominant eigenpairs of \(A\). Similar to the power method and Arnoldi’s method, a spectral transformation can be used to compute eigenvalues in other parts of the spectrum. For instance, consider simultaneous iteration with a shift-invert transformation as is described in Figure 1.9. The computation of the Rayleigh-Ritz projection of \(A\) onto the subspace spanned by \(V_+\) can be simplified by combining the
Input: \((A, W \in \mathbb{C}^{n \times k}, \text{tol})\)
Output: \(k\) approximate eigenpairs \((\theta_j, x_j)\) of \(A\).

1. Factor \([V, R] = \text{qr}(W)\)
2. Set \(H = 0\)
3. while \((\|AV - VH\| > \text{tol}\|H\|)\),
   3.1. \(\mu = \text{Select\_shift}(H)\)
   3.2. Solve \((A - \mu I)W = V\)
   3.3. Factor \([V_+, R] = \text{qr}(W)\)
   3.4. \(H = V_+^HVR^{-1} + \mu I\)
   3.5. \(V \leftarrow V_+\)
4. end
5. Compute \(k\) Ritz pairs \((\theta_j, x_j)\) of \(A\).

Figure 1.9: The Shift-Invert Simultaneous Iteration method

exact solve in step 3.2 with 3.3:

\[
\begin{align*}
(A - \mu I)V_+R &= V \\
AV_+ - \mu V_+ &= VR^{-1} \\
V_+^HAV_+ &= V_+^HVR^{-1} + \mu I.
\end{align*}
\]

The Ritz pairs \((\theta_j, x_j)\) can then be computed from \(H\) exactly like those computed in Arnoldi’s method.

The most important observation about any subspace iteration method is that the construction of the subspace \(A^iS_k = \mathcal{R}(V)\) is separate from the construction of the Ritz vectors. This means that the linear solve in step 3.2 can be computed using an iterative method. As a result, the projected matrix \(H\) in step 3.4 would have to be obtained directly by forming \(H = V_+^HAV_+\).

Subspace iteration also has some significant advantages over Krylov subspace projection. Introducing inexact solves in step 3.2 may affect convergence, but the theo-
tical properties of this iteration do not require a set accuracy. This is contrary to the
effect of inexact solves in the Krylov setting. For a sequence of closely related prob-
lems, the entire subspace basis from the previous problem can be used at the initial
basis of the next problem. Meanwhile, Krylov subspace methods need a single start-
ing vector so they can only use a linear combination of these basis vectors. Finally,
constructing vectors to augment the subspace can be done very generally. Instead
of solving the shift-invert equations directly, defect corrections can be constructed to
augment the subspace.

Davidson’s Method [16] is based on this idea. Given a $k$-dimensional subspace
$S_k = \mathcal{R}(V_k)$, where the columns of $V_k$ are orthogonal and $\theta \in \sigma(V_k^H AV_k)$ is the
largest Ritz value of the projected matrix $A$, expand the subspace with a residual
defect correction to improve the approximation of $\theta$. Suppose the Ritz pair $(\theta, \hat{x})$
the current approximation to the eigenpair $(\lambda, x)$ and $(\delta, z)$ is the correction pair that
solves

$$A(\hat{x} + z) - (\theta + \delta)(\hat{x} + z) = 0,$$

(1.32)

where $\hat{x}^H z = 0$. Using the residual $r = A\hat{x} - \theta\hat{x}$, equation (1.32) can be rewritten as

$$(A - \theta I)z = -r + \hat{x}\delta + z\delta.$$  

(1.33)

Davidson suggested approximating $(A - \theta I)$ with $(D_A - \theta I)$, where $D_A$ is the diagonal
of $A$. Furthermore, he chose to ignore the first- and second-order terms on the right-
hand side of equation (1.33). An approximate residual correction $z$ to expand the
subspace $S_k$ can then be obtained by solving

$$(D_A - \theta I)z = -r.$$  

(1.34)

Davidson’s method can be quite successful in finding the dominant eigenvalues
of strongly diagonally dominant matrices. This can be attributed to the fact that
$(D_A - \theta I)^{-1}$ is a good preconditioner for $r$ if $A$ is diagonally dominant. However, if
$(D_A - \theta I)$ is replaced with $(A - \theta I)$ in equation (1.34) and exactly solved, then $z = \hat{x}$. 
This solution does not expand the search space, contradicting the idea that solving equation (1.34) is a preconditioning step.

1.6.1 Jacobi-Davidson

Another approach to solving equation (1.33) for the correction pair \((\delta, z)\) involves dropping the second-order term \(z\delta\) and solving a bordered linear system

\[
\begin{bmatrix}
A - \theta I & \hat{x} \\
\hat{x}^H & 0
\end{bmatrix}
\begin{bmatrix}
z \\
-\delta
\end{bmatrix}
= \begin{bmatrix}
-r \\
0
\end{bmatrix}.
\] (1.35)

This linear system can be interpreted as a Newton correction to the current approximation \((\lambda, \hat{x})\) to the solution of the nonlinear equation (1.32). Once \(z\) is computed, a subspace can be formed using the previous subspace \(S_k\) and the correction vector, \(S_{k+1} = \text{span}\{V_k, z\}\). Then \(S_{k+1}\) can be used as the new subspace from which an improved approximation to the solution of (1.32) can be computed. This second-order accurate approach for computing the correction pair is known as the Jacobi-Davidson method.

Sleijpen and van der Vorst [82] proposed rewriting the nonlinear problem (1.32) as a restriction of the residual correction to the orthogonal complement of the existing space. Consider the projection onto the orthogonal complement of the current Ritz vector \((I - \hat{x}\hat{x}^H)\) and the current Ritz value as a Rayleigh quotient \(\theta = \hat{x}^H A \hat{x}\). If the correction equation (1.33) is multiplied on the left by this projection and \(\hat{x}^H z = 0\), the resulting equation is

\[
(I - \hat{x}\hat{x}^H)(A - \theta I)(I - \hat{x}\hat{x}^H)z = (I - \hat{x}\hat{x}^H)(-r + \hat{x}\delta + z\delta)
\]

\[
= -r + z\delta
\]

\[
\approx -r
\] (1.36)

Only the second-order term \(z\delta\) appears in the equality (1.36), proving that this formulation is also a second-order correction and \(z = (I - \hat{x}\hat{x}^H)z\) is constructed that is
\textbf{Input:} \((A, v_0, \text{tol})\)  
\textbf{Output:} Approximate eigenpair \((\theta, x)\) such that \(\|Ax - x\theta\| < \text{tol} \).

1. \(x = v_0 / \|v_0\|, w = Ax, \theta = x^H w\)
2. \(H_1 = [\theta], V_1 = [x], r = w - \theta x\)
3. while \((\|r\| > \text{tol})\),
   
   3.1. Solve (approximately) for \(z \perp x\)
      \[ (I - xx^H)(A - \theta I)(I - xx^H)z = -r \]
   3.2. \(c = V_j^H z, z = z - V_j c\)
   3.3. \(v_{j+1} = z / \|z\|, V_{j+1} = [V_j \ v_{j+1}]\)
   3.4. \(w = Av_{j+1}\)
   3.5. \[ \begin{bmatrix} h \\ \alpha \end{bmatrix} = V_{j+1}^H w, H_{j+1} = \begin{bmatrix} H_j & h \\ h^H & \alpha \end{bmatrix} \]
   3.6. Compute all eigenpairs of \(H_{j+1}\), select desired pair \((\theta, y)\).
   3.7. \(x \leftarrow V_{j+1} y\)
   3.8. \(r = Ax - x\theta\)
4. end

Figure 1.10: The Jacobi-Davidson method

orthogonal to \(\hat{x}\). Since the solution of the correction equation is the most expensive part of Jacobi-Davidson, it is common practice to employ an iterative linear solver for (1.36).

The Jacobi-Davidson method is presented in Figure (1.10) for a general matrix \(A\). The projected correction equation (3.1.1) is solved using a preconditioned iterative method. This correction is then used to expand the search space. The presented algorithm expands the subspace until convergence, which is not computationally practical. Restarting techniques are usually employed to keep the memory usage down. Furthermore, deflation techniques allow this method to find more than one eigenpair by forcing the subspace to be orthogonal to the converged eigenvectors. \(JDQR\) and \(JDQZ\) [24] are implementations of the Jacobi-Davidson method that employ restarting...
and deflation in solving standard and generalized eigenvalue problems, respectively.

1.6.2 Locally-Optimal Block Preconditioned Conjugate Gradient

LOBPCG [38] is a method for computing a few eigenvalues of a Hermitian matrix, and their corresponding eigenvectors. Based on the local optimization of a three-term recurrence, this method is best understood through the introduction of preconditioned simultaneous iteration. This general framework requires an iterative solver like the preconditioned Conjugate Gradient (PCG) method to enlarge the trial subspace for updating the current eigenvector approximations. The scalar iteration parameters needed to compute this update are then chosen to be locally optimal using the Rayleigh-Ritz procedure.

Consider the generalized eigenvalue problem

\[ Ax = Bx\lambda, \quad (1.37) \]

where \( A \) is Hermitian positive definite. If \( B \) is not positive definite, then some of the eigenvalues of the generalized eigenvalue problem (1.37) may be infinite. In this case, an alternative form of the generalized eigenvalue problem is considered,

\[ Bx = Ax\mu. \quad (1.38) \]

The infinite eigenvalues of the original problem (1.37) are now the zero eigenvalues of the alternative form (1.38). The alternative form of the eigenvalue problem will be used in this section for the development of the LOBPCG method. Furthermore, the matrix \( M \) will refer to a preconditioning matrix that approximates \( A^{-1} \), like those used with preconditioned iterative solvers.

Preconditioned Simultaneous iteration can be defined as a generalized polynomial method that treats a group of vectors simultaneously

\[ x_j^{(k)} = P_{mk}(M^{-1}A, M^{-1}B)x_j^{(0)}, \quad j = 1, \ldots, m, \quad (1.39) \]
Input: \((A, B, M, x_1^{(0)}, x_2^{(0)}, \ldots, x_m^{(0)})\)

Output: Approximate eigenpairs \((\mu_j^{(k)}, x_j^{(k)})\) of the largest eigenvalues and corresponding eigenvectors of \(Bx = Ax\mu\).

1. for \(i = 1, 2, \ldots\), until convergence
   1.1. Select \(x_j^{(0)}, j = 1, \ldots, m\)
   1.2. Iterate \(k\) steps to compute \(\hat{x}_j^{(k)} = P_{m_k} (M^{-1}A, M^{-1}B)x_j^{(0)}, j = 1, \ldots, m\).
   1.3. Use the Rayleigh-Ritz procedure on \(B - \mu A\) with the subspace \(S^{(k)} = \text{span}\{\hat{x}_1^{(k)}, \ldots, \hat{x}_m^{(k)}\}\) to compute Ritz pairs \((\mu_j^{(k)}, x_j^{(k)})\).

2. end

Figure 1.11: The block preconditioned eigensolver

where \(P_{m_k}\) is a polynomial of degree \(m_k\) and \(x_j^{(0)}\) are the initial vectors. The polynomial \(P_{m_k}\) does not have to be the same for different values of \(j\). When it is, preconditioned simultaneous iteration (1.39) is equivalent to subspace iteration. A recursive procedure (Figure 1.11) can be developed using information from the subspace spanned by the group of vectors \(x_j^{(k)}\),

\[ S^{(k)} = \text{span}\{x_1^{(k)}, x_2^{(k)}, \ldots, x_m^{(k)}\}, \]

to construct new initial vectors for simultaneous iteration. Usually, the Rayleigh-Ritz procedure is used to extract information from the subspace \(S^{(k)}\).

Under the general framework of block preconditioned eigensolvers (Figure 1.11), the defining factor for any method is the polynomial used in step 1.2. One option is to use the polynomial constructed through the preconditioned Conjugate Gradient (PCG) method. For finding the smallest eigenvalues of \(A - \lambda B\), it is commonly implemented using two linked two-term recurrences

\[ p^{(i)} = w^{(i)} + \beta^{(i)} p^{(i-1)}, \quad x^{(i+1)} = x^{(i)} + \alpha^{(i)} p^{(i)}, \]

where the scalar \(\alpha^{(i)}\) is chosen using a line search to minimize the Rayleigh quotient
Input: \((A, B, M, x_1^{(0)}, x_2^{(0)}, \ldots, x_m^{(0)})\)

Output: Approximate eigenpairs \((\mu_j^{(k)}, x_j^{(k)})\) of the largest eigenvalues and corresponding eigenvectors of \(Bx = Ax\mu\).

1. Select \(x_j^{(0)}\) and set \(p_j^{(0)} = 0, j = 1, \ldots, m\)

2. for \(i = 1, 2, \ldots, \text{until convergence} \)
   \[\mu_j^{(i)} = ((x_j^{(i)})^H B x_j^{(i)})/((x_j^{(i)})^H A x_j^{(i)}), j = 1, \ldots, m\]
   \[r_j = B x_j^{(i)} - \mu_j^{(i)} A x_j^{(i)}, j = 1, \ldots, m\]
   \[w_j^{(i)} = M^{-1} r_j, j = 1, \ldots, m\]
   \[x_j^{(i+1)} = \sum_{k=1}^{m} \alpha_k^{(i)} w_k^{(i)} + \tau_k^{(i)} x_k^{(i)} + \gamma_k^{(i)} p_k^{(i)}, j = 1, \ldots, m\]

   (the \(j\)-th Ritz vector corresponding to the \(j\)-th largest Ritz value)

2.6. \(p_j^{(i+1)} = \sum_{k=1}^{m} \alpha_k^{(i)} w_k^{(i)} + \gamma_k^{(i)} p_k^{(i)}\)

3. end

Figure 1.12: The Locally-Optimal Block Preconditioned Conjugate Gradient method

of \(x^{(i+1)}\) and \(\beta^{(i)}\) is computed to make the directions \(p^{(i)}\) conjugate. LOBPCG (Figure 1.12) uses a variant of the PCG polynomial to find the largest eigenvalues of \(B - \mu A\),

\[p^{(i)} = x^{(i)} - \tau^{(i-1)} x^{(i-1)}, \quad x^{(i+1)} = \alpha^{(i)} w^{(i)} + \tau^{(i)} x^{(i)} + \gamma^{(i)} p^{(i)},\]

where \(w^{(i)} = M^{-1}(B x^{(i)} - \mu^{(i)} A x^{(i)})\) and \(\mu^{(i)} = ((x^{(i)})^H B x^{(i)})/((x^{(i)})^H A x^{(i)})\). The scalars \(\tau^{(i)}\) and \(\gamma^{(i)}\) are chosen to maximize the Rayleigh quotient \(x^{(i)}\) using the Rayleigh-Ritz procedure. This method exploits the fact that the Rayleigh-Ritz procedure is an inexpensive way to compute the local minimization of the Rayleigh quotient for larger dimensional subspaces.
Chapter 2

IRA with a Fixed-Polynomial Operator

Spectral transformations are an effective technique for accelerating the convergence of the Implicitly Restarted Arnoldi (IRA) method because they map the eigenvalues of interest from the matrix pencil \((A, B)\) to dominant, well separated eigenvalues of \(\psi(A; B)\). Using a spectral transformation with an Arnoldi method requires the construction of the Krylov subspace

\[
\mathcal{K}_k(\psi(A; B), v) = \text{span}\{v, \psi(A; B)v, \psi^2(A; B)v, \cdots, \psi^{k-1}(A; B)v\}. \tag{2.1}
\]

For the shift-invert spectral transformation (1.15), the Krylov subspace is (2.1) with

\[
\psi(A; B) = \psi_{SI}(A; B) = (A - \sigma B)^{-1}B.
\]

While for the Möbius spectral transformation (1.16), the Krylov subspace is (2.1) with

\[
\psi(A; B) = \psi_M(A; B) = (A - \sigma B)^{-1}(A - \mu B).
\]

The computation of basis vectors for both these Krylov subspaces involves the solution to the linear system

\[
(A - \sigma B)x = b. \tag{2.2}
\]
There are two approaches for solving a linear system (2.2) in the construction of a Krylov subspace: direct methods and iterative methods. Direct methods are the preferred technique because convergence to the eigenvalues of interest would be achieved in very few steps. Furthermore, a direct factorization of \( A \) only needs to be computed once. It can then be applied with very little cost each step of the Arnoldi process. Unfortunately, direct methods are not practical for large-scale eigenvalue problems due to storage constraints and computational cost. The more practical alternative is to use an iterative method for these linear solves. The approximate solution obtained from \( j \) steps of any polynomial iterative method can be viewed as the application of a polynomial of \((A - \sigma B)\) to \( b \)

\[
\hat{x} = \phi_j(A - \sigma B)b,
\]

where \( \phi_j \in \Pi_j \). The dimension of this polynomial may be different for each basis vector of the Krylov subspace depending on the right-hand side \( b \) and requested accuracy of the approximate solution. Thus, iterative methods can provide only approximations of the shift-invert or Möbius spectral transformations, resulting in the construction of the subspace

\[
S_k = \text{span}\{v, \psi_{j_1}(A; B)v, \psi_{j_2}(A; B)v, \cdots, \psi_{j_{k-1}}(A; B)v\}, \quad (2.3)
\]

where

\[
\psi_{j_i}(A; B) = \begin{cases} 
\prod_{k=1}^i \phi_{j_k}(A - \sigma B)B & \text{for } \psi_{SI}(A; B) \\
\prod_{k=1}^i \phi_{j_k}(A - \sigma B)(A - \mu B) & \text{for } \psi_M(A; B).
\end{cases}
\]

Unless stringent accuracy is specified, there are no guarantees that (2.3) is even close to a Krylov subspace.

This thesis will consider an alternative approach that employs a fixed-polynomial operator to approximate any spectral transformation that requires a solution to (2.2). We define a fixed-polynomial operator as a polynomial \( \phi_j \in \Pi_j \) that approximates the solution to (2.2) through the operation

\[
\hat{x} = \phi_j(A - \sigma B)b.
\]
This polynomial is constructed prior to any eigenvalue computation and only applied once, wherever a solution to (2.2) is required. Incorporating a fixed-polynomial operator into the shift-invert and Möbius spectral transformations gives approximations of the form

\[
\psi_{\phi,j}(A; B) = \begin{cases} 
\phi_j(A - \sigma B)B & \text{for } \psi_{SI}(A; B) \\
\phi_j(A - \sigma B)(A - \mu B) & \text{for } \psi_M(A; B).
\end{cases}
\]

Since the polynomial \( \phi_j(A - \sigma B) \) is the same for any right-hand side \( b \), we can guarantee a Krylov subspace of \( \psi_{\phi,j}(A; B) \) has been constructed:

\[
K_k(\psi_{\phi,j}(A; B), v) = \text{span}\{v, \psi_{\phi,j}(A; B)v, \psi_{\phi,j}^2(A; B)v, \ldots, \psi_{\phi,j}^{k-1}(A; B)v\}. \quad (2.4)
\]

In this thesis we shall study the combination of fixed-polynomial operators with the Implicitly Restarted Arnoldi method. The polynomials presented in this thesis will be constructed using some common iterative methods for non-Hermitian linear systems: GMRES [71], Bi-Conjugate Gradient stabilized (Bi-CGSTAB) [91], and transpose-free QMR (TFQMR) [26]. These methods will be discussed in Section 2.1 and the construction of their associated fixed-polynomial operators will follow in Section 2.2. The eigenpairs of the original problem can be easily reconstructed from the Ritz vectors of the Krylov subspace (2.4), as discussed in Section 2.3. The accuracy and convergence heuristics presented in Section 2.4 will show that the accuracy of these eigenpairs is dependent upon how well \( \psi_{\phi,j}(A; B) \) approximates the intended spectral transformation. Finally, a comparison of these three fixed-polynomial operators will be presented in Section 2.5.

### 2.1 Iterative Methods for non-Hermitian Linear Systems

Consider the linear system

\[ Ax = b \]  \hfill (2.5)
where $A \in \mathbb{R}^{n \times n}$ is a non-Hermitian matrix. Most iterative methods for solving this linear system compute an approximate solution $\hat{x}$ in a subspace $\mathcal{K}$. In order for this approximation to be well defined, constraints must be imposed equivalent in number to the dimension of the subspace $\mathcal{K}$. This is normally accomplished by requiring that the residual

$$r = b - A\hat{x}$$

be orthogonal to a subspace $\mathcal{L}$. The combination of these two particular projection and constraint conditions

$$\hat{x} \in \mathcal{K} \quad \text{s.t.} \quad b - A\hat{x} \perp \mathcal{L}$$

is commonly known as the Petrov-Galerkin condition. If an initial guess, $x_0$, for the solution to the linear system can be provided, then the search subspace $\mathcal{K}$ can be replaced by the affine subspace $x_0 + \mathcal{K}$.

The choice of subspaces $\mathcal{K}$ and $\mathcal{L}$ divides methods based on the Petrov-Galerkin conditions into two groups: orthogonal and oblique projection methods. Orthogonal projection methods require the constraint subspace $\mathcal{L}$ to be the same as the search space $\mathcal{K}$. This reduces the conditions (2.6) to

$$\hat{x} \in \mathcal{K} \quad \text{s.t.} \quad b - A\hat{x} \perp \mathcal{K}$$

which is known as the Galerkin condition. Oblique projection methods are the result of the search space $\mathcal{K}$ and the constraint subspace $\mathcal{L}$ not being equal.

A practical algorithm for computing the solution of the linear system (2.5) using either an orthogonal or oblique projection method will use a vector basis for the search and constraint subspaces. Let $V_k = [v_1, v_2, \ldots, v_k] \in \mathbb{R}^{n \times k}$ and $W_k = [w_1, w_2, \ldots, w_k] \in \mathbb{R}^{n \times k}$ be the vector representation of the basis for the $k$-dimensional subspaces $\mathcal{K}$ and $\mathcal{L}$, respectively. Then any approximate solution to the linear system (2.5) can be written as the initial guess plus a linear combination of the search space vectors

$$\hat{x}_k = x_0 + V_k y.$$  (2.7)
In this thesis we will consider projection methods where the search subspace $\mathcal{K}$ is a Krylov subspace

$$\mathcal{K}_k(A, v) = \text{span}\{v, Av, A^2v, \cdots, A^{k-1}v\}.$$ 

With this choice of subspace, the approximate solution (2.7) takes a more interesting form. If the initial residual, $r_0 = b - Ax_0$, is used as the starting vector for the Krylov subspace then the approximate solution (2.7) can be written as

$$\hat{x}_k = x_0 + \phi_{k-1}(A)r_0,$$

where $\phi_{k-1} \in \Pi_{k-1}$ is called the iteration polynomial. Now the residual at step $k$ can be written as

$$r_k = b - A\hat{x}_k = b - A(x_0 + \phi_{k-1}(A)r_0) = r_0 - A\phi_{k-1}(A)r_0 = \psi_k(A)r_0,$$  \hspace{1cm} (2.8)

where $\psi_k(\tau) = 1 - \tau\phi_{k-1}(\tau) \in \Pi_k$ is the residual polynomial.

The goal of any projection method is to compute a solution $\hat{x}_k$ from the search space such that its corresponding residual $r_k$ is small in norm. When using a Krylov subspace, the residual polynomial provides a means for bounding the norm $\|r_k\|$. By taking the norm of both sizes of the equation (2.8), it is easily seen that $\|\psi_k(A)\|$ is a bound on the size of $\|r_k\|$. For $\|\psi_k(A)\|$ to be small, the magnitude of $\psi_k$ must be small over some region in the complex plane containing the spectrum of $A$, $\sigma(A)$. Suppose $\|\psi_k(A)\| = 0$; then

$$A\phi_{k-1}(A) = I,$$

which means that $A^{-1} = \phi_{k-1}(A)$. So, for any $\lambda_j \in \sigma(A)$,

$$\phi_{k-1}(\lambda_j) = \frac{1}{\lambda_j}.$$

Now suppose that $\|\psi_k(A)\|$ is small; then it may be assumed that $\phi_{k-1}(\lambda_j) \approx 1/\lambda_j$ for any $\lambda_j \in \sigma(A)$, which means that $\phi_{k-1}(A) \approx A^{-1}$. This makes the iteration
polynomial \( \phi_{k-1} \) a good candidate for a fixed-polynomial operator to approximate \( A^{-1} \).

If the spectrum of \( A \) is tightly clustered, then \( \| \psi_k(A) \| \) can be small for a low-degree polynomial. Unfortunately, most matrices do not have a tightly clustered spectrum so preconditioning must be employed to enable the construction of a low-degree polynomial \( \psi_k \) such that \( \| \psi_k(A) \| \) is small. Right preconditioning modifies the original linear system (2.5) by multiplying \( A \), on the right, by a matrix \( M^{-1} \) which gives

\[
AM^{-1}\hat{x} = b.
\]

To obtain the approximate solution to the original system one preconditioner solve \( Mx = \hat{x} \) is required. The residual (2.8) can now be written as

\[
r_k = (I - AM^{-1}\phi_{k-1}(AM^{-1}))r_0 = \psi_k(AM^{-1})r_0.
\]

The norm of the residual polynomial \( \psi_k(AM^{-1}) \) can be made small if the spectrum of \( AM^{-1} \) is tightly clustered. This requires a decent preconditioner for \( A \) that has similar spectral properties, resulting in \( AM^{-1} \) whose spectrum consists of a few sets of tightly clustered eigenvalues. Ideally the spectrum of the preconditioner \( M \) would closely approximate the spectrum of \( A \). If there was a preconditioner such that \( \| \psi_k(AM^{-1}) \| \) is small, then \( \| I - AM^{-1}\phi_{k-1}(AM^{-1}) \| \) is small. Using the earlier argument, it can be concluded that

\[
M^{-1}\phi_{k-1}(AM^{-1})
\]

is a good candidate for a fixed-polynomial operator to approximate \( A^{-1} \).

In the following sections we will present three projection methods whose iteration polynomials will be employed as a fixed-polynomial operator for approximating \( A^{-1} \): Generalized Minimum Residual (GMRES) [71], Bi-Conjugate Gradient stabilized (Bi-CGSTAB) [91], and transpose-free QMR (TFQMR) [26]. From the general discussion it may be presumed that each of these Krylov subspace projection methods will
construct the same iteration polynomial. However, the iteration polynomials for each of these methods will differ based on the influence of the constraint subspace $\mathcal{L}$.

### 2.1.1 Generalized Minimum Residual Method

The Generalized Minimum Residual (GMRES) method [71] is an oblique projection method where the constraint space is $A\mathcal{K}_k(A,r_0)$. More precisely, the GMRES method computes approximate solutions $\hat{x}$ to $Ax = b$ of the form

$$\hat{x} \in x_0 + \mathcal{K}_k(A,r_0) \quad \text{s.t.} \quad b - A\hat{x} \perp A\mathcal{K}_k(A,r_0).$$

This particular choice of the constraint space results in a residual norm minimization for all approximate solutions in the search space $x_0 + \mathcal{K}_k(A,r_0)$.

Arnoldi’s method [2] is used to compute an orthonormal basis $V_k$ for the GMRES search space $\mathcal{K}_k(A,r_0)$. The computation of this orthonormal basis results in the Arnoldi factorization

$$AV_k = V_k H_k + f e_k^T = V_{k+1} \bar{H}_k$$

(2.9)

where $H_k$ is an upper-Hessenberg matrix and $\bar{H}_k$ is an augmented upper-Hessenberg matrix (see Section 1.5.1). The approximate solution $\hat{x}_k$ can then be written as a linear combination of these basis vectors

$$\hat{x}_k = x_0 + V_k y_k, \quad y_k \in \mathbb{R}^k.$$

(2.10)

Using the fact that $v_1 = r_0/\|r_0\|$, the residual for this approximate solution can be written as

$$r_k = b - A\hat{x}_k = r_0 - AV_k y_k = \beta_0 v_1 - V_{k+1} \bar{H}_k y_k = V_{k+1}(\beta_0 e_1 - \bar{H}_k y_k)$$

where $\beta_0 = \|r_0\|$. The norm of this residual is

$$\|r_k\| = \|V_{k+1}(\beta_0 e_1 - \bar{H}_k y_k)\| = \|\beta_0 e_1 - \bar{H}_k y_k\|$$

(2.11)
since the columns of $V_{k+1}$ are orthonormal. Minimizing the residual norm of an overdetermined system like (2.11) is equivalent to solving the least-squares problem
\[
\| r_k \| = \min_{y \in \mathbb{R}^k} \| \beta_0 e_1 - \bar{H}_k y \|.
\] (2.12)
The minimizer $y_k \in \mathbb{R}^k$ for the least squares problem (2.12) gives the weights for computing the approximate solution (2.10).

The GMRES iteration polynomial $\phi_{k-1}$ can be implicitly constructed through the $k$-step Arnoldi factorization (2.9) and least squares problem (2.12). We will demonstrate the construction of this polynomial for the right-preconditioned linear system (2.9). Without loss of generality, we can assume the zero vector as an initial guess and $\| b \| = 1$; then $r_0 = b$ and
\[
M\hat{x}_k \in \mathcal{K}_k(AM^{-1}, b).
\]
The approximate solution (2.10) can be written in terms of the Krylov orthogonal basis $V_k$ for $\mathcal{K}_k(AM^{-1}, b),$
\[
\hat{x}_k = M^{-1}V_k y
\]
for some $y = [\eta_1, \eta_2, \cdots, \eta_k]^T \in \mathbb{R}^k$. Steps 3.3-3.6 of the GMRES algorithm (Algorithm 1) give an equation for the computation of each Krylov basis vector
\[
v_{j+1} = \frac{1}{\gamma_{j+1,j}}[AM^{-1}v_j - \sum_{i=1}^{j} v_i \gamma_{i,j}] = \hat{\phi}_j(AM^{-1})b,
\]
for $j = 1, 2, \cdots, k-1$, where $\gamma_{i,j} = H_k(i,j)$. This equation can be used to reformulate the approximate solution as
\[
\hat{x}_k = M^{-1}\phi_{k-1}(AM^{-1})b,
\]
where the GMRES iteration polynomial is
\[
\phi_{k-1}(z) = \sum_{j=0}^{k-1} \eta_{j+1} \hat{\phi}_j(z).
\]
Reconstruction of this polynomial can be accomplished using the coefficients of the upper-Hessenberg matrix $H_k$ and the weights for the Krylov basis vectors $y_k$ as will be shown in Section 2.2.1.

Algorithm 1: Right-preconditioned GMRES

**Input:** $(A, M, b, x_0, m)$  
**Output:** $\hat{x}_k$

1. $r_0 = b - Ax_0$, $\beta_0 = \|r_0\|$, $v_1 = r_0/\beta_0$, and $k = m$
2. $V_1 = [v_1]$, $\bar{H}_0 = []$, and $\bar{z}_1 = [\beta_0 0]^T$
3. for $j = 1, 2, \cdots, m$
   3.1. $w = AM^{-1}v_j$, $h = V_j^H w$
   3.2. $f = w - V_j h$
   3.3. $H_j \leftarrow [\bar{H}_{j-1} h]$
   3.4. $\beta_j = \|f\|$
   3.5. $\bar{H}_j \leftarrow \begin{bmatrix} H_j \\ \beta_j e_j^T \end{bmatrix}$
   3.6. If converged, $k = j$ break.
   3.7. $v_{j+1} = f/\beta_j$, $V_{j+1} = [V_j \, v_{j+1}]$
4. end
5. Compute the $y_k$ that minimizes $\|\beta_0 e_1 - \bar{H}_k y\|$, $\hat{x}_k = x_0 + M^{-1} V_k y_k$.

The GMRES algorithm given from (2.10)-(2.11) is illustrated by Algorithm 1. This implementation requires the explicit computation of the approximate solution $\hat{x}_k$ or residual $r_k$ to determine if convergence has been achieved at step 3.6. Obtaining $\hat{x}_k$ or $r_k$ involves the solution of the least-squares problem (2.12) at each step $k$, which costs $O(k^2)$ flops for an upper-Hessenberg matrix $\bar{H}_k$. Practical implementations of GMRES seek to avoid this expense by cleverly integrating the solution of the least-squares problem with the construction of the Krylov subspace $K_k(A, r_0)$ [71] to
obtain the residual norm at each iteration. We will spend the rest of this subsection
discussing a practical implementation of GMRES, ending with the algorithm used in
our numerical experiments.

The standard approach for solving the least squares problem (2.12) involves a QR
factorization of $H_k$

$$Q_{k+1}^H H_k = \begin{bmatrix} R_k \\ 0 \end{bmatrix},$$

(2.13)

where $Q_{k+1} \in \mathbb{R}^{(k+1) \times (k+1)}$ is an orthogonal matrix and $R_k \in \mathbb{R}^{k \times k}$ is an upper-
triangular matrix (see Chap. 5 of [30] or Lect. 11 of [89]). This factorization is used
to solve a least squares problem

$$\| r_k \|^2 = \min_{y \in \mathbb{R}^k} \| \beta_0 e_1 - H_k y \|^2,$$

(2.14)

which has a solution equivalent to that of (2.12). Applying $Q_{k+1}^H$ to $\beta_0 e_1$ yields

$$Q_{k+1}^H (\beta_0 e_1) = \bar{z}_k = \begin{bmatrix} z_k \\ \bar{\zeta}_{k+1} \end{bmatrix},$$

(2.15)

where $z_k = [\zeta_1, \cdots, \zeta_k]^T \in \mathbb{R}^k$ and $\bar{\zeta}_{k+1} \in \mathbb{R}$. Then the function being minimized in
(2.14) can be rewritten using (2.13) and (2.15) as

$$\| Q_{k+1}^H \beta_0 e_1 - Q_{k+1}^H H_k y \|^2 = \left\| \begin{bmatrix} z_k \\ \bar{\zeta}_{k+1} \end{bmatrix} - \begin{bmatrix} R_k \\ 0 \end{bmatrix} y \right\|^2 = \| z_k - R_k y \|^2 + | \bar{\zeta}_{k+1} |^2 \quad (2.16)$$

(2.17)

The norm of the first term in (2.16) is zero because $H_k$ is an unreduced upper-
Hessenberg matrix, thus $R_k$ is a full rank upper-triangular matrix and the linear
system

$$R_k y = z_k.$$
has an exact solution. Now the least squares problem (2.14) can be simplified using (2.17) to

\[ \| r_k \|^2 = \min_{y \in \mathbb{R}^k} \| Q_{k+1}^H \beta_0 e_1 - Q_{k+1}^H \bar{H}_k y \|^2 \\
= |\bar{\zeta}_{k+1}|^2, \]

which also simplifies the original least squares problem (2.12) to

\[ \| r_k \| = \min_{y \in \mathbb{R}^k} \| \beta_0 e_1 - \bar{H}_k y \| \\
= |\bar{\zeta}_{k+1}|. \quad (2.18) \]

The simplest way to compute the QR factorization of an upper-Hessenberg matrix (2.13) is to use plane rotations to zero out the subdiagonal elements. However, if the QR factorization of \( \bar{H}_k \) can be computed progressively as each new column appears from the Arnoldi process, then by (2.18) the residual norm can be obtained at each iteration. This approach was first realized by Paige and Saunders who used Givens rotations to progressively reduce the tridiagonal matrix computed by the Lanczos method in the LSQR algorithm [60] for solving Hermitian linear systems. Saad and Schultz extended this approach for GMRES, enabling the progressive reduction of the upper-Hessenberg matrix computed by the Arnoldi method [71].

Givens rotations are transformations that rotate a vector through an angle \( \theta \) to zero out a single entry. In matrix form they are rank-two corrections of the identity matrix,

\[ G(i, j, \theta) = \begin{bmatrix}
1 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
0 & \cdots & c & \cdots & s & \cdots & 0 \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
0 & \cdots & -s & \cdots & c & \cdots & 0 \\
\vdots & \ddots & \vdots & \ddots & \vdots \\
0 & \cdots & 0 & \cdots & 0 & \cdots & 1 \\
i & \cdots & i \\
j & \cdots & j
\end{bmatrix} \] \quad (2.19)
where \( c = \cos(\theta) \) and \( s = \sin(\theta) \) for some angle of rotation \( \theta \) in the \((i,j)\) coordinate plane (see Chap. 5 of [30]). Givens rotations are ideal for reducing upper-Hessenberg matrices to upper-triangular form because they can efficiently zero out one subdiagonal element at a time. This also makes them ideal for the progressive reduction of an upper-Hessenberg matrix.

Assume at step \( k + 1 \) of the Arnoldi process that we have the Givens rotations, \( G(i, i+1, \theta_i) \) for \( i = 1, \cdots, k \), that produce a QR factorization of \( \tilde{H}_k \) (2.13), where

\[
Q_{k+1}^H = G(k, k+1, \theta_k)^H G(k, k-1, \theta_{k-1})^H \cdots G(1, 2, \theta_1)^H. \tag{2.20}
\]

The upper Hessenberg matrix computed at step \( k \) is of the form

\[
\tilde{H}_{k+1} = \begin{bmatrix}
\tilde{H}_k & h_{k+1} \\
0 & \ddots & \beta_{j+1} \\
\end{bmatrix},
\]

Embedding the current product of Givens rotations (2.20) into a \((k + 2) \times (k + 2)\) identity matrix

\[
\check{Q}_{k+2}^H = \begin{bmatrix}
Q_{k+1}^H & 0 \\
\vdots & \ddots & \ddots & \ddots \\
0 & \cdots & 0 & 1 \\
\end{bmatrix}
\]

and applying it to \( \tilde{H}_{k+1} \) results in

\[
\check{Q}_{k+2}^H \tilde{H}_{k+1} = \begin{bmatrix}
R_k & \hat{\tilde{r}}_{k+1} \\
0 & \hat{\beta}_{j+1} \\
\end{bmatrix},
\]

where \( \hat{\tilde{r}}_{k+1} = \check{Q}_{k+2}^H h_{k+1} \). The principle \( k \times k \) submatrix of \( \check{Q}_{k+2}^H \tilde{H}_{k+1} \) is in upper-triangular form through the construction of the previous Givens rotations. Only one
Input: \((A, M, b, x_0, m, \epsilon)\)
Output: \(\hat{x}_k\)

1. \(r_0 = b - Ax_0, \beta_0 = \|r_0\|, v_1 = r_0/\beta_0, \text{ and } k = m\)
2. \(V_1 = [v_1], \bar{R}_0 = [], Q_0^H = 1, \text{ and } \bar{z}_1 = [\beta_0 0]^T\)
3. for \(j = 1, 2, \cdots, m\)
   3.1. \(w = AM^{-1}v_j, h = V^H_jw\)
   3.2. \(f = w - V_jh\)
   3.3. Update \(h\) using previous Givens rotations \(\hat{r} = Q^H_{j-1}h\)
   3.4. \(R_j \leftarrow [\bar{R}_{j-1} \hat{r}]\)
   3.5. \(\beta_j = \|f\|\)
   3.6. \(\bar{R}_j \leftarrow \begin{bmatrix} R_j & 0 \\ \beta_j e_j^T \end{bmatrix}\)
   3.7. Compute Givens rotation \(G(j, j + 1, \theta_j)\) to zero out \(\beta_j\) in \(\bar{R}_j\)
   3.8. Update the vector \(\bar{z}_j\) using (2.21)
   3.9. If \(\|r_j\| < \epsilon\|r_0\|, k = j\) break.
   3.10. \(v_{j+1} = f/\beta_j, V_{j+1} = [V_j \, v_{j+1}]\)
4. end
5. Compute \(y_k = R_k^{-1}z_k\) and \(\hat{x}_k = x_0 + M^{-1}V_k y_k\).

Figure 2.1: Right-preconditioned GMRES Algorithm

more Givens rotation, \(G(k + 1, k + 2, \theta_{k+1})\), is necessary to zero out \(\beta_{j+1}\) and obtain the QR factorization of \(\bar{H}_{k+1}\)

\[
Q^H_{k+2} \bar{H}_{k+1} = G(k + 1, k + 2, \theta_{k+1})^H \hat{Q}^H_{k+2} \bar{H}_{k+1} = \begin{bmatrix} R_{k+1} & \\
0 & \end{bmatrix}.
\]

Givens rotations can be used to progressively compute the QR factorization of the upper Hessenberg matrix as it is being constructed through the Arnoldi process. However, in order to solve the least squares problem (2.12) they must also be progressively applied to the vector \(\beta_0 e_1\). Using the vector \(\bar{z}_k\) defined by (2.15) and the
structure of the Givens rotation (2.19), the entries of $\bar{z}_k$ satisfy the recurrence

$$\begin{align*}
\zeta_k &= c_k \bar{\zeta}_k \\
\bar{\zeta}_{k+1} &= s_k \bar{\zeta}_k,
\end{align*}$$

(2.21)

where $c_k = \cos(\theta_k)$ and $s_k = \sin(\theta_k)$ are defined by the Givens rotation $G(k, k + 1, \theta_k)$ and $\bar{\zeta}_k$ is the last entry of $\bar{z}_{k-1}$. The current residual norm (2.18) is available each iteration through the recurrence relationship (2.21). Figure 2.1 illustrates an implementation of GMRES that incorporates the solution of the least squares problem with the construction of the Krylov subspace.

The GMRES method is considered a long-term recurrence method because all the previous Krylov subspace vectors $V_k$ are needed to compute the next subspace vector $v_{k+1}$. This requires storing the entire sequence of Krylov subspace vectors, which can be prohibitive if $k$ is large. Furthermore, computation of the next subspace vector $v_{k+1}$ requires more operations as $k$ increases. These less than desirable characteristics have motivated the development of restarting and acceleration techniques for the GMRES method. While these techniques will not be formally presented in this thesis, we may refer to restarted GMRES in future discussions. A basic restarted GMRES algorithm, GMRES($m$), uses the GMRES algorithm in Figure 2.1 iteratively until convergence is reached.

### 2.1.2 CGS Method

Another avenue of research has been in short-term recurrence methods for non-Hermitian problems like the Bi-Conjugate Gradient (BiCG) [23] and Quasi-Minimal Residual (QMR) [25] methods. These are oblique projection methods where the constraint space $\mathcal{L}$ is $\mathcal{K}_k(A^H, w)$. More precisely, both the BiCG and QMR methods compute approximate solutions $\hat{x}$ to $Ax = b$ of the form

$$\hat{x} \in x_0 + \mathcal{K}_k(A, r_0) \quad \text{s.t.} \quad b - A\hat{x} \perp \mathcal{K}_k(A^H, \bar{r}_0),$$
where \((\tilde{r}_0, r_0) \neq 0\).

The Lanczos biorthogonalization algorithm, or two-sided Lanczos algorithm, in Figure 2.2 is used to build a biorthogonal pair of basis for the subspaces

\[
K_k(A, v_1) = \text{span}\{v_1, Av_1, A^2v_1, \cdots, A^{k-1}v_1\}
\]

and

\[
K_k(A^H, w_1) = \text{span}\{w_1, A^Hw_1, (A^H)^2w_1, \cdots, (A^H)^{k-1}w_1\},
\]

where \((v_1, w_1) \neq 0\). Assuming a breakdown has not occurred in step 3.5, this algorithm constructs the matrix factorization

\[
AV_m = V_m T_m + \delta_{m+1} v_{m+1} e_m^T
\]

\[
A^H W_m = W_m T_m^H + \beta_{m+1} w_{m+1} e_m^T
\]

\[
W_m^H AV_m = T_m
\]

where

\[
T_m = \begin{bmatrix}
\alpha_1 & \beta_2 \\
\delta_2 & \alpha_2 & \beta_3 \\
& \ddots & \ddots \\
& \delta_{m-1} & \alpha_{m-1} & \beta_m \\
& & \delta_m & \alpha_m
\end{bmatrix}
\]

It is important to note here that practical implementations of the Lanczos algorithm must detect the breakdown in step 3.5. Furthermore, in contrast to Arnoldi’s method, this situation does not automatically mean that an invariant subspace has been computed. Look-ahead techniques are commonly utilized in practical implementations of two-sided Lanczos, introducing complex logic to characterize and recover from a breakdown in step 3.5.

In general, short-term recurrence methods are attractive due to their computational efficiency and low memory requirements. Methods based on two-sided Lanczos, like BiCG and QMR, require storage for only six vectors regardless of the size of \(k\),
\textbf{Input:} \((A, v_1, w_1, m)\)  
\textbf{Output:} \((V_m, W_m, T_m)\) where \(V_m^H W_m = I_m\)

1. Normalize \(v_1, w_1\) such that \((v_1, w_1) = 1\)
2. \(\beta_1 = \delta_1 = 0, \ w_0 = v_0 = 0\)
3. \textbf{for} \(j = 1, 2, \ldots, m\)
   3.1. \(\alpha_j = (Av_j, w_j)\)
   3.2. \(\hat{v}_{j+1} = Av_j - \alpha_j v_j - \beta_j v_{j-1}\)
   3.3. \(\hat{w}_{j+1} = A^H w_j - \alpha_j w_j - \delta_j w_{j-1}\)
   3.4. \(\delta_{j+1} = |(\hat{v}_{j+1}, \hat{w}_{j+1})|^{1/2}\)
   3.5. If \(\delta_{j+1} = 0\), return
   3.6. \(\beta_{j+1} = (\hat{v}_{j+1}, \hat{w}_{j+1})/\delta_{j+1}\)
   3.7. \(w_{j+1} = \hat{w}_{j+1}/\beta_{j+1}\)
   3.8. \(v_{j+1} = \hat{v}_{j+1}/\delta_{j+1}\)
4. \textbf{end}

\textbf{Figure 2.2:} Lanczos Biorthogonalization Algorithm

and perform fewer operations per step than the Arnoldi method. Furthermore, the number of operations per step is constant and not increasing with the size of the search or constraint subspace. One downside of the Lanczos biorthogonalization algorithm is the application of \(A^H\), which is not trivial to provide in many situations. This has motivated the development of the transpose-free algorithms like CGS [84], Bi-CGSTAB [91] and TFQMR [26].

The \textit{Conjugate Gradient Squared} (CGS) algorithm introduced by Sonneveld [84] was the transpose-free algorithm that inspired the development of BiCGSTAB and TFQMR. CGS is derived from BiCG, cleverly avoiding the application of \(A^H\) by exploiting the residual polynomial \(\psi_k\) and conjugate-direction polynomial \(\pi_k\) created by the BiCG algorithm (Figure 2.3). Given the initial conditions for BiCG, steps 1 and 2, we can see that steps 3.3 and 3.4 allow us to write the residual polynomial at
Input: \((A, x_0, m)\)
Output: \(\bar{x}_k\)

1. \(r_0 = b - Ax_0\), choose \(\tilde{r}_0\) s.t. \((\tilde{r}_0, r_0) \neq 0\)
2. \(p_0 = r_0, \tilde{p}_0 = \tilde{r}_0\)
3. for \(j = 0, 1, \cdots, m\)
   3.1. \(\alpha_j = (r_j, \tilde{r}_j)/(Ap_j, \tilde{p}_j)\)
   3.2. \(\hat{x}_{j+1} \leftarrow \hat{x}_j + \alpha_j p_j\)
   3.3. \(r_{j+1} \leftarrow r_j + \alpha_j Ap_j\)
   3.4. \(\tilde{r}_{j+1} \leftarrow \tilde{r}_j + \alpha_j A^H \tilde{p}_j\)
   3.5. If converged, break.
   3.6. \(\beta_{j+1} = (r_{j+1}, \tilde{r}_{j+1})/(r_j, \tilde{r}_j)\)
   3.7. \(p_{j+1} = r_{j+1} + \beta_{j+1} p_j\)
   3.8. \(\tilde{p}_{j+1} = \tilde{r}_{j+1} + \beta_{j+1} \tilde{p}_j\)
4. end

Figure 2.3: Biconjugate Gradient (BiCG) Algorithm

step \(j\) as

\[
\begin{align*}
  r_j & = \psi_j(A)r_0 & (2.22) \\
  \tilde{r}_j & = \psi_j(A^H)\tilde{r}_0
\end{align*}
\]

where \(\psi_j\) is a polynomial of degree \(j\) satisfying \(\psi_j(0) = 1\). Likewise, steps 3.7 and 3.8 allow us to write the conjugate-direction polynomial at step \(j\) as

\[
\begin{align*}
  p_j & = \pi_j(A)r_0 & (2.23) \\
  \tilde{p}_j & = \pi_j(A^H)\tilde{r}_0,
\end{align*}
\]

where \(\pi_j\) is a polynomial of degree \(j\). Relationships (2.22) and (2.23) can be used to
produce alternate formulas for the computation of $\alpha_j$

\[
\alpha_j = \frac{(\psi_j(A)r_0, \psi_j(A^H)\tilde{r}_0)}{(A\pi_j(A)r_0, \pi_j(A^H)\tilde{r}_0)} = \frac{(\psi_j^2(A)r_0, \tilde{r}_0)}{(A\pi_j^2(A)r_0, \tilde{r}_0)}
\]  
(2.24)

and $\beta_{j+1}$

\[
\beta_{j+1} = \frac{(r_{j+1}, \tilde{r}_{j+1})}{(r_j, \tilde{r}_j)} = \frac{(\psi_{j+1}(A)r_0, \psi_{j+1}(A^H)\tilde{r}_0)}{(\psi_j(A)r_0, \psi_j(A^H)\tilde{r}_0)} = \frac{(\psi_{j+1}^2(A)r_0, \tilde{r}_0)}{(\psi_j^2(A)r_0, \tilde{r}_0)}.
\]  
(2.25)

The formulas (2.24) and (2.25) provide a definition for the BiCG iteration parameters that just requires the vectors $\psi_k^2(A)r_0$ and $\pi_k^2(A)r_0$.

Developing a method that uses (2.24) and (2.25) requires a recursion relationship for building the polynomials $\psi_k^2$ and $\pi_k^2$. First, notice that (2.22) and (2.23) can be directly substituted into steps 3.3 and 3.7 of BiCG to define recurrence relationships for the residual and conjugate-direction polynomials

\[
\psi_{k+1}(A) = \psi_k(A) - \alpha_k A\pi_k(A)
\]  
(2.26)

and

\[
\pi_{k+1}(A) = \psi_{k+1}(A) + \beta_{k+1}\pi_k(A),
\]  
(2.27)

respectively. Then, squaring (2.26) and (2.27) gives

\[
\psi_{k+1}^2(A) = \psi_k^2(A) - 2\alpha_k A\psi_k(A)\pi_k(A) + \alpha_k^2 A^2\pi_k^2(A)
\]  
(2.28)

and

\[
\pi_{k+1}^2(A) = \psi_{k+1}^2(A) + 2\beta_{k+1}\psi_{k+1}(A)\pi_k(A) + \beta_{k+1}^2\pi_k^2(A).
\]  
(2.29)

The cross terms $\psi_k(A)\pi_k(A)$ and $\psi_{k+1}(A)\pi_k(A)$ can be dealt with by introducing the recurrence vectors

\[
u_k = \psi_k(A)\pi_k(A)r_0, \\
q_{k+1} = \psi_{k+1}(A)\pi_k(A)r_0.
\]
Multiplying $\psi_k(A)$ by the recurrence for $\pi_k(A)$ (2.27) results in
\[ \psi_k(A)\pi_k(A) = \psi_k^2(A) + \beta_k\psi_k(A)\pi_{k-1}(A). \] (2.30)

Then the recurrence relationship for $\psi_{k+1}(A)\pi_k(A)$ is derived from multiplying (2.26) by $\pi_k(A)$
\[ \psi_{k+1}(A)\pi_k(A) = \psi_k(A)\pi_k(A) - \alpha_kA\pi_k^2(A). \] (2.31)

The cross term $\psi_k(A)\pi_k(A)$ in (2.28) can then be simplified using (2.30). Multiplying by $r_0$, we get the final form of the recurrence relationships for the CGS polynomials:
\[ r_{k+1} = r_k - \alpha_kA(2r_k + 2\beta_kq_k - \alpha_kAp_k), \]
\[ p_{k+1} = r_{k+1} + 2\beta_{k+1}q_{k+1} + \beta_{k+1}^2p_k, \]
\[ u_k = r_k + \beta_kq_k \]
and
\[ q_{k+1} = u_k - \alpha_kAp_k. \]

The CGS algorithm in Figure 2.4 uses these recurrence relationships to compute approximate solutions of the form
\[ \hat{x}_k \in x_0 + K_{2k}(A, r_0) \]
whose residuals involve the square of the BiCG residual polynomial
\[ r_k = (\psi_k(A))^2r_0. \] (2.32)

In practice, CGS usually converges twice as fast as BiCG. However, CGS is susceptible to the rounding errors introduced through the application of the BiCG residual polynomial squared. This numerical instability can lead to irregular convergence behavior. The CGS algorithm in Figure 2.4 is also vulnerable to breakdown in steps 3.1 and 3.6, which can be avoided by look-ahead techniques. BiCGSTAB and TFQMR were both derived from CGS specifically to improve its convergence behavior.
Input: \((A, x_0, m)\)
Output: \(\hat{x}_k\)

1. \(r_0 = b - Ax_0\), choose \(\tilde{r}_0\) s.t. \((\tilde{r}_0, r_0) \neq 0\)

2. \(p_0 = r_0, u_0 = r_0\)

3. for \(j = 0, 1, 2, \cdots, m\)
   3.1. \(\alpha_j = (r_j, \tilde{r}_0)/(Ap_j, \tilde{r}_0)\)
   3.2. \(q_{j+1} = u_j - \alpha_j Ap_j\)
   3.3. \(\hat{x}_{j+1} \leftarrow \hat{x}_j + \alpha_j (u_j + q_{j+1})\)
   3.4. \(r_{j+1} \leftarrow r_j - \alpha_j A(u_j + q_{j+1})\)
   3.5. If converged, break.
   3.6. \(\beta_{j+1} = (r_{j+1}, \tilde{r}_0)/(r_j, \tilde{r}_0)\)
   3.7. \(u_{j+1} = r_{j+1} + \beta_{j+1} q_{j+1}\)
   3.8. \(p_{j+1} = u_{j+1} + \beta_{j+1}(q_{j+1} + \beta_{j+1} p_j)\)

4. end

Figure 2.4: Conjugate Gradient Squared (CGS) Algorithm

### 2.1.3 BiCGSTAB Method

Since the irregular convergence behavior of CGS can be attributed to the form of the residual (2.32), it might be better to consider a more general form

\[
r_k = \chi_k(A)\psi_k(A)r_0.
\]  

(2.33)

This was proposed by van der Vorst [91] who noticed that computing the BiCG iteration parameters only requires that

\[
(\chi_j(A)\psi_i(A)r_0, \tilde{r}_0) = 0, \ i \neq j
\]

for a suitable set of polynomials \(\chi_j\) of degree \(j\). Standard BiCG implementations choose \(\chi_j = \psi_j\) and compute \(\tilde{r}_j = \psi_j(A^H)\tilde{r}_0\). The CGS algorithm takes advantage of this choice to derive recursion relationships for \(\psi^2_j(A)\) from \(\psi_j(A)\).
The BiCGSTAB algorithm [91] uses a residual polynomial of the form (2.33) where $\chi_k$ is chosen to keep the residual norm small while retaining the rapid convergence behavior of CGS. In particular, $\chi_k$ is of the form

$$
\chi_k(z) = (1 - \omega_k z)(1 - \omega_{k-1} z) \cdots (1 - \omega_1 z),
$$

(2.34)

where $\omega_k$ is chosen at the $k$th iteration to minimize

$$
\|r_k\| = \|(I - \omega_k A)\chi_{k-1}(A)\psi_k(A)r_0\|.
$$

The conjugate direction vector $p_k$ is of the form

$$
p_k = \chi_k(A)\pi_k(A)r_0.
$$

(2.35)

Recurrence relationships for the BiCGSTAB residual and conjugate direction vector can be derived using the BiCG relations (2.26), (2.27) and (2.34) resulting in

$$
r_k = (I - \omega_k A)\chi_{k-1}(A)[\psi_{k-1}(A) - \alpha_{k-1} A\pi_{k-1}(A)]r_0
$$

$$
= (I - \omega_k A)[r_{k-1} - \alpha_{k-1} A p_{k-1}]
$$

for the residual vector and

$$
p_k = \chi_k(A)[\psi_k(A) + \beta_k \pi_{k-1}(A)]r_0
$$

$$
= r_k + \beta_k (I - \omega_k A)p_{k-1}
$$

for the conjugate direction vector. These relationships can now be used with the biorthogonality properties of the BiCG polynomials to recover the iteration parameters

$$
\alpha_{k-1} = \frac{(r_{k-1}, \tilde{r}_0)}{(A p_{k-1}, \tilde{r}_0)}
$$

and

$$
\beta_k = \frac{(\alpha_{k-1}/\omega_k)[(r_k, \tilde{r}_0)/(r_{k-1}, \tilde{r}_0)]}{\omega_k}
$$

The iteration polynomial for the right-preconditioned BiCGSTAB algorithm can be implicitly constructed through the polynomials $\psi_k$, $\pi_k$, and $\chi_k$. Through the recurrence relationships (2.26), (2.27), and (2.34) it is clear that each of these polynomials
Input: \((A, M, x_0, m)\)
Output: \(\hat{x}_k\)

1. \(r_0 = b - Ax_0\), choose \(\tilde{r}_0\) s.t. \((\tilde{r}_0, r_0) \neq 0\)
2. \(p_0 = r_0\)
3. for \(j = 0, 1, 2, \cdots, m - 1\)
   3.1. \(\hat{\rho} = M^{-1}p_j\)
   3.2. \(v = A\hat{\rho}\)
   3.3. \(\alpha_j = (r_j, \tilde{r}_0)/(v, \tilde{r}_0)\)
   3.4. \(r_{j+1/2} = r_j - \alpha_j v\)
   3.5. If converged, \(\hat{x}_{j+1} = \hat{x}_j + \alpha_j \hat{\rho}\), break
   3.6. \(\hat{r} = M^{-1}r_{j+1/2}\)
   3.7. \(u = A\hat{r}\)
   3.8. \(\omega_{j+1} = (r_{j+1/2}, u)/(u, u)\)
   3.9. \(\hat{x}_{j+1} = \hat{x}_j + \alpha_j \hat{\rho} + \omega_{j+1} \hat{r}\)
   3.10. \(r_{j+1} = r_{j+1/2} - \omega_{j+1} u\)
   3.11. If converged, break
   3.12. \(\beta_{j+1} = (\alpha_j/\omega_{j+1})[(r_{j+1}, \tilde{r}_0)/(r_j, \tilde{r}_0)]\)
   3.13. \(p_{j+1} = r_{j+1} + \beta_{j+1}(p_j - \omega_{j+1} v)\)
4. end

Figure 2.5: Right-preconditioned BiCGSTAB Algorithm

is of degree \(k\) and can be reconstructed through the iteration parameters \(\alpha_{k-1}, \beta_k,\) and \(\omega_k\). Like CGS, BiCGSTAB computes approximate solutions of the form

\[
\hat{x}_k \in x_0 + K_{2k}(A, r_0)
\]

This means that the iteration polynomial at step \(k\), \(\phi_{2k-1}\), is a polynomial of degree \(2k - 1\).

The expression for the BiCGSTAB iteration polynomial can be derived from the computation of the approximate solution \(\hat{x}_k\) in step 3.9 of Figure 2.5. An alternate form for the approximate solution can be obtained by using step 3.4 and the definitions
Performing induction on this form of the approximate solution we arrive at a more useful expression for $\hat{x}_k$

$$\hat{x}_k = x_0 + M^{-1} \sum_{i=0}^{k-1} \chi_i(AM^{-1})[\alpha_i(1 - \omega_i z)\pi_i(z) + \omega_i \psi_i(z)]r_0.$$ 

Thus the BiCGSTAB iteration polynomial is

$$\phi_{2k-1}(z) = \sum_{i=0}^{k-1} \chi_i(z)[\alpha_i(1 - \omega_i z)\pi_i(z) + \omega_i \psi_i(z)].$$

The highest degree term in this polynomial is

$$\alpha_k \omega_k z \chi_{k-1}(z)\pi_{k-1}(z)$$

which proves that the BiCGSTAB iteration polynomial derived above is of degree $2k - 1$. The construction and application of this polynomial will be discussed in Section 2.2.2.

The BiCGSTAB algorithm presented in this section is often viewed as having the combined effect of BiCG and GMRES(1) [92]. This is because the residual at step $k$ is of the form

$$r_k = \psi_k(A)\chi_k(A)r_0,$$

where the $\psi_k(A)$ is the BiCG residual polynomial and $\chi_k(A)$ is a product of $k$ linear factors

$$\chi_k(A) = (I - \omega_1 A)(I - \omega_2 A) \cdots (I - \omega_k A)$$

that are defined by a parameter, $\omega_j$, chosen at iteration $j$ to minimize the residual $r_j$. If $A \in \mathbb{R}^{n \times n}$, then $\chi_k(A)$ has only real roots like GMRES(1), which is not ideal if $A$
has complex eigenvalues. In fact, for discretized advection-dominated PDEs having almost purely imaginary eigenvalues, it has been observed that the convergence of BiCGSTAB can stagnate while BiCG converges to the solution [81].

Generalizations of the BiCGSTAB algorithm have been proposed to avoid the stagnation resulting from using independent linear factors in $\chi_k(A)$. Gutzknecht developed the BiCGStab2 algorithm [31] that constructs second-degree polynomials for $\chi_k(A)$. This algorithm performs a correction on the linear factor from the previous odd-numbered step during the next even-numbered step to create these second-degree polynomials. Sleijpen and Fokkema generalized the relationship between BiCG and GMRES(1) in BiCGSTAB to BiCG and GMRES($\ell$), resulting in the BiCGSTAB($\ell$) algorithm [81].

### 2.1.4 Transpose-Free Quasi Minimum Residual Method

The TFQMR algorithm proposed by Freund [26] introduces a quasi-minimal residual approach to the CGS algorithm to improve upon the irregular convergence behavior. Using the sequence of vectors $u_k$ and $q_{k+1}$ computed by CGS, iterates can be defined with a quasi-minimization property. Accomplishing this takes a slight reorganization of the CGS iteration. First remember that the relations for the vectors $u_k$ and $q_{k+1}$ are

\begin{align*}
  u_k &= \psi_k(A)\pi_k(A)r_0 \\
  q_{k+1} &= \psi_{k+1}(A)\pi_k(A)r_0.
\end{align*}

Then set

\begin{equation}
  y_m = \begin{cases}
    u_{k-1} & \text{if } m = 2k - 1 \text{ is odd,} \\
    q_k & \text{if } m = 2k \text{ is even,}
  \end{cases}
\end{equation}

and

\begin{equation}
  w_m = \begin{cases}
    \psi_{k-1}^2(A)r_0 & \text{if } m = 2k - 1 \text{ is odd,} \\
    \psi_k(A)\psi_{k-1}(A)r_0 & \text{if } m = 2k \text{ is even}
  \end{cases}
\end{equation}
for $m = 1, 2, \cdots$. Using (2.36, 2.37) and the BiCG polynomial residual recurrence (2.26) we get the relation

$$Ay_m = \frac{1}{\alpha_{\lfloor (m-1)/2 \rfloor}}(w_m - w_{m+1}),$$

where it is assumed that $\alpha_{\lfloor (m-1)/2 \rfloor} \neq 0$ for all $m$. Then we can write the relation as

$$AY_m = W_{m+1}\tilde{B}_m,$$  \hspace{1cm} (2.40)

where

$$\tilde{B}_m = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ -1 & 1 & \ddots & \vdots \\ 0 & \ddots & \ddots & 0 \\ \vdots & \ddots & -1 & 1 \\ 0 & \cdots & 0 & -1 \end{bmatrix} \times \text{diag}(\frac{1}{\alpha_0}, \frac{1}{\alpha_0}, \frac{1}{\alpha_1}, \cdots, \frac{1}{\alpha_{\lfloor (m-1)/2 \rfloor}}).$$

It is at this point that TFQMR diverges from CGS to utilize the factorization (2.40) for computing an approximate solution

$$\hat{x}_k \in x_0 + K_{2k}(A, r_0)$$

with residual norm properties. Given the assumption that $\alpha_{\lfloor (m-1)/2 \rfloor} \neq 0$ for all $m$, the vectors $y_m$ span the Krylov subspace $K_m(A, r_0) = K_{2k}(A, r_0)$. This means that the solution $\hat{x}_m$ at step $m$ can be written as

$$\hat{x}_m = x_0 + Y_mz_m$$  \hspace{1cm} (2.41)

for some $z_m \in \mathbb{C}^m$. The corresponding residual vector is

$$r_m = b - A\hat{x}_m$$

$$= r_0 - AY_mz_m$$

$$= W_{m+1}(e_1 - \tilde{B}_mz_m),$$  \hspace{1cm} (2.42)

where $w_1 = \psi_0^2(A)r_0 = r_0$ using (2.39) and the definition of the BiCG residual polynomial.
The columns of $W_{m+1}$ are not orthonormal, unlike GMRES, making it more difficult to compute a $z_m \in \mathbb{C}^m$ that minimizes $\|r_m\|$. Instead, the quasi-minimum residual approach minimizes the second term in (2.42) while scaling the vectors of $W_{m+1}$ to bound its norm. Using the scaling matrix

$$\Omega_{m+1} = diag(\omega_1, \omega_2, \cdots, \omega_{m+1}), \quad \omega_k > 0, \quad k = 1, 2, \cdots, m + 1,$$

(2.42) can be rewritten as

$$r_m = W_{m+1} \Omega_{m+1}^{-1} \Omega_{m+1} (e_1 - \bar{B}_m z_m)$$

$$= W_{m+1} \Omega_{m+1} (\omega_1 e_1 - \bar{B}_{m,\Omega} z_m), \quad (2.43)$$

where $\bar{B}_{m,\Omega} = \Omega_{m+1} \bar{B}_m$ is a lower bidiagonal matrix with full column rank. The quasi-minimum residual norm approach requires that $z_m$ in (2.41) is the solution of the least squares problem

$$\tau_m = \min_{z \in \mathbb{C}^m} \| \omega_1 e_1 - \bar{B}_{m,\Omega} z \|. \quad (2.44)$$

The scaling matrix $\Omega_{m+1}$ is central to the TFQMR bound on the norm of the residual (2.43), affecting both the norm of $W_{m+1} \Omega_{m+1}$ and the least squares problem (2.44). The standard strategy for choosing the weights $\omega_k$ is

$$\omega_k = \| w_k \|, \quad k = 1, 2, \cdots, m + 1 \quad (2.45)$$

This results in the product $W_{m+1} \Omega_{m+1}$ having columns that are unit length making it easy to bound the norm of the first term in the residual (2.43),

$$\| W_{m+1} \Omega_{m+1} \| \leq m + 1. \quad (2.46)$$

We can use (2.44) and (2.46) to provide an bound on the residual at step $m$,

$$\| r_m \| = \| W_{m+1} \Omega_{m+1}^{-1} (\omega_1 e_1 - \bar{B}_{m,\Omega} z_m) \|$$

$$\leq \| W_{m+1} \Omega_{m+1}^{-1} \| \| \omega_1 e_1 - \bar{B}_{m,\Omega} z_m \|$$

$$\leq (m + 1) \tau_m. \quad (2.47)$$
Computing the bound on the residual (2.47) at each step \( m \) requires the solution of the least squares problem (2.44). Avoiding this expensive computation is made possible through the integration of the solution of the least-squares problem with the construction of the Krylov subspace \( \mathcal{K}_m(A, r_0) \). The approach used in GMRES to perform this integration (see Section 2.1.1) is also utilized in TFQMR. The rest of the subsection will discuss a practical implementation of TFQMR, which includes an overview of the approach used to solve the least squares problem. The end result will be the algorithm used in our numerical experiments.

As with GMRES, the standard approach for solving the least squares problem (2.44) involves a QR factorization of \( \bar{B}_{m,\Omega} \)

\[
Q_{m+1}^H \bar{B}_{m,\Omega} = \begin{bmatrix} R_m \\ 0 \end{bmatrix},
\]

(2.48)

where \( Q_{m+1} \in \mathbb{R}^{m+1 \times m+1} \) is an orthogonal matrix and \( R_m \in \mathbb{R}^{m \times m} \) is an upper-triangular matrix. Applying \( Q_{m+1}^H \) to \( \omega_1 e_1 \) yields

\[
Q_{m+1}^H(\omega_1 e_1) = \bar{g}_m = \begin{bmatrix} g_m \\ \bar{\gamma}_{m+1} \end{bmatrix},
\]

(2.49)

where \( g_m = [\gamma_1, \cdots, \gamma_m]^T \in \mathbb{R}^m \), \( \bar{\gamma}_{m+1} \in \mathbb{R} \), and the \( z_m \) that solves (2.44) is the solution of the linear system

\[
R_m z_m = g_m.
\]

(2.50)

The matrix \( Q_{m+1} \) is the product of \( m \) Givens rotations (2.20), so the entries of \( \bar{g}_m \) satisfy the recurrence

\[
\gamma_m = c_m \bar{\gamma}_m
\]

(2.51)

\[
\bar{\gamma}_{m+1} = s_m \bar{\gamma}_m,
\]

where \( c_m = \cos(\theta_m) \) and \( s_m = \sin(\theta_m) \) are defined by the Givens rotation \( G(m, m + 1, \theta_m) \) and \( \bar{z}_m \) is the last entry of \( \bar{g}_{m-1} \). This simplifies the least squares problem
(2.44) to

$$\tau_m = |\gamma_{m+1}|,$$

(2.52)

making it inexpensive to compute the residual norm bound at each iteration

$$\|r_m\| \leq (m + 1) |\gamma_{m+1}|.$$

While computing the bound for the residual norm can be seen as a byproduct of the QR factorization of $\bar{B}_{m,\Omega}$, updating the approximate solution $\hat{x}_m$ at each iteration is more involved. TFQMR is a short-term recurrence method, so it is not necessary or efficient to store all of the vectors $y_m$ that span the Krylov space and the upper triangular matrix $R_m$ to compute the approximate solution (2.41). Instead, practical implementations of TFQMR utilize the auxiliary solution vectors

$$\tilde{x}_m = x_0 + Y_m \tilde{z}_m,$$

where $\tilde{z}_m = B_{m,\Omega}^{-1} \omega_1 e_1$

(2.53)

to provide a simple mechanism for updating the approximate solution $\hat{x}_m$ at each iteration. The matrix $B_{m,\Omega} \in \mathbb{R}^{m \times m}$ is the matrix obtained by deleting the last row of $\bar{B}_{m,\Omega}$,

$$B_{m,\Omega} = \begin{bmatrix} \omega_1 \alpha_0^{-1} & 0 & \cdots & 0 \\ -\omega_2 \alpha_0^{-1} & \omega_2 \alpha_0^{-1} & \ddots & \vdots \\ 0 & -\omega_3 \alpha_0^{-1} & \ddots & 0 \\ \vdots & \ddots & \ddots & \omega_m \alpha_{\lfloor (m-1)/2 \rfloor}^{-1} \\ 0 & \cdots & 0 & -\omega_{m+1} \alpha_{\lfloor (m-1)/2 \rfloor}^{-1} \end{bmatrix},$$

and is nonsingular through the definition of $\alpha_k$ and $\omega_k$. Thus, the auxiliary iterates (2.53) are well defined and the vector $\tilde{z}_m$ is

$$\tilde{z}_m = [\alpha_0 \alpha_0 \alpha_1 \alpha_1 \cdots \alpha_{\lfloor (m-1)/2 \rfloor}]^T.$$
The function being minimized in (2.44) with respect to \( \tilde{z}_m \) is

\[
\| \omega_1 e_1 - \bar{B}_{m,\Omega} \tilde{z}_m \| = \left\| \omega_1 e_1 - \begin{bmatrix} B_{m,\Omega} \\ -\omega_{m+1} \alpha_{\lfloor (m-1)/2 \rfloor} e_m^T \end{bmatrix} \tilde{z}_m \right\|
\]

\[
= \omega_{m+1} \alpha_{\lfloor (m-1)/2 \rfloor} e_m^T \tilde{z}_m
\]

\[
= \omega_{m+1}. \tag{2.54}
\]

A relationship between the auxiliary iterates \( \tilde{x}_m \) and the approximate solution \( \hat{x}_m \) can be derived from a relationship between \( z_m \) and \( \tilde{z}_m \). The later relationship can be established through the QR factorization of \( \bar{B}_{m,\Omega} \) (2.48)

\[
Q_{m+1}^H \bar{B}_{m,\Omega} = G(m, m+1, \theta_m)^H \begin{bmatrix} Q_m^H & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} B_{m,\Omega} \\ -\omega_{m+1} \alpha_{\lfloor (m-1)/2 \rfloor} e_m^T \end{bmatrix}
\]

\[
= G(m, m+1, \theta_m)^H \begin{bmatrix} Q_m^H B_{m,\Omega} \\ -\omega_{m+1} \alpha_{\lfloor (m-1)/2 \rfloor} e_m^T \end{bmatrix}.
\]

The orthogonality of Givens rotations gives us a formula for the QR factorization of \( B_{m,\Omega} \) in terms of the QR factorization of \( \bar{B}_{m,\Omega} \)

\[
Q_m^H B_{m,\Omega} = \text{diag}(1, \cdots, 1, c_m) R_m. \tag{2.55}
\]

Using (2.50), (2.51), and (2.55), \( \tilde{z}_m \) can be rewritten as

\[
\tilde{z}_m = (Q_m^H B_{m,\Omega})^{-1} \begin{bmatrix} g_{m-1} \\ \gamma_m \end{bmatrix}
\]

\[
= R_m^{-1} \begin{bmatrix} R_{m-1} z_{m-1} \\ \gamma_m / c_m^2 \end{bmatrix}. \tag{2.56}
\]

This can be combined with (2.50) to obtain a relationship between \( z_m \) and \( \tilde{z}_m \)

\[
z_m = (1 - c_m^2) \begin{bmatrix} \tilde{z}_{m-1} \\ 0 \end{bmatrix} + c_m^2 \tilde{z}_m. \tag{2.57}
\]

Multiplying both sides of (2.57) by \( Y_m \) and adding \( x_0 \) results in a relationship between the auxiliary iterates \( \tilde{x}_m \) and the approximate solution \( \hat{x}_m \)

\[
\hat{x}_m = (1 - c_m^2) \tilde{x}_{m-1} + c_m^2 \tilde{z}_m. \tag{2.58}
\]
All the attributes for the auxiliary iterates can now be used to assemble a practical implementation of the TFQMR algorithm. The function being minimized in (2.44) can be written using $\tilde{z}_m$, (2.49), (2.51), and (2.56), as

$$
\| \omega_1 e_1 - \vec{B}_{m,\Omega} \tilde{z}_m \| = \| Q^{-H}_{m+1} \omega_1 e_1 - Q^{-H}_{m+1} \vec{B}_{m,\Omega} \tilde{z}_m \| = \left\| \begin{bmatrix} g_{m-1} \\ \gamma_m \\ \tilde{\gamma}_{m+1} \end{bmatrix} - \begin{bmatrix} g_{m-1} \\ \gamma_m / c_m^2 \\ 0 \end{bmatrix} \right\| = \left\| \begin{bmatrix} \gamma_m |s_m|^2 / c_m^2 \\ \tilde{\gamma}_{m+1} \end{bmatrix} \right\| = |\tilde{\gamma}_m| |s_m| / c_m.
$$

(2.59)

Let $\vartheta_m = |s_m| / c_m$ be the tangent of the angle $\theta_m$, then from (2.54) and (2.59) we get

$$
\omega_{m+1} = |\tilde{\gamma}_m| \vartheta_m.
$$

Using this equivalence and (2.52), simple formulas for the tangent and cosine of $\theta_m$ as well as the current value of the least squares problem (2.44) can be derived

$$
\vartheta_m = \omega_{m+1} / \tau_{m-1}, \quad c_m = \frac{1}{\sqrt{1 + \vartheta_m^2}}, \quad \text{and} \quad \tau_m = \tau_{m-1} \vartheta_m c_m.
$$

Consider the vector

$$
d_m = \frac{1}{\alpha_{(m-1)/2}} (\hat{x}_m - \hat{x}_{m-1}).
$$

This can be used to create a simpler formula for the update of the approximate solution (2.58)

$$
\hat{x}_m = \hat{x}_{m-1} + \eta_m d_m, \quad \text{where} \quad \eta_m = c_m^2 \alpha_{(m-1)/2}.
$$

(2.60)

The final step is to obtain a simple recurrence relationship for $d_m$ using the definition of the auxiliary iterate (2.53)

$$
d_m = y_m + \frac{\vartheta_m^2}{\alpha_{(m-1)/2}} d_{m-1}.
$$

(2.61)

This completes the derivation of a practical implementation of TFQMR, which is illustrated in Figure 2.6.
Input: \((A, M, x_0, m, \epsilon)\)
Output: \(\hat{x}_k\)

1. \(r_0 = b - Ax_0\), choose \(\tilde{r}_0\) s.t. \((\tilde{r}_0, r_0) \neq 0\)
2. \(w_0 = r_0, y_0 = r_0, v_0 = AM^{-1}y_0, \hat{y} = v_0, \) and \(d_0 = 0\)
3. \(\tau_0 = \|r_0\|, \rho_0 = (\tilde{r}_0, r_0), \alpha_0 = \rho_0/(\hat{y}, \tilde{r}_0), \psi_0 = 0, \) and \(\eta_0 = 0\)
4. for \(j = 0, 1, 2, \ldots, m - 1\)
   4.1. If \(j\) is even, then
   4.2. \(\alpha_{j+1} = \alpha_j = \rho_j/(v_j, \tilde{r}_0)\)
   4.3. \(y_{j+1} = y_j - \alpha_j v_j\)
   4.4. End
   4.5. \(w_{j+1} = w_j - \alpha_j \hat{y}\)
   4.6. \(d_{j+1} = y_j + (\psi_j^2/\alpha_j)\eta_j d_j\)
   4.7. \(\psi_{j+1} = \|w_{j+1}\|/\tau_j\)
   4.8. \(c_{j+1} = 1/\sqrt{1 + \psi_j^2}\)
   4.9. \(\tau_{j+1} = \tau_j \psi_{j+1} c_{j+1}; \eta_{j+1} = c_{j+1}^2 \alpha_j\)
   4.10. \(\hat{x}_{j+1} = \hat{x}_j + \eta_{j+1} M^{-1} d_{j+1}\)
   4.11. If \((j + 2)\tau_{j+1} < \epsilon, \) break
   4.12. If \(j\) is odd, then
   4.13. \(\rho_{j+1} = (r_{j+1}, \tilde{r}_0); \beta_{j+1} = \rho_{j+1}/\rho_j\)
   4.14. \(y_{j+1} = w_{j+1} + \beta_{j+1} y_j\)
   4.15. \(v_{j+1} = \beta_{j-1}(\hat{y} + \beta_{j-1} v_{j-1})\)
   4.16. \(\hat{y} = AM^{-1}y_{j+1}; v_{j+1} \leftarrow v_{j+1} + \hat{y}\)
   4.17. Else
   4.18. \(\hat{y} = AM^{-1}y_{j+1}\)
   4.19. End
5. end

Figure 2.6: Right-preconditioned TFQMR Algorithm
The TFQMR iteration polynomial $\phi_{m-1}$ can be derived from the computation of the approximate solution $\hat{x}_k$ (2.58), shown in step 4.6 of Figure 2.6. The first step uses (2.36), (2.37), and (2.38) to acquire an alternate form for $y_m$ with respect to a right-preconditioned operator

$$y_m = \psi_{[m/2]}(AM^{-1})\pi_{(m-1)/2}(AM^{-1})r_0.$$  \hspace{1cm} (2.62)

The second step is to perform induction on the recurrence relationship for $d_m$ (2.61) to get

$$d_m = y_m + \sum_{j=1}^{m-1} \left( \prod_{k=j}^{m-1} \frac{\vartheta_k^2 \eta_k}{\alpha_{[k/2]}} \right) y_j$$ \hspace{1cm} (2.63)

Using (2.62) and (2.63) with (2.60) gives an alternate formula for the approximate solution $\hat{x}_m$

$$\hat{x}_m = \hat{x}_{m-1} + \eta_m \psi_{[m/2]}(AM^{-1})\pi_{(m-1)/2}(AM^{-1})r_0 + \sum_{j=1}^{m-1} \left( \prod_{k=j}^{m-1} \frac{\vartheta_k^2 \eta_k}{\alpha_{[k/2]}} \right) \psi_{[j/2]}(AM^{-1})\pi_{(j-1)/2}(AM^{-1})r_0.$$  \hspace{1cm} (2.64)

Induction can be performed on this alternate formula, resulting in a more useful expression for $\hat{x}_m$

$$\hat{x}_m = x_0 + \sum_{i=1}^{m} \eta_i \psi_{[i/2]}(AM^{-1})\pi_{(i-1)/2}(AM^{-1})$$

$$+ \eta_i \sum_{j=1}^{i-1} \left( \prod_{k=j}^{i-1} \frac{\vartheta_k^2 \eta_k}{\alpha_{[k/2]}} \right) \psi_{[j/2]}(AM^{-1})\pi_{(j-1)/2}(AM^{-1})r_0.$$  \hspace{1cm} (2.64)

It is clear from (2.64), that the iteration polynomial for TFQMR is

$$\phi_{m-1}(z) = \sum_{i=1}^{m} \eta_i \left[ \psi_{[i/2]}(z)\pi_{(i-1)/2}(z) + \sum_{j=1}^{i-1} \left( \prod_{k=j}^{i-1} \frac{\vartheta_k^2 \eta_k}{\alpha_{[k/2]}} \right) \psi_{[j/2]}(z)\pi_{(j-1)/2}(z) \right].$$

The term in this polynomial of highest degree is

$$\eta_m \psi_{[m/2]}(z)\pi_{(m-1)/2}(z),$$

which proves that the TFQMR iteration polynomial derived above is of degree $m - 1$. The construction and application of this polynomial will be discussed in Section 2.2.3.
2.2 Fixed-Polynomial Operators

In this thesis, we define a fixed-polynomial operator as a polynomial $\phi_k \in \Pi_k$ that approximates the solution to

$$(A - \sigma B)x = b,$$  \hspace{1cm} (2.65)

through the operation

$$\hat{x} = \phi_k(A - \sigma B)b.$$  

This polynomial is constructed once prior to any eigenvalue computation and only applied once wherever a solution to (2.65) is required. If iterative solvers are used to construct a polynomial $\phi_k$ of low degree, then preconditioning is essential. Enlisting a right-preconditioner for $A - \sigma B$ results in the construction of a fixed polynomial operator that approximates the solution to (2.65) through the operation

$$\hat{x} = M^{-1}_\sigma \phi_k(A_\sigma M^{-1}_\sigma),$$  \hspace{1cm} (2.66)

where $A_\sigma = A - \sigma B$ and $M_\sigma$ is the right-preconditioner for $A_\sigma$. In this section we will discuss how to construct fixed-polynomial operators from the iteration polynomials of BiCGSTAB, TFQMR, and GMRES and how to apply them to compute the approximate solution (2.66).

First, it should be noted that the use of pre-constructed polynomials to compute the approximate solution to (2.65) is not novel. Hybrid iterative methods are a class of linear solvers that use this same approach in, generally, a two-phase implementation. Phase I acquires information about the spectrum of the matrix, which is used in Phase II to design a polynomial iteration. The constructed polynomial is then applied one or more times until the required residual reduction is achieved. Most hybrid algorithms rely directly on Ritz values obtained from the Arnoldi process in Phase I. However, there are limitations to this approach since these values may be misleading for non-normal matrices. Alternative hybrid methods have been proposed by
\textbf{Input:} $A$, $B$, $M$, $\sigma$, $\text{nev}$, $\text{tol}$, $\text{maxdeg}$

\textbf{Output:} $(\lambda_j, \hat{x}_j) j = 1, \cdots, \text{nev}$ eigenpairs of $(A, B)$ closest to $\sigma$.

Let $A_\sigma = A - \sigma B$, $M_\sigma = M - \sigma B$.

1. Construct the operator
   
   1.1. Let $v_1$ be a random vector, where $\|v_1\| = 1$.
   
   1.2. Use linear solver to solve $A_\sigma x = Bv_1$ with $M_\sigma$ as a right preconditioner. Run iteration until relative residual falls below specified tolerance $\text{tol}$, or $\deg(\phi(A_\sigma M_\sigma^{-1})) > \text{maxdeg}$.
   
   1.3. Output iteration parameters (eg. $H_k$ and $y$ for GMRES).

2. Compute eigenvectors of the operator
   
   Use the fixed-polynomial operator $M_\sigma^{-1}\phi(A_\sigma M_\sigma^{-1})$ with IRA to compute the eigenvectors corresponding to the $\text{nev}$ eigenvalues of $(A, B)$ closest to $\sigma$.

3. Recover eigenvalues of $A$
   
   For each purified approximate eigenvector $\hat{x}_j$, use the Rayleigh quotient $\frac{\hat{x}_j^H A \hat{x}_j}{\hat{x}_j^H B \hat{x}_j}$ to compute the approximate eigenvalue $\lambda_j$.

Figure 2.7: Algorithm for implementing IRA with fixed-polynomial operator

Nachtigal, Reichel, and Trefethen [57] and Manteuffel and Starke [49] that circumvent computing eigenvalue estimates in Phase 1.

This thesis takes a unique approach in directly applying iteration polynomials from some common iterative methods as fixed-polynomial operators. Furthermore, it utilizes these operators in a preconditioning scheme to accelerate the convergence of the IRA method to the eigenvalues of interest. The simplicity of implementing this preconditioning scheme with any Krylov subspace method is appealing (Figure 2.7). The fixed-polynomial operator is constructed through one linear solve with a random right-hand side. It is then applied during every Arnoldi step until convergence. So, the cost and storage for this operator is fixed once it is constructed. In practice, the dimension of the polynomial can be restricted to keep the cost of the operator low.

In the following sections we will discuss how to construct fixed-polynomial operators for each of the GMRES, BiCGSTAB, and TFQMR iteration polynomials.
Instead of computing the roots of the iteration polynomials derived in the previous sections and designing a polynomial iteration, we will implicitly apply the polynomials using the iteration parameters from each algorithm. These iteration parameters are obtained from a single linear solve for (2.2) using a random vector for $b$. In practice, using a vector $b$ that is equally weighted in all eigendirections seems to facilitate the construction of an effective fixed-polynomial operator.

### 2.2.1 GMRES Fixed-Polynomial Operator

The GMRES iteration polynomial was derived in Section 2.1.1 to be

$$
\phi_{k-1}(z) = \sum_{j=0}^{k-1} \eta_{j+1} \hat{\phi}_j(z),
$$

(2.67)

where $y_k = [\eta_1, \eta_2, \cdots, \eta_k]^T \in \mathbb{R}^k$ is the solution to the least squares problem. The polynomial $\hat{\phi}_j(z)$ is defined recursively through the Gram-Schmidt process to be

$$
v_{j+1} = \frac{1}{\gamma_{j+1,j}} [AM^{-1}v_j - \sum_{i=1}^{j} v_i \gamma_{i,j}] = \hat{\phi}_j(AM^{-1})b,
$$

for $j = 1, 2, \cdots, k - 1$, where $\gamma_{i,j} = H_k(i, j)$. Looking at the GMRES iteration polynomial in this way reveals the fact that it can be easily reconstructed from just $H_k$ and $y_k$. The algorithm for applying the GMRES iteration polynomial using this approach is presented in Figure 2.8.

There are some important observations about the construction and application of the GMRES iteration polynomial (2.67). First, the acquisition of the iteration parameters $H_k$ during the construction process requires the storage of both the Gram-Schmidt coefficients in $H_k$ and the upper triangular matrix $R_k$ from the QR factorization of $H_k$. This is because of the efficiency in obtaining the current residual norm through the integration of the QR factorization with the Arnoldi process. Second, the set of vectors $V_k$ generated by the GMRES iterative method (Figure 2.1.1) are orthogonal, while the vectors $V_k$ generated by this process (Figure 2.8) are not. Furthermore, all of them must be stored during the application of the GMRES iteration.
Input: \((A_\sigma, M_\sigma, H_k, y, b)\)
Output: \(\hat{x} = M_\sigma^{-1}\phi_{k-1}(A_\sigma M_\sigma^{-1})b\)

1. \(\hat{v}_1 = b, V_1 = [\hat{v}_1]\)
2. for \(j = 1, 2 \cdots, k - 1\)
   2.1. \(w = A_\sigma M_\sigma^{-1}\hat{v}_j, h = H_k(1 : j, j)\)
   2.2. \(f = w - V_j h\)
   2.3. \(\hat{v}_{j+1} = f/\gamma_{j+1, j}, V_{j+1} = [V_j \hat{v}_{j+1}]\)
3. end
4. \(\hat{x} = M_\sigma^{-1}V_k y\)

Figure 2.8: Applying the GMRES fixed-polynomial operator

polynomial. This is a result of the derivation of (2.67), which uses the recursive formula for the vectors \(V_k\) from the long-term recurrence used by GMRES. An iteration polynomial constructed from a short-term recurrence method, like BiCGSTAB or TFQMR, would only require a few vectors to be stored. Thus, short-term recurrence methods may provide a more efficient fixed-polynomial operator.

Nachtigal, Reichel, and Trefethen [57] proposed an alternative method for obtaining the GMRES iteration polynomial. This alternative formulation directly computes the coefficients, \(\alpha_i\) for \(k = 0, \cdots, k - 1\), that define the iteration polynomial

\[
\phi_{k-1}(z) = \alpha_0 + \alpha_1 z + \cdots + \alpha_{k-1} z^{k-1}.
\] (2.68)

These coefficients are determined from the relationship between the set of Krylov basis vectors \(V_k\) and the Krylov vectors \(K_k = \{r_0, AM^{-1}r_0, \cdots, (AM^{-1})^{k-1}r_0\}\), which span the same space for any \(k\). This relationship can be written in matrix notation as

\[
V_k = K_k C_k,
\] (2.69)

where \(C_k\) is an upper triangular matrix that can be constructed during the GMRES iteration. Each column \(j\) of the upper triangular matrix \(C_k\) comes from the iterative
formula for constructing the next Krylov basis vector at step $j$,

$$v_{j+1} = \frac{1}{\gamma_{j+1,j}} [AM^{-1}v_j - V_j h],$$

giving

$$\begin{bmatrix} c_{1,j+1} \\ \vdots \\ c_{j+1,j+1} \end{bmatrix} = \gamma_{j+1,j}^{-1} \begin{bmatrix} 0 \\ c_{1,j} \\ \vdots \\ c_{j,j} \end{bmatrix} - \gamma_{j+1,j}^{-1} \begin{bmatrix} C_jh \end{bmatrix}.$$ 

The relationship (2.69) and the solution of the least squares problem $y_k$ can be used to compute the vector

$$C_k y_k = [\alpha_0, \cdots, \alpha_{k-1}]^T,$$

which contains the coefficients of the iteration polynomial (2.68). This polynomial can be applied using Horner iteration or the corresponding residual polynomial can be factored and then the iteration polynomial can be applied using Richardson iteration.

The two approaches for generating and applying the GMRES iteration polynomial presented by Nachtigal, Reichel, and Trefethen [57] provide quite a savings in storage and computation over the initial formula for the GMRES iteration polynomial (2.67). However, the manner in which these polynomials are applied is not prescribed by their construction and can affect their stability. For instance, an appropriate ordering of the roots is necessary for the stability of the Richardson iteration, while intermediate steps of the Horner iteration can have residuals large enough to introduce rounding errors in the final result. The initial formula presented in this section (2.67) prescribes the manner in which the polynomial is applied and thus expects to circumvent the instability issues of the latter formulas. This is why the approach for applying the GMRES iteration polynomial illustrated by Figure 2.8 was chosen for the numerical experiments in this thesis.
2.2.2 BiCGSTAB Fixed-Polynomial Operator

The BiCGSTAB iteration polynomial was derived in Section 2.1.3 to be

\[ \phi_{2k-1}(z) = \sum_{i=0}^{k-1} \chi_i(z)[\alpha_i(1 - \omega_{i+1}z)\pi_i(z) + \omega_{i+1}\psi_i(z)], \]

where \( \psi_k(z) \) and \( \pi_k(z) \) are the CGS residual and conjugate direction polynomials, respectively, and are defined recursively by

\[
\begin{align*}
\psi_{k+1}(z) &= \psi_k(z) - \alpha_k z \pi_k(z) \\
\pi_{k+1}(z) &= \psi_{k+1}(z) + \beta_{k+1} \pi_k(z).
\end{align*}
\]

A third polynomial \( \chi_k(z) \), enlisted for the residual norm minimization at each step \( k \), is defined recursively by

\[ \chi_k(z) = (1 - \omega_k)\chi_{k-1}(z). \]

This formula for the BiCGSTAB iteration polynomial reveals the fact that it can be easily reconstructed from the coefficients \( \alpha_j, \beta_j, \) and \( \omega_j \) for \( j = 1, \ldots, k \). The algorithm for applying the BiCGSTAB iteration polynomial using this approach is presented in Figure 2.9.

Since BiCGSTAB is a short-term recurrence method, interesting observations can be made about the construction and application of its iteration polynomial. First, the acquisition of the iteration parameters \( \alpha_j, \beta_j, \) and \( \omega_j \) only require their storage, which is three vectors of length \( k \). Second, additional storage is required for the preconditioned conjugate direction and residual vectors during the application of the iteration polynomial. The implementation in Figure 2.9 uses only one additional vector to store both preconditioned vectors, requiring a division of the solution update shown in step 3.9 of the original BiCGSTAB algorithm (Figure 2.5). This means that, regardless of the size of the polynomial, the BiCGSTAB fixed-polynomial operator will only need five vectors for workspace in its application.
Input: \((A_\sigma, M_\sigma, T_k, b)\) where \(T_k = [\alpha_0 \beta_1 \omega_1; \cdots; \alpha_{k-1} \beta_k \omega_k] \in \mathbb{R}^{k \times 3}\)

Output: \(\hat{x}\)

1. \(r = b, p = b\)
2. for \(j = 0, 1, 2, \cdots, k\)
   2.1. \(\hat{i} = M_\sigma^{-1}p\)
   2.2. \(\tilde{p} = A_\sigma \hat{t}\)
   2.3. \(x \leftarrow x + \alpha_j \hat{t}\)
   2.4. \(r \leftarrow r - \alpha_j \tilde{p}\)
   2.5. \(\hat{i} = M_\sigma^{-1}r\)
   2.6. \(\tilde{r} = A_\sigma \hat{t}\)
   2.7. \(x \leftarrow x + \omega_{j+1} \hat{t}\)
   2.8. \(r \leftarrow r - \omega_{j+1} \tilde{r}\)
   2.9. \(p = r + \beta_{j+1}(p - \omega_{j+1} \tilde{p})\)
3. end

Figure 2.9: Applying the BiCGSTAB fixed-polynomial operator

### 2.2.3 TFQMR Fixed-Polynomial Operator

The TFQMR iteration polynomial was derived in Section 2.1.4 to be

\[
\phi_{m-1}(z) = \sum_{i=1}^{m} \eta_k \left[ \psi_{[i/2]}(z) \pi_{[i-1]/2}(z) + \sum_{j=1}^{i-1} \left( \prod_{k=j}^{i-1} \frac{\vartheta_k^2 \eta_k}{\alpha_{[k/2]}} \right) \psi_{[j/2]}(z) \pi_{[j-1]/2}(z) \right],
\]

where \(\vartheta_k = |s_k|/c_k\) is the tangent of the Givens rotation angle \(\theta_k\) and \(\eta_k = c_k^2 \alpha_{[k-1]/2}\).

The polynomials \(\psi_{\ell}(z)\) and \(\pi_{\ell}(z)\) are the CGS residual and conjugate direction polynomials, respectively, and are defined recursively by

\[
\psi_{\ell}(z) = \psi_{\ell-1}(z) - \alpha_{\ell-1} z \psi_{\ell-1}(z)
\]
\[
\pi_{\ell}(z) = \psi_{\ell}(z) + \beta_{\ell} \pi_{\ell}(z).
\]

The formula for the TFQMR iteration polynomial is by far the most complex of the three polynomials presented in this thesis. This derivation of the polynomial allows
for more than one approach to its application. We will discuss three possible schemes for constructing and applying the TFQMR iteration polynomial, ending with the approach that will be used in our numerical experiments.

The first scheme exploits the quasi-minimal residual norm approach employed by TFQMR to compute approximate solutions with a form similar to GMRES,

\[ \hat{x} = x_0 + Y_m z_m. \]

Thus, a GMRES-style reconstruction could be adopted for applying this iteration polynomial. It would only require \( \alpha_j \) and \( \beta_j \) for \( j = 1, \cdots, \lfloor m/2 \rfloor \) to recursively construct the matrix \( Y_m \). The vector of linear weights, \( z_m \), is the solution to the least squares problem (2.44). While the GMRES-style reconstruction is easy to understand, \( z_m \) is never explicitly constructed and is not a natural byproduct of the TFQMR algorithm making this an unattractive approach.

Another scheme for reconstructing the TFQMR polynomial involves the auxiliary solution vectors (2.53) that were utilized during the development of a practical TFQMR implementation. The relationship between the auxiliary solution \( \tilde{x}_m \) and approximate solution \( \hat{x}_m \) at step \( m \) is

\[ \hat{x}_m = (1 - c_m^2) \tilde{x}_{m-1} + c_m^2 \tilde{x}_m. \]

Using this relationship to apply the TFQMR iteration polynomial would only require \( \alpha_j \) and \( \beta_j \) for \( j = 1, \cdots, \lfloor m/2 \rfloor \) to recursively construct the vectors \( Y_m \) and the cosine \( c_j \) of each Givens rotation angle. The implementation in Figure 2.10 illustrates this approach. Additional storage is required for applying the preconditioner to \( y_j \). However, storing this vector avoids the need for a second application of the preconditioner to update of the auxiliary vector in step 2.2. In the end, regardless of the degree of the polynomial, only six workspace vectors are needed to apply the TFQMR iteration polynomial using this implementation.

The third scheme results directly from the TFQMR iteration polynomial, employing the iteration parameters that are computed through the TFQMR algorithm.
Input: \((A_\sigma, M_\sigma, T_{\lfloor m/2 \rfloor}, c_m, b)\) where
\[ T_{\lfloor m/2 \rfloor} = [\alpha_0 \beta_1; \cdots ; \alpha_{\lfloor (m-1)/2 \rfloor} \beta_{\lfloor m/2 \rfloor}] \in \mathbb{R}^{\lfloor m/2 \rfloor \times 2} \]

Output: \(\hat{x}\)

1. \(\hat{x} = 0, x_{aux} = 0, y = b, \hat{t} = M_\sigma^{-1}y, \hat{y} = A_\sigma \hat{t}, v = \hat{y}, \) and \(w = b\)

2. for \(j = 0, 1, 2, \cdots, m\)
   2.1. \(w \leftarrow w - \alpha_{\lfloor j/2 \rfloor} \hat{y}\)
   2.2. \(x_{aux} \leftarrow x_{aux} + \alpha_{\lfloor j/2 \rfloor} \hat{t}\)
   2.3. \(x = (1 - c_j^2)x + c_j^2x_{aux}\)
   2.4. If \(j\) is even, then
   2.5. \(y \leftarrow y - \alpha_{\lfloor j/2 \rfloor} v\)
   2.6. \(\hat{t} = M_\sigma^{-1}y, \hat{y} = A_\sigma \hat{t}\)
   2.7. Else
   2.8. \(y = w + \beta_{j-1}y\)
   2.9. \(v = \beta_{j-1}(\hat{y} + \beta_{\lfloor (j-1)/2 \rfloor} v)\)
   2.10. \(\hat{t} = M_\sigma^{-1}y, \hat{y} = A_\sigma \hat{t}\)
   2.11. \(v \leftarrow v + \hat{y}\)
   2.12. End

3. end

Figure 2.10: Applying the TFQMR iteration polynomial (with auxiliary solution vectors) (Figure 2.6). This approach requires \(\alpha_j\) and \(\beta_j\) to recursively construct the matrix \(Y_m\). Constructing the vector \(d_{j+1}\) and updating the approximate solution \(\hat{x}_{j+1}\) in steps 4.6 and 4.10, respectively, would also require \(\eta_j\) and \(\vartheta_j\) for \(j = 1, \cdots, m\). The third scheme, illustrated in Figure 2.11, is similar to that of GMRES or BiCGSTAB because it is just the application of the original iterative method with pre-computed iteration parameters. For that reason, this is the approach used in our numerical experiments.

A few observations can be made about the selected approach for applying the
Input: \((A_\sigma, M_\sigma, T_{[m/2]}, c_m, b)\) where 
\[T_{[m/2]} = [\alpha_0 \beta_1; \cdots; \alpha_{(m-1)/2} \beta_{[m/2]}] \in \mathbb{R}^{[m/2] \times 2}\]

Output: \(\hat{x}\)

1. \(w = b, y = b, \hat{t} = M_\sigma^{-1}b, v = A_\sigma \hat{t}, \hat{y} = v,\) and \(\hat{d} = 0\)
2. for \(j = 0, 1, 2, \cdots, m - 1\)
   2.1. \(w \leftarrow w - \alpha_{[j/2]} \hat{y}\)
   2.2. \(\hat{d} = \hat{t} + (\vartheta^2 / \alpha_{[j/2]}) \eta_j \hat{d}\)
   2.3. \(\hat{x} \leftarrow \hat{x} + \eta_{j+1} \hat{d}\)
   2.4. If \(j\) is even, then
   2.5. \(y \leftarrow y - \alpha_{[j/2]} v\)
   2.6. \(\hat{t} = M_\sigma^{-1} y, \hat{y} = A_\sigma \hat{t}\)
   2.7. Else
   2.8. \(y = w + \beta_{[j-1)/2]} y\)
   2.9. \(v = \beta_{[j-1)/2]} (\hat{y} + \beta_{[j-1)/2]} v)\)
   2.10. \(\hat{t} = M_\sigma^{-1} y, \hat{y} = A_\sigma \hat{t}\)
   2.11. \(v \leftarrow v + \hat{y}\)
   2.12. End
3. end

Figure 2.11: Applying the TFQMR iteration polynomial

TFQMR iteration polynomial (Figure 2.11). The acquisition of the iteration parameters \(\alpha_j, \beta_j, \eta_j,\) and \(\vartheta^2\) require the storage of two vectors of length \([m/2]\) and two vectors of length \(m\). Additional storage is required for applying the preconditioner to the current \(y_j\), but this circumvents the need for an additional application of the preconditioner during the update of \(\hat{d}\) in step 2.2. Again, this means that regardless of degree of the polynomial, the TFQMR fixed polynomial operator as shown in Figure 2.11 will only need six workspace vectors in its application. In Section 2.5, we will compare all three fixed-polynomials operators in terms of computational and mem-
ory costs. In the next section we will address the reconstruction of the approximate eigenpairs from any fixed-polynomial operator.

### 2.3 Reconstructing Eigenpairs

Reconstructing approximate eigenpairs of $A$ from a fixed-polynomial operator is straightforward. Let $\sigma$ be the shift in a shift-invert transformation of the standard eigenvalue problem, where $A_\sigma = A - \sigma I$ and $M_\sigma = M - \sigma I$. If IRA is applied to the fixed-polynomial operator $M_\sigma^{-1}\phi(A_\sigma M_\sigma^{-1})$, the resulting $k$-step Arnoldi factorization is

$$M_\sigma^{-1}\phi(A_\sigma M_\sigma^{-1})V_k = V_k H_k + f e_k^T.$$

To find the eigenvalues closest to the shift, select the eigenvalues of largest magnitude of $H_k y = y\theta$ and let $x = V_k y$. Then for each eigenpair $(\theta, y)$ of $H_k$,

$$M_\sigma^{-1}\phi(A_\sigma M_\sigma^{-1})x = x\theta + f e_k^T y = \hat{x}\theta,$$

where $\hat{x} = x + f e_k^T y / \theta$ is the purified eigenvector. Now an approximate eigenpair $(\lambda, q)$ of $A$ can be obtained from

$$A_\sigma \hat{x} - \hat{x} = A \hat{x} - \hat{x}(\sigma + 1 / \theta),$$

where $q = \hat{x} / \|\hat{x}\|$ and $\lambda = \sigma + 1 / \theta$.

The reconstruction is similar for the generalized eigenvalue problem. Let $\sigma$ be the shift, $A_\sigma = A - \sigma B$, and $M_\sigma = M - \sigma B$. The fixed-polynomial operator for this problem is

$$(A - \sigma B)^{-1}B \approx M_\sigma^{-1}\phi(A_\sigma M_\sigma^{-1})B.$$

Applying IRA to this operator results in the factorization

$$M_\sigma^{-1}\phi(A_\sigma M_\sigma^{-1})B V_k = V_k H_k + f e_k^T.$$
Using the same algebraic manipulation as before, for each eigenpair \((\theta, y)\) of \(H_k\),

\[
M_{\sigma}^{-1} \phi(A_{\sigma} M_{\sigma}^{-1}) B x = x \theta + f e_k^T y = \hat{x} \theta,
\]

where \(\hat{x}\) is the purified eigenvector. Then the approximate eigenpair \((\lambda, q)\) can be obtained from

\[
A_{\sigma} \hat{x} - B \frac{1}{\theta} = A \hat{x} - B (\sigma + \frac{1}{\theta}) \theta,
\]

where \(q = \hat{x}/\|\hat{x}\|\) and \(\lambda = \sigma + \frac{1}{\theta}\).

There are some theoretical and practical issues that should be mentioned. Clearly, the formula for directly calculating the approximate eigenvalues \((2.71)\) and \((2.73)\) cannot be obtained from equations \((2.70)\) and \((2.72)\), respectively. This is because the computational formulas discussed here ignore important error terms which will be examined in the next section. Furthermore, in practice it has been found that computing the eigenvalue with a Rayleigh quotient using the purified eigenvector,

\[
\lambda = q^H A q \quad \text{or} \quad \lambda = \frac{q^H A q}{q^H B q},
\]

is better than calculating the eigenvalue directly using \((2.71)\) or \((2.73)\). This can be attributed to specific properties of the Rayleigh quotient as discussed in [95].

2.4 Accuracy and Convergence Heuristics

While computing an approximate eigenpair of \(A\) from a fixed-polynomial operator is straightforward, it is unclear how good the approximation is. Furthermore, it is also unclear whether the eigenpairs of a fixed-polynomial operator will converge to the eigenpairs of \(A\), even if they are computed exactly. In this section we will discuss some accuracy and convergence heuristics that indicate a large dependence upon the residual polynomial of the iterative method used to construct the fixed-polynomial operator. Residual bounds can be derived from equations \((2.70)\) and \((2.72)\) using terms that are ignored in \((2.71)\) and \((2.73)\), respectively. First, we need to provide
some definitions and a lemma that will be used to derive bounds for both the standard and generalized eigenproblems.

**Definition 1.** The spectrum of a matrix $A \in \mathbb{C}^{n \times n}$ is the set of numbers

$$\sigma(A) = \{ \lambda \in \mathbb{C} : \text{rank}(A - \lambda I) < n \}.$$  

Each $\lambda \in \sigma(A)$ is an eigenvalue of $A$ and a root of the characteristic polynomial of $A$, $p_A(\lambda) = \det(A - \lambda I)$.

**Definition 2.** The algebraic multiplicity of any $\lambda \in \sigma(A)$ is the multiplicity of $\lambda$ as a root of $p_A(\lambda)$ and is denoted by $n_a(\lambda)$. The geometric multiplicity of any $\lambda \in \sigma(A)$ is the dimension of $\mathcal{N}(A - \lambda I)$ and is denoted by $n_g(\lambda)$.

**Definition 3.** If for any $\lambda \in \sigma(A)$, the algebraic multiplicity exceeds the geometric multiplicity, $n_a(\lambda) > n_g(\lambda)$, then $\lambda$ is a defective eigenvalue. Any matrix that has a defective eigenvalue is referred to as a defective matrix.

Nondefective matrices are also referred to as diagonalizable matrices ([30], p.316). This means that for any nondefective matrix $A \in \mathbb{C}^{n \times n}$ there exists a nonsingular $X \in \mathbb{C}^{n \times n}$ such that

$$A = XZX^{-1}, \quad (2.74)$$

where $Z = \text{diag}(\lambda_1, \cdots, \lambda_n)$ and $\lambda_i \in \sigma(A)$. For a diagonalizable matrix $A$ we can bound the norm of any residual polynomial $\psi$ evaluated at $A$ as stated in the following lemma:

**Lemma 1.** If $A \in \mathbb{C}^{n \times n}$ is diagonalizable, then there exists a nonsingular $X \in \mathbb{C}^{n \times n}$ such that

$$\|\psi(A)\| \leq \kappa(X) \max_{z \in \sigma(A)} |\psi(z)|.$$ 

**Proof.** Suppose that $A \in \mathbb{C}^{n \times n}$ is diagonalizable, then there exists a nonsingular
\( X \in \mathbb{C}^{n \times n} \) that satisfies (2.74). Then,

\[
\| \psi(A) \| = \| \psi(XZX^{-1}) \|
\]

\[
= \| X \psi(Z)X^{-1} \|
\]

\[
\leq \| X \| \| \psi(Z) \| \| X^{-1} \|
\]

\[
= \kappa(X) \max_{z \in \sigma(A)} | \psi(z) |.
\]

This lemma gives us a bound for the norm of any polynomial evaluated using a nondefective matrix \( A \), facilitating the heuristic studies performed in the next section. Furthermore, it is also a useful result for the residual bounds presented in the next two theorems.

**Theorem 1.** Assume that \( A_\sigma M_\sigma^{-1} \in \mathbb{C}^{n \times n} \) is diagonalizable, where \( A_\sigma = A - \sigma I \) and \( M_\sigma = M - \sigma I \). Consider the Arnoldi method applied to a fixed-polynomial operator \( M_\sigma^{-1} \phi(A_\sigma M_\sigma^{-1}) \), where \( \phi(z) \) is the iteration polynomial constructed by an iterative method and \( \psi(z) = 1 - z \phi(z) \) is its corresponding residual polynomial. Then the approximate eigenpairs \((\lambda, q)\) of \( A \) reconstructed using the computed eigenpairs \((\theta, \hat{x} = x + f \frac{e^T y}{\theta})\) of the fixed-polynomial operator such that

\[
\lambda = \sigma + \frac{1}{\theta} \quad \text{and} \quad q = \frac{\hat{x}}{\| \hat{x} \|}
\]

obey the residual bound

\[
\| Aq - q\lambda \| \leq \frac{\kappa(X)}{| \theta |} \max_{z \in \sigma(A_\sigma M_\sigma^{-1})} | \psi(z) | + C_1 \| f \| e_k^T y \frac{1}{| \theta |^2 \| \hat{x} \|},
\]

where \( C_1 = 1 + \kappa(X) \max_{z \in \sigma(A_\sigma M_\sigma^{-1})} | \psi(z) | \) for some nonsingular \( X \in \mathbb{C}^{n \times n} \).

**Proof.** Suppose the Arnoldi method is applied to the fixed-polynomial operator \( M_\sigma^{-1} \phi(A_\sigma M_\sigma^{-1}) \). From (2.70) each computed eigenpair \((\theta, x)\) of the fixed-polynomial operator satisfies

\[
M_\sigma^{-1} \phi(A_\sigma M_\sigma^{-1}) x = \hat{x} \theta,
\]
where \( \hat{x} = x + f \frac{e_k^T y}{\theta} \) is a purified eigenvector. By subtracting \( A_{\sigma}^{-1} \hat{x} \) from each side, this equation can be rewritten as

\[
\hat{x} \theta - A_{\sigma}^{-1} \hat{x} = M_{\sigma}^{-1} \phi(A_{\sigma} M_{\sigma}^{-1}) x - A_{\sigma}^{-1} \hat{x}.
\]

Multiplying both sides of the equation by \( A_{\sigma} \) and dividing by \( \theta \) results in

\[
A_{\sigma} \hat{x} - \frac{1}{\theta} \hat{x} = \frac{1}{\theta} [A_{\sigma} M_{\sigma}^{-1} \phi(A_{\sigma} M_{\sigma}^{-1}) x - \hat{x}]
= \frac{1}{\theta} [A_{\sigma} M_{\sigma}^{-1} \phi(A_{\sigma} M_{\sigma}^{-1}) x - (x + f \frac{e_k^T y}{\theta})]
= - \frac{1}{\theta} \psi(A_{\sigma} M_{\sigma}^{-1}) x - f \frac{e_k^T y}{\theta^2}.
\]

Using the definition of \( A_{\sigma} \) and dividing the equation by \( \| \hat{x} \| \), the left-hand side of this equation can be rewritten in terms of the approximate eigenpair \((\lambda, q)\) of \( A \)

\[
A q - q \lambda = - \frac{1}{\theta \| \hat{x} \|} \psi(A_{\sigma} M_{\sigma}^{-1}) x - f \frac{e_k^T y}{\theta^2 \| \hat{x} \|}, \tag{2.75}
\]

A residual bound for the computed eigenpair \((\lambda, q)\) of \( A \) is obtained by taking the norm of \( (2.75) \)

\[
\| A q - q \lambda \| = \left\| \frac{1}{\theta \| \hat{x} \|} \psi(A_{\sigma} M_{\sigma}^{-1}) x + f \frac{e_k^T y}{\theta^2 \| \hat{x} \|} \right\|
\leq \frac{\| x \|}{\theta \| \hat{x} \|} \| \psi(A_{\sigma} M_{\sigma}^{-1}) \| + \frac{\| f \| \| e_k^T y \|}{\theta^2 \| \hat{x} \|}
\leq \frac{1}{\theta} \left( 1 + \frac{\| f \| \| e_k^T y \|}{\theta \| \hat{x} \|} \right) \| \psi(A_{\sigma} M_{\sigma}^{-1}) \| + \frac{\| f \| \| e_k^T y \|}{\theta^2 \| \hat{x} \|}.
\]

If \( A_{\sigma} M_{\sigma}^{-1} \in \mathbb{C}^{n \times n} \) is diagonalizable, then there exists a nonsingular \( X \in \mathbb{C}^{n \times n} \) such that

\[
\| A q - q \lambda \| \leq \frac{\kappa(X)}{\theta} \left( 1 + \frac{\| f \| \| e_k^T y \|}{\theta \| \hat{x} \|} \right) \max_{z \in \sigma(A_{\sigma} M_{\sigma}^{-1})} \| \psi(z) \| + \frac{\| f \| \| e_k^T y \|}{\theta^2 \| \hat{x} \|}
= \frac{\kappa(X)}{\theta} \max_{z \in \sigma(A_{\sigma} M_{\sigma}^{-1})} \| \psi(z) \| + C_1 \frac{\| f \| \| e_k^T y \|}{\theta^2 \| \hat{x} \|}, \tag{2.76}
\]

where \( C_1 = 1 + \kappa(X) \max_{z \in \sigma(A_{\sigma} M_{\sigma}^{-1})} \| \psi(z) \| \).

A similar residual bound can be derived for the generalized eigenvalue problem.
**Theorem 2.** Assume that \( A_\sigma M_\sigma^{-1} \in \mathbb{C}^{n \times n} \) is diagonalizable, where \( A_\sigma = A - \sigma B \) and \( M_\sigma = M - \sigma B \). Consider the Arnoldi method applied to a fixed-polynomial operator \( M_\sigma^{-1} \phi(A_\sigma M_\sigma^{-1})B \), where \( \phi(z) \) is the iteration polynomial constructed by an iterative method and \( \psi(z) = 1 - z\phi(z) \) is its corresponding residual polynomial. Then the approximate eigenpairs \((\lambda, q)\) of \((A, B)\) reconstructed using the computed eigenpairs \((\theta, \hat{x} = x + f \frac{e_k^T y}{\theta})\) of the fixed-polynomial operator such that

\[
\lambda = \sigma + \frac{1}{\theta} \quad \text{and} \quad q = \frac{\hat{x}}{\|\hat{x}\|}
\]

obey the residual bound

\[
\|Aq - Bq\lambda\| \leq \frac{\kappa(X) \|Bx\|}{\|\theta\| \|\hat{x}\|} \max_{z \in \sigma(A_\sigma M_\sigma^{-1})} |\psi(z)| + \frac{\|Bf\| \|e_k^T y\|}{\|\theta\|^2 \|\hat{x}\|},
\]

for some nonsingular \( X \in \mathbb{C}^{n \times n} \).

**Proof.** Suppose the Arnoldi method is applied to the fixed-polynomial operator \( M_\sigma^{-1} \phi(A_\sigma M_\sigma^{-1})B \). From (2.72) each computed eigenpair \((\theta, x)\) of the fixed-polynomial operator satisfies

\[
M_\sigma^{-1} \phi(A_\sigma M_\sigma^{-1})Bx = \hat{x}\theta,
\]

where \( \hat{x} = x + f \frac{e_k^T y}{\theta} \) is a purified eigenvector. By subtracting \( A_\sigma^{-1} B\hat{x} \) from each side, this equation can be rewritten as

\[
\hat{x}\theta - A_\sigma^{-1} B\hat{x} = M_\sigma^{-1} \phi(A_\sigma M_\sigma^{-1})Bx - A_\sigma^{-1} B\hat{x}.
\]

Multiplying both sides of the equation by \( A_\sigma \) and dividing by \( \theta \) results in

\[
A_\sigma \hat{x} - B\hat{x} \frac{1}{\theta} = \frac{1}{\theta} \ [A_\sigma M_\sigma^{-1} \phi(A_\sigma M_\sigma^{-1})Bx - B\hat{x}]
= \frac{1}{\theta} \ [A_\sigma M_\sigma^{-1} \phi(A_\sigma M_\sigma^{-1})Bx - B(x + f \frac{e_k^T y}{\theta})]
= -\frac{1}{\theta} \psi(A_\sigma M_\sigma^{-1})Bx - Bf \frac{e_k^T y}{\theta^2}.
\]

Using the definition of \( A_\sigma \) and dividing the equation by \( \|\hat{x}\| \), the left-hand side of this equation can be rewritten in terms of the approximate eigenpair \((\lambda, q)\) of \((A, B)\)

\[
Aq - Bq\lambda = -\frac{1}{\theta \|\hat{x}\|} \psi(A_\sigma M_\sigma^{-1})Bx - B f \frac{e_k^T y}{\theta^2 \|\hat{x}\|}, \quad (2.77)
\]
A residual bound for the computed eigenpair \((\lambda, q)\) of \((A, B)\) is obtained by taking the norm of (2.77)
\[
\|Aq - Bq\lambda\| = \left\| \frac{1}{\theta \|\hat{x}\|} \psi(A_{\sigma}M_{\sigma}^{-1})Bx + Bf \frac{e_k^T y}{\theta^2 \|\hat{x}\|} \right\|
\leq \frac{\|Bx\|}{\|\theta \|\|\hat{x}\|} \|\psi(A_{\sigma}M_{\sigma}^{-1})\| + \frac{\|Bf\| |e_k^T y|}{\|\theta \|^2 \|\hat{x}\|}.
\]
If \(A_{\sigma}M_{\sigma}^{-1} \in \mathbb{C}^{n \times n}\) is diagonalizable, then there exists a nonsingular \(X \in \mathbb{C}^{n \times n}\) such that
\[
\|Aq - Bq\lambda\| \leq \frac{\kappa(X)\|Bx\|}{|\theta \|\|\hat{x}\|} \max_{z \in \sigma(A_{\sigma}M_{\sigma}^{-1})} |\psi(z)| + \frac{\|Bf\| |e_k^T y|}{|\theta \|^2 \|\hat{x}\|}.
\] (2.78)

The first term in the residual bounds (2.76) and (2.78) are of primary concern since the Ritz value \(\theta\) is an approximate eigenvalue of the fixed-polynomial operator and is expected to be large in magnitude. For the standard eigenvalue problem that first term is simple, relating the quality of the reconstructed eigenpairs of \(A\) to how small the residual polynomial is in magnitude over the spectrum of the preconditioned matrix \(A_{\sigma}M_{\sigma}^{-1}\). If \(M_{\sigma} = A_{\sigma}\) is the exact preconditioner for \(A_{\sigma}\), then \(A_{\sigma}M_{\sigma}^{-1} = I\) and the polynomial \(\psi\) would be of single degree with a root at 1. This would result in the elimination of the first term in the residual bound (2.76). Then the residual bound for the reconstructed eigenpairs of \(A\) would be strictly dependent upon the Ritz residual of the computed eigenpairs of the fixed-polynomial operator. This information is known during the computation and can be controlled. The same argument could be made for the residual bound for the generalized eigenvalue problem (2.78). While it is not feasible to have an exact preconditioner, this goes to show that the bound behaves correctly in this extreme case.

In practice, the residual bounds presented in this section provide a heuristic for understanding the relation between the computed eigenvalues of a fixed-polynomial operator and the reconstructed eigenvalues of the original eigenvalue problem. An exact preconditioner cannot be expected, still leaving in question the behavior of the
residual polynomial over the spectrum of the preconditioned operator. Unfortunately, any iterative method used to construct \( \psi \) can only offer a bound for \( \| \psi(A_{\sigma}M_{\sigma}^{-1})v_1 \| \), where \( v_1 \) is the initial vector used to construct the polynomial. Thus, the tolerance used to build the fixed-polynomial operator can only serve as a rough estimate of the maximum magnitude of the residual polynomial is over the spectrum of \( A_{\sigma}M_{\sigma}^{-1} \). In the next section we will study the behavior the three fixed-polynomial operators presented in Section 2.2 with respect to the residual bounds presented here.

### 2.5 Cost Comparison of Fixed-Polynomial Operators

In this section we will compare the GMRES, BiCGSTAB, and TFQMR fixed-polynomial operators from Section 2.2. This comparison will include the memory usage as well as the computational cost required for each application. We will also discuss the differences between the fixed-polynomial operators and the original iterative methods used to derive them in Section 2.1. Finally, a comparison of the spectral characteristics for each fixed-polynomial operator will be made using a model problem from fluid dynamics.

In this discussion, the fixed-polynomial operators are assumed to be of the same degree, hence the cost of the operator \( A_{\sigma} \) and preconditioner \( M_{\sigma} \) will not be included. The degree of the fixed-polynomial operator \( k \) is assumed to be much less than the dimension of the operator, \( k \ll n \). In the comparisons between the original iterative methods, any preallocated space for basis vectors or projections will not be counted as memory usage. Memory usage will be measured by the number of double-precision floating point numbers (“doubles”) necessary for serial computations, while the computational cost will be measured by floating point operations (flops).

The GMRES fixed-polynomial operator (Figure 2.8) is the most expensive fixed-polynomial operator of the three, requiring more memory and floating point oper-
ations per application. Since the GMRES method has a long-term recurrence, a GMRES fixed-polynomial operator of dimension \( k \) requires the storage of all \( k \) vectors of \( V_k \) in its application. Another vector is required for temporary storage in step 2.1. The storage of the Gram-Schmidt coefficients in the upper-Hessenberg matrix \( \tilde{H}_k \) and linear coefficients in the vector \( y_k \) requires an additional \( k(k+5)/2 \) doubles. This brings the total memory required by an application of the GMRES fixed-polynomial operator to \( (k+1)n + k(k+5)/2 \) doubles. The most computationally expensive part of the application of the GMRES fixed-polynomial operator consists of the operations with the block of vectors \( V_k \) in steps 2.2 and 4. The collective total of these calculations is \( nk(k+1)/2 \) flops.

The GMRES iterative method (Figure 2.1) used to construct the fixed-polynomial operator has two additional duties where significant computations are required: orthogonalization of the basis vectors \( V_k \) and the least-squares solution process. The orthogonalization process is the most expensive of the two, requiring at least \( nk(k+1)/2 \) flops for a single Gram-Schmidt step or \( 3nk(k+1)/2 \) flops if the DGKS correction is necessary. The solution of the least-squares problem is a significant computation in GMRES, but still only \( O(k^2) \) flops, so we will omit it from the computational expense estimate of \( 2nk(k+1) \).

The BiCGSTAB fixed-polynomial operator (Figure 2.9) is the least expensive fixed-polynomial operator of the three in terms of memory usage. In fact the memory required by a BiCGSTAB fixed-polynomial operator is constant for any size polynomial; only five vectors of workspace are required in its application. The iteration parameters \( \alpha_j, \beta_j, \) and \( \omega_j \) used to reconstruct the iteration polynomial only take three vectors of \( \lfloor k/2 \rfloor \) length. This brings the total memory required by an application of the BiCGSTAB fixed-polynomial operator to \( 5n + 3\lfloor k/2 \rfloor \) doubles. Computationally, this polynomial operator requires one vector scaling and six vector updates, or “axpys”, of the form

\[
x \leftarrow \alpha x + \beta y.
\]
This collective total of these calculations is $19n$ flops. The BiCGSTAB iterative method (Figure 2.5) used to construct the fixed-polynomial operator uses the same amount of memory, but requires the computation of all the iteration parameters. If done wisely, the computations performed in steps 3.3, 3.8, and 3.12 require an additional six inner products resulting in a total computational expense of $31n$ flops.

The TFQMR fixed-polynomial operator (Figure 2.11) does not require much more memory usage than BiCGSTAB and is the least expensive in terms of computational cost. The memory required by a TFQMR fixed-polynomial operator is also constant for any size polynomial; only six vectors of workspace are required in its application. The iteration parameters $\alpha_j$ and $\beta_j$ used to reconstruct the iteration polynomial only require two vectors of $\lfloor k/2 \rfloor$ length. This brings the total memory required by an application of the TFQMR fixed-polynomial operator to $6n + 2\lfloor k/2 \rfloor$ doubles. Depending on whether the iterate is even or odd, the polynomial operator requires four or six vector updates, respectively. Using an average of five vector updates, $15n$ flops is the computational expense of any application of the TFQMR fixed-polynomial operator. The TFQMR iterative method (Figure 2.6) used to construct the fixed-polynomial operator uses the same amount of memory, but requires the computation of all the iteration parameters. The computations performed in steps 4.2, 4.7, and 4.13 require an additional inner product and norm calculation for each iteration resulting in a total computational expense of $19n$ flops.

The comparison of memory usage and computational cost for the fixed-polynomial operators and their respective iterative methods is summarized in Table 2.1. The difference in computational cost between a fixed-polynomial operator and its respective iterative method is largely due to the precomputation of the iteration parameters. The iteration parameters for all three methods are obtained using inner-products, which requires communication and synchronization for parallel computations creating a computational bottleneck that is avoided by using fixed-polynomial operators.

To explore the spectral characteristics for each fixed-polynomial operator we will
Table 2.1: Summary of fixed-polynomial operator comparisons (per application), fixed-polynomial operator (fpo), original iterative method (solver), where $k$ is the degree of the polynomial and $n$ is the dimension of the operator

<table>
<thead>
<tr>
<th>Iterative Method</th>
<th>Memory Usage (doubles)</th>
<th>Computational Cost (flops)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GMRES (fpo)</td>
<td>$(k + 1)n + k(k + 5)/2$</td>
<td>$nk(k + 1)/2$</td>
</tr>
<tr>
<td>GMRES (solver)</td>
<td>$(k + 1)n + k(k + 5)/2$</td>
<td>$2nk(k + 1)$</td>
</tr>
<tr>
<td>BiCGSTAB (fpo)</td>
<td>$5n + 3\lfloor k/2 \rfloor$</td>
<td>19$n$</td>
</tr>
<tr>
<td>BiCGSTAB (solver)</td>
<td>$5n + 3\lfloor k/2 \rfloor$</td>
<td>31$n$</td>
</tr>
<tr>
<td>TFQMR (fpo)</td>
<td>$6n + 2\lfloor k/2 \rfloor$</td>
<td>15$n$</td>
</tr>
<tr>
<td>TFQMR (solver)</td>
<td>$6n + 2\lfloor k/2 \rfloor$</td>
<td>19$n$</td>
</tr>
</tbody>
</table>

employ a streamline upwinded Petrov-Galerkin (SUPG) finite element discretization of the two-dimensional convection-diffusion equation

$$-\nu \nabla^2 u + \mathbf{w} \cdot \nabla u = f$$

restricted to the unit square $\Omega = [0, 1] \times [0, 1]$. This is a model problem in fluid dynamics that has been analyzed [22] and used to illustrate convergence bounds for GMRES [17, 19, 47]. In these experiments we will look at the vertical wind case, $\mathbf{w} = (0, 1)$, with the diffusion coefficient as $\nu = 0.01$, and Dirichlet boundary conditions. Bilinear finite elements are used to discretize the equation over the unit square using $N$ unknowns in each coordinate direction, yielding a matrix $A \in \mathbb{R}^{n \times n}$, where $n = N^2$.

The SUPG discretization introduces a stabilization, or upwinding, parameter that can be “tuned” to produce approximate solutions of the discrete problem with the best possible error for any degree polynomial approximation. In these experiments we will use the upwinding parameter suggested in [22], which results in $A$ being highly non-normal; the eigenvalues of $A$ are sensitive to perturbations of the matrix $A$. One way to illustrate the sensitivity of the eigenvalues of a matrix $A$ is through
Figure 2.12: Pseudospectrum of 2D convection-diffusion equation discretized with SUPG
its pseudospectra, which can be defined as

\[ \Lambda_\epsilon(A) = \{ z \in \mathbb{C} : z \in \Lambda(A + E) \text{ for some } E \text{ with } \|E\| \leq \epsilon \}. \]

Figure 2.12 shows a pseudospectral plot for \( A \) created by EigTool [96] for \( N = 13 \) and \( \epsilon \in [10^{-17}, 10^{-3}] \).

The spectral characteristics of the fixed-polynomial operators will be explored using this interesting model problem. First we will look at the convergence curves (Figure 2.13) for each of the three iterative methods to see what degree fixed-polynomial operator would be needed to be achieve a reasonable relative residual reduction. These curves illustrate the approximate relative residual norm for each iterative method using the solid blue lines while the solid red symbols indicate the actual relative residual norm, which would be obtained if the current solution was available at each step. The actual residual matches up with the approximate residual norm for all methods except TFQMR. This is because the approximate residual norm for TFQMR is a bound resulting from the quasi-minimum residual approach and can be fairly loose due to
the matrix $W_{m+1}$ (see Section 2.1.4). In practice, this may cause the construction of a TFQMR fixed-polynomial operator that more expensive than necessary.

This convergence curve in Figure 2.13 illustrates the fact that the GMRES method constructs an iteration polynomial of lower degree for any relative residual tolerance compared to TFQMR or BiCGSTAB. This behavior is theoretically supported by the Faber-Manteuffel Theorem [21]. This fundamental result in Krylov subspace theory states that no single, short-term recurrence Krylov subspace method for general non-symmetric linear systems exists that minimizes the error in an inner-product norm independent of the initial guess. The GMRES method is a long-term recurrence method, whose residual polynomial at step $k$ is the solution to the minimization problem

$$
\min_{\psi \in \Pi_k, \psi(0)=1} \|\psi(A)r_0\|.
$$

The Faber-Manteuffel Theorem asserts that there is not an equivalent minimization
property for short-term recurrence methods. Therefore, given any relative residual tolerance
\[ \frac{\|r_k\|}{\|r_0\|} = \frac{\|\psi(A)r_0\|}{\|r_0\|}, \]
the GMRES residual and iteration polynomial will be of equal or lesser degree than those constructed by BiCGSTAB or TFQMR.

As discussed in the previous section, the accuracy bounds for the computed eigenpairs of any fixed-polynomial operator are strongly dependent upon the behavior of its associated residual polynomial over the spectrum of \( A \). In these comparisons we will note the maximum magnitude of the residual polynomials over the spectrum of \( A \). This is important in the first term of the residual bound (2.76) for determining how well a computed eigenpair from the fixed-polynomial operator relates to an approximate eigenpair of \( A \).

The GMRES residual polynomial is well understood, but the TFQMR and BiCGSTAB residual polynomials are not as well understood. To gain an understanding of the three polynomial operators, we shall analyze some contour plots of their respective residual polynomials over the spectrum of \( A \) and \( AM^{-1} \), where \( M \) is an incomplete LU factorization of \( A \). These polynomials will be constructed using constraints on the relative residual tolerance and polynomial degree.

First, consider constructing the fixed-polynomial operators for \( A \) using a relative residual tolerance of \( 10^{-4} \). Figure 2.14 shows some contour plots of all three corresponding residual polynomials over the spectrum of \( A \). The GMRES, BiCGSTAB, and TFQMR residual polynomials are of degree 18, 26, and 26, respectively. From these contours we can see that the GMRES residual polynomial is placing roots at the convex hull of the spectrum, which is expected, resulting in a maximum magnitude of the residual polynomial over \( \sigma(A) \) of \( 3.06 \times 10^{-5} \). The BiCGSTAB residual polynomial has the form
\[ \chi_k(A)\psi_k(A), \]
Figure 2.14: Contour plots of the GMRES (top), BiCGSTAB (middle), and TFQMR (bottom) residual polynomials over $\sigma(A)$ for $N = 13$ using tol=$10^{-4}$. The color indicates the magnitude ($\log_{10}$) of the polynomial in the complex plane. The black lines are lemniscates where $|\psi(z)| = \text{const}$ for certain values of const.
Figure 2.15: Contour plot of the GMRES residual polynomial over $\sigma(A)$ for $N = 13$ and $k = 26$

where $\psi_k(A)$ is the BiCG residual polynomial and $\chi_k(A)$ is analogous to the GMRES(1) residual polynomial (see Section 2.1.3). This explains the large white spot where the residual polynomial is small in magnitude, because it is where many of the roots of $\chi_k(A)$ lie. The other roots surrounding the left side of the spectrum are the mostly roots of the BiCG residual polynomial. The maximum magnitude of the BiCGSTAB residual polynomial over $\sigma(A)$ is $3.54 \times 10^{-4}$. The maximum magnitude of the TFQMR residual polynomial is very small over the spectrum of $A$: $1.10 \times 10^{-9}$. This behavior is supported by the convergence curve for TFQMR in Figure 2.13 which illustrates that there is a big drop in relative residual between the 25th and 26th iteration from $10^{-2}$ to $10^{-6}$, respectively. Thus, TFQMR has placed all 26 roots around the convex hull of the spectrum to create a polynomial that is small over the entire spectrum of $A$.

The contours in Figure 2.14 may seem to indicate that the best residual polynomial is constructed by the TFQMR fixed-polynomial operator. However, the TFQMR and BiCGSTAB residual polynomials are of degree 26, while the GMRES residual
polynomial is only of degree 18. When the residual polynomial constructed by the GMRES fixed-polynomial operator is of dimension 26, it has a very similar contour plot (Figure 2.15) to that of TFQMR and a maximum magnitude of $1.83 \times 10^{-10}$ over $\sigma(A)$. It is also interesting to compare the residual polynomials constructed by all three fixed-polynomial operators when they are restricted to the same degree. The contour plots of these residual polynomials in Figure 2.16 for $k = 18$ clearly indicate that the GMRES fixed-polynomial operator constructs a residual polynomial that has the lowest maximum magnitude over $\sigma(A)$. The BiCGSTAB residual polynomial is fairly reasonable, placing roots to the left and right of the spectrum of $A$. However, the TFQMR method is placing roots far away from the spectrum, resulting in the residual polynomial having the largest magnitude over $\sigma(A)$. 
Figure 2.16: Contour plots of the GMRES (top), BiCGSTAB (middle), and TFQMR (bottom) residual polynomials over $\sigma(A)$ for $N = 13$ using $k = 18$
Now we will consider the effect of right preconditioning on the construction of the three fixed-polynomial operators and their corresponding residual polynomials. The matrix $A$ is the same discretized 2D convection-diffusion equation used in the unpreconditioned experiments. The right preconditioner $M$ is an incomplete LU factorization of $A$ using two different drop tolerances: $10^{-1}$ and $10^{-2}$. As expected, for the same relative residual tolerance, the fixed-polynomial operators constructed for $AM^{-1}$ are of lower degree than for an unpreconditioned $A$ (Figure 2.17).

Preconditioning the discretized convection-diffusion equation results a matrix that is not nearly as non-normal as the original matrix $A$. This can be seen in the pseudospectral plots of $AM^{-1}$ for where $M$ is an incomplete LU factorization of $A$ with a drop tolerance of $10^{-1}$ (Figure 2.18) and $10^{-2}$ (Figure 2.20). For the preconditioned matrix it was only necessary to plot contours for perturbations as small as $\epsilon = 10^{-10}$, while the pseudospectral plots of $A$ include perturbations as small as machine precision. The pseudospectral plots of the preconditioned matrix also illustrate clustering of the eigenvalues, which increases as the drop tolerance for the preconditioner decreases.

Consider the fixed-polynomial operators constructed using an incomplete LU pre-
Figure 2.18: Pseudospectrum of 2D convection-diffusion equation discretized with SUPG right preconditioned with an incomplete LU factorization (drop tolerance = $10^{-1}$)

As indicated by the convergence curve in Figure 2.17, the GMRES, BiCGSTAB, and TFQMR residual polynomials are of degree 5, 8, and 8, respectively. Figure 2.19 shows some contour plots of all three corresponding residual polynomials over the spectrum of $AM^{-1}$. From these contours we can see that the GMRES residual polynomial is placing 4 of its 5 roots on the right side of the spectrum, resulting in a maximum magnitude over $\sigma(AM^{-1})$ of $2.49 \times 10^{-3}$. The BiCGSTAB residual polynomial is placing all of its roots on or to the right of the spectrum, giving a maximum magnitude over $\sigma(AM^{-1})$ that is the highest of all three methods: $2.50 \times 10^{-2}$. The maximum magnitude of the TFQMR residual polynomial is the smallest over the spectrum: $3.25 \times 10^{-4}$. From the contours, it appears that TFQMR is placing its roots on the right side of the spectrum, with several at $(1, 0)$. 
Figure 2.19: Contour plots of the GMRES (top), BiCGSTAB (middle), and TFQMR (bottom) residual polynomials over $\sigma(AM^{-1})$ where $M = luinc(A, 1e^{-1})$, $N = 13$, and $tol=10^{-4}$. 
Now consider the fixed-polynomial operators constructed using an incomplete LU preconditioner with a lower drop tolerance of $10^{-2}$ and the same relative residual tolerance of $10^{-4}$. With this preconditioner, all the eigenvalues of $AM^{-1}$ are clustered about $(1, 0)$ (Figure 2.20). As indicated by the convergence curve in Figure 2.17, the GMRES, BiCGSTAB, and TFQMR residual polynomials are of degree 3, 4, and 4, respectively. Figure 2.21 shows some contour plots of all three corresponding residual polynomials over the spectrum of $AM^{-1}$. From these contours we can see that the GMRES residual polynomial is placing one root and a complex conjugate pair around $(1, 0)$, resulting in a maximum magnitude over $\sigma(AM^{-1})$ of $1.53 \times 10^{-5}$. BiCGSTAB seems to have placed all of its roots at $(1, 0)$ but the magnitude of the polynomial is large at the left and right sides of the spectrum, resulting in a maximum magnitude of $4.02 \times 10^{-4}$. The maximum magnitude of the TFQMR residual polynomial, $1.11 \times 10^{-5}$, is close to that of GMRES. From the contours, the large white spot at $(1, 0)$ indicates that the TFQMR methods is placing most of its roots at there, but has also
Figure 2.21: Contour plots of the GMRES (top), BiCGSTAB (middle), and TFQMR (bottom) residual polynomials over $\sigma(AM^{-1})$ where $M = luinc(A, 1e^{-2})$, $N = 13$, and $tol=10^{-4}$. 
Table 2.2: Summary of the maximum magnitude of the residual polynomials constructed by the three fixed-polynomial operators using a relative residual tolerance of $10^{-4}$ over $\sigma(AM^{-1})$ for various $M$.

| Iterative Method | $\max_{\sigma(AM^{-1})} |\psi(AM^{-1})|$ |
|------------------|----------------------------------|
| $M = I$          | $M = luinc(A, 1e^{-1})$          | $M = luinc(A, 1e^{-2})$ |
| GMRES            | $3.06 \times 10^{-5}$            | $2.49 \times 10^{-3}$          | $1.53 \times 10^{-5}$ |
| BiCGSTAB         | $3.54 \times 10^{-4}$            | $2.5 \times 10^{-2}$           | $4.02 \times 10^{-4}$ |
| TFQMR            | $1.10 \times 10^{-9}$            | $3.91 \times 10^{-4}$          | $1.11 \times 10^{-5}$ |

placed a root to the left of the spectrum.

In this section we compared the cost of building and applying the three fixed-polynomial operators from Section 2.2 as well as examined their spectral characteristics. It was determined that the GMRES fixed-polynomial operator costs the most in terms of memory usage and computations; both increase quadratically with respect to the degree of the polynomial. Meanwhile, the memory usage and computations for both BiCGSTAB and TFQMR increase linearly with respect to the degree of the polynomial. For all three methods using a fixed-polynomial operator instead of the original iterative method significantly reduces the computational cost. Furthermore, the computations that are eliminated through the use of a fixed-polynomial operator are those that require communication and synchronization in the original method.

Considering the accuracy heuristics developed in Section 2.4, we compared the maximum magnitude of the residual polynomials over the spectrum of a discretized 2D convection-diffusion equation. Preconditioning of this discretized operator was also included in this comparison. The results, summarized in Table 2.2, indicate that TFQMR constructs the residual polynomials that has the smallest magnitude over the spectrum. However, the fixed-polynomial operators constructed by TFQMR for a given relative residual tolerance can be quite a bit higher in degree than the GMRES
fixed-polynomial operator. In the next chapter we will continue this comparison and present the numerical performance of these fixed-polynomial operators on a variety of realistic eigenproblems.
Chapter 3

Performance Studies for Several Practical Eigenproblems

Several test cases are presented in this chapter to illustrate the numerical behavior of this preconditioning scheme. Hermitian and non-Hermitian eigenvalue problems are utilized in these experiments. For some of these test cases, comparisons are made between IRA with a fixed polynomial operator and other popular eigenvalue methods, like Jacobi-Davidson [82] (see Section 1.6.1) or Locally Optimal Block Preconditioned Conjugate Gradient (LOBPCG) [38] (see Section 1.6.2). For other test cases, comparisons are made between the different iteration polynomials or the iteration polynomials and the iterative method they are derived from.

The discretized Laplacian operator is a commonly used, pedagogical example for testing numerical algorithms. It will be used in Section 3.2 to provide a compelling argument for the investigation of this preconditioning scheme. This argument will include the comparison of the GMRES fixed-polynomial operator to preconditioned GMRES started anew for each linear solve. Furthermore, the discretized Laplacian operator will also be used to compare the performance of IRA with the GMRES fixed-polynomial operator to the Jacobi-Davidson method and LOBPCG.

In Section 3.3 some examples from the Matrix Market NEP collection [4] are used
to compare the GMRES, BiCGSTAB, and TFQMR fixed-polynomial operators. Here we shall consider the interaction between the polynomial degree and dimension of the Krylov subspace in the accurate computation of the rightmost eigenvalues. This parameter study enables us to conclude that the GMRES fixed-polynomial operator is consistently superior to the other two fixed-polynomial operators. Thus, the GMRES fixed-polynomial operator was selected to demonstrate the performance of this preconditioning scheme in the rest of the numerical experiments.

In Section 3.4, IRA with the GMRES fixed-polynomial operator is compared to the Jacobi-Davidson method for two practical eigenproblems. In Section 3.4.1, a linear stability analysis is performed on a wind-driven barotropic ocean model. In Section 3.4.2, the eigenvalue of largest imaginary part from a discretized sequence of 2D scalar wave equations is computed, which is useful for accelerating the solution of a 2D dynamic rate equation model of a semiconductor laser.

### 3.1 Software and Experiment Parameters

A Matlab implementation of each of these methods is used to obtain the results presented in this chapter. There is one exception for the 2D Laplacian operator, where some results were obtained using FORTRAN and ARPACK [44]. The implementation of JDQR [83] and LOBPCG [37] were treated as “out-of-the-box” type routines. No prior knowledge about these methods is assumed, so these routines are not tweaked to obtain the best performance on each problem. The presented results are meant to be fair comparisons that illustrate the characteristics of using a fixed-polynomial operator with IRA.

In comparing these methods, both performance and accuracy are analyzed. Performance is gauged by matrix-vector products and accuracy is measured by the Ritz residual or relative error for each approximate eigenvalue. When possible, the results are accumulated over an increasing problem size to detect any trends. For some
numerical examples, a comparison of these three methods is made by obtaining the most accurate approximate eigenvalues for a fixed computational cost. LOBPCG and JDQR adhere to a fixed cost by setting the maximum number of iterations accordingly. However, JDQR computes desired eigenvalues one-at-a-time to the desired tolerance. If the tolerance is set too small, not all of the desired eigenvalues will be calculated. Thus, the tolerance for JDQR can vary for each problem.

Unless otherwise mentioned, the parameters for the numerical experiments discussed in this chapter can be found in Tables 3.1, 3.2, and 3.3. The number of eigenvalues of interest depends on the eigenproblem and can vary from one to five. The maximum subspace allowed for either IRA or Jacobi-Davidson is twenty-five and

Table 3.1: Experiment parameters for Jacobi-Davidson (JDQR)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>k</td>
<td>varies</td>
<td>Number of desired eigenvalues</td>
</tr>
<tr>
<td>sigma</td>
<td>0</td>
<td>Shift</td>
</tr>
<tr>
<td>Tol</td>
<td>varies</td>
<td>Convergence tolerance</td>
</tr>
<tr>
<td>jmin</td>
<td>10</td>
<td>Minimum dimension search space</td>
</tr>
<tr>
<td>jmax</td>
<td>25</td>
<td>Maximum dimension search space</td>
</tr>
</tbody>
</table>

Table 3.2: Experiment parameters for LOBPCG

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uini</td>
<td>randn(n,5)</td>
<td>Initial guess, n is dimension of matrix</td>
</tr>
<tr>
<td>mytol</td>
<td>$10^{-12}$</td>
<td>Convergence tolerance</td>
</tr>
<tr>
<td>maxit</td>
<td>varies</td>
<td>Maximum number of iterations</td>
</tr>
</tbody>
</table>


Table 3.3: Experiment parameters for IRA with Fixed-Polynomial Operator

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>k</td>
<td>varies</td>
<td>Number of desired eigenvalues</td>
</tr>
<tr>
<td>m</td>
<td>25</td>
<td>Maximum dimension of Krylov subspace</td>
</tr>
<tr>
<td>sigma</td>
<td>varies</td>
<td>Shift</td>
</tr>
<tr>
<td>maxit</td>
<td>varies</td>
<td>Maximum number of iterations</td>
</tr>
<tr>
<td>gtol</td>
<td>varies</td>
<td>Residual tolerance of GMRES polynomial</td>
</tr>
<tr>
<td>maxdeg</td>
<td>25</td>
<td>Maximum degree of GMRES polynomial</td>
</tr>
</tbody>
</table>

The blocksize used with LOBPCG is five. The setting for any parameter that varies will be defined in each section. It can be assumed that, for any parameter that is not stated in the tables, the default setting provided by the code was used.

3.2 Discretized 2D Laplacian Operator

We will present numerical results from the computation of the smallest five eigenvalues of a discrete approximation to the self-adjoint linear operator $\Delta u$. This operator is restricted to the unit square

$$\Omega := \{(x, y) \in \mathbb{R}^2 : 0 < x < 1, 0 < y < 1\}$$

and subject to Dirichlet boundary conditions. Discretization by finite differences with a grid size $h = \frac{1}{N+1}$ gives a matrix of the form

$$A = \begin{bmatrix}
T_N & -I_N & 0 & \cdots & 0 \\
-I_N & T_N & -I_N & \ddots & \\
0 & \ddots & \ddots & \ddots & 0 \\
\vdots & & -I_N & T_N & -I_N \\
0 & \cdots & 0 & -I_N & T_N
\end{bmatrix}$$
where $I_N$ is the identity matrix in $\mathbb{R}^{N \times N}$ and

$$
T_N = 
\begin{bmatrix}
4 & -1 & 0 & \cdots & 0 \\
-1 & 4 & -1 & \vdots \\
0 & \ddots & \ddots & \ddots & 0 \\
\vdots & & -1 & 4 & -1 \\
0 & \cdots & 0 & -1 & 4
\end{bmatrix}
$$

The value of $N$ is varied throughout the numerical experiments to illustrate any performance trends.

Results are obtained using FORTRAN and ARPACK [44] and compared for IRA in regular mode (largest Ritz values are used as roots of the restart polynomial) and shift-invert mode. For the shift-invert mode, the GMRES fixed-polynomial operator is compared to preconditioned GMRES started anew for each linear solve. A block Jacobi preconditioner (block size $N$) is used for both the fixed-polynomial operator and preconditioned GMRES solver. This preconditioner is applied symmetrically to preserve symmetry and thus allow for the use of MINRES, a residual-minimizing algorithm that employs a three-term recurrence. For the preconditioned GMRES solver, results are compared when it is restarted, GMRES(25), and not restarted, GMRES($\infty$). Both GMRES(25) and GMRES($\infty$) are applied with a relative tolerance of $10^{-6}$. The fixed-polynomial operator is constructed with a relative tolerance of $10^{-2}$. The maximum Krylov subspace allowed for IRA is twenty-five, which is the same as the maximum dimension of the fixed-polynomial operator.

For this example, it is observed that IRA with the GMRES fixed-polynomial operator is the best blend of performance and accuracy. As the problem size increases, it uses far fewer matrix-vector products (Figure 3.1) than IRA in regular mode, GMRES(25), or GMRES($\infty$). Moreover, IRA applied to the GMRES fixed-polynomial operator is just as accurate in computing all five eigenvalues (Figure 3.2) as the regular mode and more accurate than applying preconditioned GMRES, which is only accurate to roughly the accuracy to which the linear systems are solved.
Figure 3.1: Matrix-vector products needed to compute smallest five eigenvalues

The comparison of both preconditioned GMRES solvers serves to illustrate the tradeoff between storage constraints and computational cost. As a Krylov subspace method, it is impossible for GMRES to know in advance how large the subspace must be to obtain a good solution to the linear system. GMRES(∞) assumes that there is enough storage to keep all the basis vectors for the subspace, while GMRES(25) assumes a moderate amount of available storage. While restarting GMRES slows down the convergence, it requires a fixed amount of storage. Not restarting GMRES requires an increasing amount of storage for the Krylov basis, especially as the problem size grows (Figure 3.2). Regardless, neither preconditioned GMRES solver compares to using the GMRES fixed-polynomial operator.

Now we will include some other well-known eigensolvers in the numerical experiments for computing the smallest five eigenvalues of a discrete approximation to the self-adjoint linear operator \( \Delta u \). The eigensolvers are Jacobi-Davidson (JDQR) and LOBPCG. The discretization and preconditioning of the operator are the same as
Figure 3.2: Ritz residual for the smallest five approximate eigenvalues and maximum dimension of the Krylov subspace required by GMRES(∞) discussed at the beginning of this section. However, the dimension of the matrices in this example are slightly smaller, varying from 625 to 19600.

For these experiments there are two cost levels determined by the number of matrix-vector products: 500 (Figure 3.3) and 1000 (Figure 3.4). LOBPCG and JDQR adhere to these cost levels by setting the maximum number of iterations accordingly. However, JDQR computes the five eigenvalues one-at-a-time to the desired tolerance. If the tolerance is set too small, not all five eigenvalues will be calculated. Thus, the tolerance for JDQR is different for each matrix dimension so that all five eigenvalues were calculated to the highest accuracy possible for the fixed number of matrix-vector products. The GMRES fixed-polynomial operator is constructed with a relative tolerance of $10^{-2}$. Other cost levels were analyzed, but these two are representative of the overall observed trends.

The results indicate that if five eigenvalues are required, then IRA with the fixed-
polynomial operator computes the smallest eigenvalue more accurately than JDQR or LOBPCG, especially when the matrix-vector products are held to 500 (Figure 3.3). However, the relative error is higher for eigenvalues further away from the shift. This trend is similarly seen with LOBPCG. As would be expected, JDQR computes all five of the eigenvalues to the same accuracy.

This experiment is designed to compare these three methods with one particular goal: compute the same number of eigenvalues for the same cost. These methods are then compared by analyzing the accuracy of those computed eigenvalues. The design of this experiment illustrates a perspective that may be unfair to JDQR. This is because Jacobi-Davidson is intended to compute one eigenvalue at a time to a prescribed accuracy, not multiple eigenvalues to varying precisions.

Another perspective is to compute a few eigenvalues for a fixed accuracy, where the number of computed eigenvalues is less important and the computational cost
Figure 3.4: Relative error for the smallest five approximate eigenvalues (1000 matrix-vector products)

is still fixed. Figure 3.4 includes information about the accuracy and number of eigenvalues computed when the tolerance is held constant with JDQR. As the problem size increases, JDQR cannot compute all five eigenvalues to the set tolerance. Only the information from the computed eigenvalues is displayed in Figure 3.4. If no data is plotted, then it can be assumed that the eigenvalue was not computed for this matrix dimension. The results indicate that with JDQR the accuracy can be maintained for the smallest eigenvalue if the number of computed eigenvalues is not as important.
3.3 Numerical Comparison of Fixed-Polynomial Operators using the NEP Collection

In this section we will compare the GMRES, BiCGSTAB, and TFQMR fixed-polynomial operators using some realistic eigenvalue problems commonly found in science and engineering from the Matrix Market NEP collection [4]. The three problems are: the reaction-diffusion Brusselator model, a diffusion model for simulating the growth of crystal structures, and the Olmstead model. The computational task for all three model problems is to compute the rightmost eigenvalues and their corresponding eigenvectors.

The numerical results presented in this section were obtained from a Matlab® implementation of the GMRES, BiCGSTAB, and TFQMR fixed-polynomial operators and a simple unrestarted Arnoldi method. These results explore the interaction between the degree of the polynomial and the dimension of the Krylov subspace in achieving an accurate approximation of the rightmost eigenvalue. As was discussed in Section 2.5, restricting the degree of the iteration polynomial does not enable a fair comparison of the performance of the associated fixed-polynomial operator. This parameter study will allow for the fair comparison of these three fixed-polynomial operators by analyzing their accuracy over a range of polynomial degrees and subspace dimensions.

3.3.1 Brusselator Model

Consider the two-dimensional reaction-diffusion model

\[
\begin{align*}
\frac{\partial u}{\partial t} &= \frac{D_u}{L^2} \left( \frac{\partial^2 u}{\partial X^2} + \frac{\partial^2 u}{\partial Y^2} \right) - (B + 1)u + u^2v + C \\
\frac{\partial v}{\partial t} &= \frac{D_v}{L^2} \left( \frac{\partial^2 v}{\partial X^2} + \frac{\partial^2 v}{\partial Y^2} \right) - u^2v + Bu
\end{align*}
\]

(3.1)

for the variables \( u \) and \( v \) over the unit square, \( \Omega = [0, 1] \times [0, 1] \), with homogeneous Dirichlet boundary conditions. The variables \( u \) and \( v \) represent the concentrations
Figure 3.5: Spectrum of reaction-diffusion Brusselator model (rdb2048)

of two reactions. The parameters are given as $B = 5.45$, $C = 2$, $D_u = 0.004$, $D_v = 0.008$, and $L = 0.5$. These equations are then finite differenced over a uniform grid with 32 grid points in each direction and written as $\dot{x} = f(x)$ where $x = [u_{1,1}, v_{1,1}, u_{1,2}, v_{1,2}, ..., u_{n,n}, v_{n,n}]^T$. The stability of the reaction-diffusion model can be ascertained by computing the rightmost eigenvalues of the Jacobian matrix $A = \partial f / \partial x$, which is of dimension 2048.

The computational challenge with this test problem is the fact that the rightmost eigenvalues are a complex conjugate pair and the next eigenvalue has a multiplicity of two (see Table 3.4). Furthermore, the imaginary part of this conjugate pair is one order of magnitude larger than the real part and the real part of the conjugate pair is close to the origin. The entire spectrum of this reaction diffusion model is shown in Figure 3.5 with a magnification of the portion of the spectrum of interest on the right-hand side. In these experiments we chose to use the fixed-polynomial operators to approximate the shift-invert transformation, $(A - \sigma I)^{-1}$, where the shift is placed on the positive real axis ($\sigma = 5$). An incomplete LU preconditioner with a drop tolerance of $10^{-1}$ was used as the right-preconditioner for $A - \sigma I$. This preconditioner has the
same number of non-zeros as the matrix $A_{\sigma} = A - \sigma I$.

First we will look at the convergence curves for each of the three iterative methods to see what degree fixed-polynomial operator would be needed to achieve a reasonable relative residual reduction for this problem. Figure 3.6 illustrates the approximate relative residual norm for each iterative method using the solid blue lines while the solid red symbols indicate the actual relative residual norm, which would be obtained if the current solution was available at each step. As expected, the GMRES method will
construct a fixed-polynomial operator of degree lower than BiCGSTAB or TFQMR for a fixed relative residual tolerance. Furthermore, comparing the approximate relative residual norm and actual relative residual for the TFQMR method indicates that a larger degree fixed-polynomial operator may be constructed than is necessary.

Comparing the performance of the three fixed-polynomial operators will be done through a two-dimensional parameter study of the polynomial degree versus the dimension of the Krylov subspace. Given the convergence curves in Figure 3.6, the range of polynomial degrees was chosen to be from 20 to 56. These polynomials were used to create Krylov subspaces ranging in dimension from 20 to 82 through the Arnoldi process. The Arnoldi process is unrestarted, so the number of matrix and preconditioner applications is equal to the polynomial degree times the number of Arnoldi steps. The rightmost approximate eigenpair is then extracted from the Krylov subspace using the techniques in Section 2.3 and the relative residual with respect to $A$ is computed.

The numerical results from this parameter study are illustrated by the color plots in Figure 3.7. The color indicates the accuracy of the rightmost eigenpair over the parameter space. An expected trend is that a smaller Krylov subspace generated using polynomials of lower degree will yield less accurate eigenpairs. This is indicated by the red hues in the lower left-hand corner of the plots in Figure 3.7. Conversely, the larger Krylov subspaces generated using polynomials of higher degree should yield more accurate eigenpairs, which is indicated by the blue hues in the upper right-hand corner of the plots. Vertical stripes are observed, indicating that a larger Krylov subspace is not resulting in more accurate eigenpairs. This means the degree of the fixed-polynomial operator is the limiting factor. Likewise, horizontal stripes indicate the dimension of the Krylov subspace is the limiting factor.

The numerical results presented in Figure 3.7 indicate that the GMRES fixed-polynomial operator is the best fixed-polynomial operator to use with the reaction-diffusion Brusselator model. This is because the accuracy of the rightmost
Figure 3.7: Parameter study of polynomial degree versus Krylov subspace dimension for computing the rightmost eigenpair of the Brusselator model; GMRES (top), BiCGSTAB (middle), and TFQMR (bottom)
eigenvalues computed using this operator is higher for almost any combination of polynomial degree and Krylov subspace dimension. The performance of the BiCGSTAB fixed-polynomial operator is largely limited by the polynomial degree, especially when the degree of the polynomial is in the twenties. The TFQMR fixed-polynomial operator is less limited by the polynomial degree, but still does not perform as well as GMRES. Furthermore, the larger vertical stripes indicate regions where a higher degree fixed-polynomial operator has no positive effect on the accuracy. These regions are consistent with the plateaus of the TFQMR convergence curve in Figure 3.6.

### 3.3.2 Crystal Structure Model

The phenomena of a needle crystal solidifying in some undercooled liquid can be described by the dual diffusion equations

\[
\frac{\partial U_l}{\partial t} = \alpha \nabla^2 U_l, \quad \frac{\partial U_s}{\partial t} = \alpha \nabla^2 U_s, 
\]

where \(U_l\) is the temperature of the liquid, \(U_s\) is the temperature of the solid, and \(\alpha\) is the thermal diffusivity \([98]\). The motion of the solid-liquid interface, \(\vec{r}\), is related to the temperature fields by

\[
\frac{d\vec{r}}{dt} \cdot \vec{n} = \alpha (\nabla U_s \cdot \vec{n} - \nabla U_l \cdot \vec{n}),
\]

where \(\vec{n}\) is the unit outward pointing normal to the interface. By writing the equations of motion in a moving frame and transforming the problem to a parabolic coordinate system, a basic stationary solution to the free boundary problem can be obtained. The sensitivity of the stationary solution corresponding to a simple parabolic-shaped moving front to small perturbations is of research interest.

Applying linear stability theory to the stationary solution results in the eigenvalue
problem

\[
\frac{1}{\eta^2 + \xi^2} \left[ \frac{\partial^2 U}{\partial \xi^2} + \frac{\partial^2 U}{\partial \eta^2} + 2P(\eta \frac{\partial U}{\partial \eta} - \eta \frac{\partial U}{\partial \xi}) \right] = \lambda U, \quad (3.2)
\]

\[
-\frac{1}{1 + \xi^2} \left[ \frac{\partial U}{\partial \eta} + 4P^2 N + 2P(N + \eta \frac{\partial N}{\partial \xi}) \right] = \lambda N,
\]

\[U = 2PN \text{ at } \eta = 1,\]

where \(P\) is the Peclet number, \((\xi, \eta)\) are coordinates in the parabolic system, and \(\eta = N(\xi, t)\) represents the parameterized liquid-solid interface. The rightmost eigenvalues with positive real part correspond to the unstable modes that involve excitation of the temperature field and interface geometry. The eigenvalue problem (3.2) is discretized using centered finite differences over a two-dimensional domain with fifty unknowns in each direction. Using a Peclet number of 0.1, the resulting operator is of dimension 2500 and has the spectrum presented in Figure 3.8. There are many eigenvalues that have a positive real part, the five rightmost are documented in Table 3.5, so the solidification is found to be unstable.

Figure 3.8: Spectrum of crystal growth diffusion model (cry2500)
Table 3.5: Rightmost five eigenvalues of crystal growth diffusion model

<table>
<thead>
<tr>
<th>$\lambda$</th>
<th>Eigenvalue(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda_1$</td>
<td>3.2766</td>
</tr>
<tr>
<td>$\lambda_2$</td>
<td>3.0852</td>
</tr>
<tr>
<td>$\lambda_3$</td>
<td>2.9235</td>
</tr>
<tr>
<td>$\lambda_4$</td>
<td>2.7821</td>
</tr>
<tr>
<td>$\lambda_5$</td>
<td>2.6560</td>
</tr>
</tbody>
</table>

For this parameter study we will compute the rightmost eigenvalue of the crystal growth diffusion model using the shift-invert transformation, $(A - \sigma I)^{-1}$. The placement of the shift is on the positive real axis ($\sigma = 15$) and is the same shift used in previous experiments [98]. An incomplete LU preconditioner with a drop tolerance of $10^{-1}$ is used as the right-preconditioner for $A - \sigma I$. The convergence curves for each of the three iterative methods are presented in Figure 3.9. The approximate relative residual for each iterative method is indicated by the solid blue lines and the actual residual norm is indicated by a solid red symbol. Again, the GMRES method constructs a fixed-polynomial operator of lower degree than either BiCGSTAB or TFQMR for a fixed relative residual tolerance. However, the convergence curve for the TFQMR method indicates that a higher degree fixed-polynomial operator must be created for a fixed relative residual tolerance than either GMRES or BiCGSTAB.

Comparing the performance of the three fixed-polynomial operators will be done through a two-dimensional parameter study of the polynomial degree versus the dimension of the Krylov subspace. Considering the convergence curves for the crystal growth model, the polynomial degree was chosen to be from 20 to 56. These polynomials were used by the Arnoldi process to create Krylov subspaces ranging in dimension from 40 to 104. An approximation to the rightmost eigenpair is extracted from this subspace and the relative residual with $A$ is computed. These residuals, collected for
all combinations of polynomial degree and subspace dimensions, are displayed in the color plots in Figure 3.10.

The numerical results obtained for the crystal growth diffusion model again indicate that the GMRES fixed-polynomial operator is better than the BiCGSTAB or TFQMR fixed-polynomial operator. From the color plots in Figure 3.10 it is clear that none of the fixed-polynomial operators are able to compute the rightmost eigenpair for the smaller Krylov subspace dimensions. However, as the size of the subspace increases, a GMRES fixed-polynomial is less limited by the polynomial degree than BiCGSTAB or TFQMR. TFQMR does not approximate the rightmost eigenpair as well as GMRES or BiCGSTAB, even with a higher degree polynomial. The TFQMR parameter study also exhibits larger vertical bands of color corresponding to the plateaus in the convergence curve.
Figure 3.10: Parameter study of polynomial degree versus Krylov subspace dimension for computing the rightmost eigenpair of the crystal growth model; GMRES (top), BiCGSTAB (middle), and TFQMR (bottom)
3.3.3 Olmstead Model

The flow of a layer of viscoelastic fluid heated from below is modeled by the equations

\[
\frac{\partial u}{\partial t} = (1 - C) \frac{\partial^2 v}{\partial X^2} + C \frac{\partial^2 u}{\partial X^2} + Ru - u^3
\]

\[
B \frac{\partial v}{\partial t} = u - v
\]

where \( u \) is the speed of the fluid and \( v \) is related to viscoelastic forces. The domain is the interval from zero to one, \( \Omega = [0, 1] \), where Dirichlet boundary conditions are imposed for both \( u \) and \( v \). Centered finite differences are used to discretize the equations (3.3) over the unit interval with \( N = 1000 \) unknowns. Afterwards, the discretized equation can be written as \( \dot{x} = f(x) \) where \( x = [u_1, v_1, u_2, v_2, \ldots, u_N, v_N] \) and the dimension of the Jacobian matrix \( A = \partial f/\partial x \) is 2000. A linear stability analysis for this model problem was performed by Meerbergen and Spence [54] using the parameters \( B = 2, C = 0.1, R = 4.7, \) and trivial steady state \([u, v] = 0\). We will use those same parameters in our numerical experiments. The spectrum of the resulting matrix and a magnification of the rightmost eigenvalues can be found in Figure 3.11.

<table>
<thead>
<tr>
<th>( \lambda )</th>
<th>Eigenvalue(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \lambda_1 )</td>
<td>4.5101</td>
</tr>
<tr>
<td>( \lambda_2 )</td>
<td>3.8899</td>
</tr>
<tr>
<td>( \lambda_3 )</td>
<td>2.4065</td>
</tr>
<tr>
<td>( \lambda_4, \lambda_5 )</td>
<td>1.3000 ( \pm 1.9900 )</td>
</tr>
</tbody>
</table>

The Olmstead model was used by Meerbergen and Spence [54] to demonstrate the real rational spectral transformation

\[
I - (A - \alpha I)^{-2}(A - \sigma I)(A - \bar{\sigma} I)
\]
as an alternative to the shift-invert transformation

\[(A - \sigma I)^{-1} \].

The rational spectral transformation is especially useful when the rightmost eigenvalues are a complex conjugate pair \((R = 0.6)\). However, when \(R = 4.7\) the rightmost eigenvalues are real, so for this parameter study we will use the shift-invert transformation. The placement of the shift is on the positive real axis \((\sigma = 5)\). Once again, an incomplete LU preconditioner with a drop tolerance of \(10^{-1}\) is used as the right preconditioner for \(A - \sigma I\). The convergence curves for each of the three iterative methods are presented in Figure 3.12. The approximate relative residual for each iterative method is indicated by the solid blue lines and the actual residual norm is indicated by a solid red symbol.

The convergence curves in Figure 3.12 indicate the preconditioner constructed for the Olmstead model is very effective. So for this parameter study, the polynomial degree was chosen to range from 4 to 32. These polynomials were used by the Arnoldi process to create Krylov subspaces ranging in dimension from 6 to 36. Similar to the
other examples, an approximation to the rightmost eigenpair is extracted from this subspace and the relative residual with respect to $A$ is computed. These residuals, collected for all combinations of polynomial degree and subspace dimensions, are displayed in the color plots in Figure 3.13.

The numerical results for the Olmstead model contain fewer samples of the parameter space than the Brusselator model or the crystal structure model. However, they still indicate that the GMRES fixed-polynomial operator is better than the BiCGSTAB or TFQMR fixed-polynomial operator. For this eigenproblem the BiCGSTAB fixed-polynomial operator is severely limited by the polynomial degree. It takes polynomials of up to twice the degree of GMRES to achieve equal accuracy in the rightmost eigenpair. The TFQMR fixed-polynomial operator is also more limited by the polynomial degree than GMRES. A polynomial of up to five degrees more than GMRES is required to achieve equal accuracy. In the end, for any obtainable accuracy of the rightmost eigenpair, a GMRES fixed-polynomial operator can be constructed.
Figure 3.13: Parameter study of polynomial degree versus Krylov subspace dimension for computing the rightmost eigenpair of the Olmstead model; GMRES (top), BiCGSTAB (middle), and TFQMR (bottom)
of lower degree than either TFQMR or BiCGSTAB.

3.4 Numerical Comparison of the GMRES Fixed-Polynomial Operator and Jacobi-Davidson

The previous section contained the numerical results of a parameter study of the GMRES, BiCGSTAB, and TFQMR fixed-polynomial operators using some practical eigenproblems from science and engineering. The results indicate that the GMRES method constructs a lower-degree iteration polynomial that is a more consistent and effective preconditioner for IRA than those constructed by BiCGSTAB or TFQMR. This behavior justifies the additional memory and computational cost required by the GMRES fixed-polynomial operator. Thus, it will be the only fixed-polynomial operator utilized as a preconditioning technique in the remaining numerical experiments. In this section we will compare IRA with the GMRES fixed-polynomial operator to Jacobi-Davidson using an ocean circulation model and semiconductor laser model.

3.4.1 Wind-Driven Barotropic Ocean Model

This example will consider the stability of a single homogeneous layer of fluid driven by a sinusoidal wind stress. This fluid is enclosed by a rectangular basin of uniform depth on a mid-latitude $\beta$-plane, where the effect of the earth’s sphericity is modeled by a linear variation of the planetary vorticity [62]. IRA with the GMRES fixed-polynomial and Jacobi-Davidson (JDQR) will be used to perform the stability analyses on this wind-driven barotropic ocean model. The model will be introduced first, followed by the definition of its parameters. Some interesting behavior of the fixed-polynomial operator is presented before results are discussed.

Consider the potential vorticity/streamfunction formulation of the barotropic cir-
culation problem [51]
\[
\frac{\partial \psi}{\partial t} + J[\psi, \nabla^2 \psi] + \beta \frac{\partial \psi}{\partial x} = \nu \nabla^4 \psi + \frac{1}{\rho_0 H} \nabla \times \tau.
\]  
(3.4)

Here, \( \psi \) is the streamfunction, \( \tau \) is the wind stress, \( \beta \) is the meridional gradient of the planetary vorticity, \( \nu \) is the eddy viscosity, \( \rho_0 \) is the uniform ocean density, and \( H \) is the uniform depth of the ocean. \( J \) is the Jacobian operator
\[
J[\psi, \nabla^2 \psi] = -\frac{\partial \psi}{\partial y} \frac{\partial \nabla^2 \psi}{\partial x} + \frac{\partial \psi}{\partial x} \frac{\partial \nabla^2 \psi}{\partial y}.
\]

The boundary conditions for (3.4) are free-slip
\[
\nabla^2 \psi = 0 \text{ and } \frac{\partial^2 \nabla^2 \psi}{\partial n^2} = 0 \text{ on } \partial \Omega,
\]

where \( n \) is normal to the boundary.

This model idealizes the domain to a closed rectangular ocean basin \( \Omega \) where the wind forcing \( \tau \) is sinusoidal (Figure 3.14). This forcing produces a double-gyre with an anticyclonic gyre in the northern basin and a cyclonic gyre in the southern basin, as depicted by the streamfunctions in Figure 3.15.

The barotropic model (3.4) is nondimensionalized using a length scale corresponding to the horizontal extent of the domain \( L \) and a velocity scale \( U \) given by the Sverdrup relation [58]
\[
U = \frac{\pi \tau_0}{\rho_0 H \beta L}.
\]

A time-scale corresponding to \( L/U \) allows (3.4) to be rewritten as
\[
\frac{\partial q}{\partial t} + J[\psi, q] = \sin(\pi y) - \frac{\nu}{\beta L^3} \nabla^4 \psi,
\]

where \( q \) is the potential vorticity
\[
q = Ro \nabla^2 \psi + y,
\]

and \( Ro = \frac{U}{\beta L \tau} \) is the Rossby number.
A linear stability analysis of this barotropic flow model requires linearizing (3.5) about a steady state. This linearized problem is then discretized on the rectangular domain using the Arakawa discretization [1] for the Jacobian and centered differences for the other terms. Dirichlet boundary conditions are imposed. The resulting standard eigenvalue problem is sparse and the spectrum of \( A \) is complex. The stability of this model is determined by the rightmost eigenvalues of the discretized operator \( A \). An example of the spectrum and sparsity pattern of the linearized model is given in Figure 3.16.

The success of this preconditioning scheme depends upon the ability of the fixed-polynomial operator to approximate the spectral transformation. Restricting the degree of the polynomial has a definite affect on this. Furthermore, the quality of the preconditioner can profoundly impact the accuracy of this operator. A comparison of spectra from a sequence of restricted-degree, fixed-polynomial operators is displayed for various preconditioners. The base case is to analyze spectra from polynomials constructed without any preconditioner (Figure 3.17). For this example, it is obvious
Figure 3.15: Steady-state streamfunctions depicting double-gyre structure;
(a) $Ro = .0001$, (b) $Ro = .0016$, (c) $Ro = .0036$

that a low-degree polynomial does not contain enough information about the spectrum of the original matrix.

The quality and availability of a preconditioner is very important in the construction of a low-degree, fixed-polynomial operator. Previous results have shown that by using a modest preconditioner, the relative residual tolerance used to construct this operator does not have to be very low for the fixed-polynomial operator to have enough information about the spectrum. A good illustration of this trend is shown in Figure 3.17, where the same matrix previously used to construct the fixed-polynomial operator with no preconditioner (Figure 3.17) is now preconditioned using an incomplete LU factorization with a drop tolerance of $10^{-2}$. This sequence of spectra confirms that this preconditioning scheme has the ability to construct a low-dimensional, fixed-polynomial operator that approximates a spectral transformation very well.

For this example IRA with the GMRES fixed-polynomial is compared to Jacobi-
Davidson (JDQR) for computing the five rightmost eigenvalues of the nondimensionalized model (3.5) linearized about a steady state. The discretization of the linearized equation yields matrices that range in dimension from 700 to 3000. An incomplete LU factorization was used as a preconditioner for both methods. To determine any dependence on preconditioner quality, two drop tolerances were used: $10^{-3}$ and $10^{-4}$.

The previous discussion used an incomplete LU preconditioner with a drop tolerance of $10^{-2}$, which was sufficient for a problem of a smaller dimension. However, as the matrix dimension increases, a preconditioner constructed using that drop tolerance is insufficient for the convergence of either JDQR or IRA with the GMRES fixed-polynomial operator.

Similar to the small-scale Laplacian example, these two methods are to compute the same number of eigenvalues for the same cost. Then the comparison is in the accuracy of the computed eigenvalues. Depending upon the quality, each preconditioner is assigned a cost level which is determined by the number of matrix-vector

Figure 3.16: Sparsity pattern and spectrum of linearized model; dim = 903
Figure 3.17: Spectra of restricted-degree, fixed-polynomial operators; dim = 903; no preconditioner (top) and $M = \text{luinc}(A_\sigma, 10^{-2})$ (bottom)
products. The lower quality incomplete LU preconditioner has a drop tolerance of $10^{-3}$ and allows 1000 matrix-vector products (Figure 3.18), while the higher quality preconditioner has a drop tolerance of $10^{-4}$ and allows 500 matrix-vector products (Figure 3.19).

JDQR computes the five eigenvalues one-at-a-time to the desired tolerance. If the tolerance is set too high, not all five eigenvalues will be calculated. Thus, the tolerance for JDQR is different for each matrix dimension so that all five eigenvalues are calculated to the highest accuracy possible. These problems are then rerun using a fixed tolerance $10^{-6}$ with JDQR to analyze the number of eigenvalues it could compute at a higher tolerance. The GMRES fixed-polynomial operator is constructed with a relative tolerance of $10^{-6}$.

The results indicate that if five eigenvalues are required, then IRA with the

Figure 3.18: Relative error for the five rightmost eigenvalues; $M = \text{luinc}(A_\sigma, 10^{-3})$ (1000 matrix-vector products)
3.4.2 Semiconductor Laser Model

For a 2D dynamic rate equation model of a semiconductor laser, the time dependent field interaction is crucial. It is necessary to calculate the electric field and confinement factor with time during the evolution of the pulses. This requires the solution to the 2D scalar wave equation

$$\frac{\partial^2 E(x, y)}{\partial x^2} + \frac{\partial^2 E(x, y)}{\partial y^2} + (\eta(x, y)^2 k_o^2 - \beta^2) E(x, y) = 0$$

(3.6)

on the rectangular domain $\Omega = [0, L_x] \times [0, L_y]$, subject to Dirichlet boundary conditions, every pico-second. For a time period of 5 nano-seconds, the scalar wave
equation needs to be solved 5000 times. In equation (3.6), \( E(x, y) \) is the electric field, \( \eta \) is the complex refractive index, \( \beta \) is the mode propagation constant, \( k_o = \frac{2\pi}{\lambda} \) is the wave number, and \( \lambda \) is the wavelength [99].

The matrix eigenvalue equation is obtained by using Galerkin’s method on the continuous equation (3.6), expanding the electric field using basis functions which are the products of sine functions,

\[
\phi_i(x, y) = \frac{2}{(L_x L_y)^{1/2}} \sin \left( \frac{m_i \pi x}{L_x} \right) \sin \left( \frac{n_i \pi y}{L_y} \right). \tag{3.7}
\]

Here the integers \( m_i \) and \( n_i \) are the wave numbers for the basis \( i \) in the \( x \) and \( y \) direction, respectively. Furthermore, \( N_x \) and \( N_y \) are the number of waves in the \( x \) and \( y \) direction, respectively, and \( N = N_x N_y \). The basis functions (3.7) vanish at the boundary of \( \Omega \) and form an orthonormal set of functions

\[
\langle \phi_i(x, y), \phi_j(x, y) \rangle = \int_0^{L_y} \int_0^{L_x} \phi_i(x, y) \phi_j(x, y) \, dx \, dy = \delta_{ij}.
\]

The electric field can be expanded in terms of this orthogonal basis as

\[
E(x, y) = \sum_{i=1}^{N} a_i \phi_i(x, y).
\]

This expansion can be substituted into the scalar wave equation (3.6), multiplied on the left side by \( \phi_j(x, y) \), and integrated over the domain \( \Omega \) to obtain the matrix equation

\[
\sum_{i=1}^{N} (A_{ij} - \beta \delta_{ij}) a_i = 0 \tag{3.8}
\]

where

\[
A_{ij} = \left( \frac{(m_i \pi)^2}{L_x} + \frac{(n_i \pi)^2}{L_y} \right) \delta_{ij} + k_o^2 \langle \phi_i(x, y) \eta(x, y), \phi_j(x, y) \eta(x, y) \rangle.
\]

The matrix equation (3.8) is complex and non-Hermitian due to the refractive index \( \eta \) in the second term of \( A_{ij} \).
The eigenvalues of the matrix equation (3.8) represent both bound and continuum modes of the waveguide. Thus, the eigenvalues of interest are those whose real part lie within a range of values determined by the semiconductor laser. Within that range, the eigenvalue with largest imaginary part is of interest because it corresponds to the mode with the largest modal gain. It is this eigenpair that can be used to accelerate the solution of the 2D dynamic rate equation model.

For this example, fifteen waves are used in each of the $x$ and $y$ directions, resulting in a complex non-Hermitian matrix of dimension 225. Although the dimension of this matrix is small and computing a direct factorization may be inexpensive, there are a couple of reasons to approach this problem with an iterative eigensolver. One reason is the length of the sequence. It is computationally expensive to compute the direct factorization of 5000 matrices, even if they are of small dimension. If only a couple eigenvalues are necessary, then iterative eigensolvers may be more cost effective. Depending upon the problem, the eigenvalues of interest for a matrix sequence may be related from one time-step to the next. This information can be exploited by IRA with the GMRES fixed-polynomial operator or Jacobi-Davidson (JDQR) to accelerate convergence.

This example was provided by Canice O’Brien, a former researcher from the Physics Department at Trinity College Dublin who recognized that direct methods were too computationally expensive. The short sequence of 20 matrix equations provided through this collaboration depicts an extremely coarse time discretization of a nano-second time period. The results from this short sequence will allow speculation about the performance of IRA with the GMRES fixed-polynomial on matrix sequences and how it compares to JDQR. An example of the spectrum is given in Figure 3.20. The eigenvalues of interest are well-separated, but not in an easy part of the spectrum for IRA to compute eigenvalues without a spectral transformation. However, a shift can be obtained using knowledge about the range where the eigenvalues of interest are expected to lie.
IRA with the GMRES fixed-polynomial and JDQR are used to track the two eigenvalues of largest imaginary part whose real parts are in the range of interest. An LU factorization of the first matrix in the sequence is used to compute the eigenvalues and eigenvectors of interest, then employed as a preconditioner for all of the subsequent matrices in the sequence. The quality of the LU factorization as a preconditioner for these matrices is tracked to see if it degrades and should be refactored. The shift for the spectral transformation is held constant throughout the sequence of matrices ($\sigma = 11.30$). Both these methods are accelerated by using the eigenvectors corresponding to the eigenvalues of interest from one matrix in the sequence as an initial guess for eigenvectors of the next matrix in the sequence. The tolerance for JDQR is set at $10^{-6}$ and the GMRES fixed-polynomial operator is constructed with a relative tolerance of $10^{-4}$.

First, it should be noted that the eigenvalues of the operators do not vary much throughout the sequence (Figure 3.21). The eigenvalues change color from dark to
light to show the progression through the sequence of matrices which appears to have an elliptical path. This periodic behavior is a result of the periodic pulsed light output from the semiconductor laser [99]. Thus the LU factorization of the initial operator is a great preconditioner for the later matrices in the sequence.

The relative errors of the computed eigenvalues for both methods are illustrated in Figure 3.22. These errors are about the same for both methods. The dotted line in this figure indicates the tolerance used in constructing the GMRES fixed-polynomial. It is interesting to note that the relative errors of the eigenvalues computed using IRA with the GMRES fixed-polynomial are more dependent upon the residual reduction of GMRES than the imposed tolerance. JDQR has residuals that are more consistent with the imposed tolerance.

The periodicity in the quality of the preconditioner is apparent in the top plot of Figure 3.22. As would be expected, the initial LU factorization becomes a better preconditioner at the end of the matrix sequence. Refactoring the matrix at the
Figure 3.22: Relative errors for eigenvalues of interest (top), quality of preconditioner and cost of IRA with GMRES fixed-polynomial operator (bottom)
peak of this period to obtain a better preconditioner might be considered. The cost of this refactorization should not be ignored, especially when the current preconditioner improves in quality later in the sequence.

The performance of both methods in computing these two eigenvalues for each matrix in the sequence is indicated by the bottom plot of Figure 3.22. The computational cost of JDQR and IRA with the GMRES fixed-polynomial operator is about the same for all the matrices in the sequence. There are a few instances where JDQR is less expensive, but overall the two methods are comparable. It is interesting to note that these instances coincide with the deterioration in the quality of the preconditioner, which may indicate that GMRES fixed-polynomial is sensitive to this. It also is possible that JDQR takes more advantage of the information from the previous eigenvectors in computing the eigenvectors of the next operator. Either way, both methods are more efficient than employing a dense method to track the eigenvalues of interest for a matrix sequence.
Chapter 4

Performance Study for a Large-Scale Application: 8:1 Thermal Cavity Problem

In June of 2001, a special session at the First MIT Conference on Computational Fluid and Solid Dynamics was dedicated to understanding the fluid dynamics of the 8:1 thermally driven cavity [14]. This simple model problem was chosen as a representative of buoyancy-driven enclosure flows that are capable of transitioning between steady to various modes of unsteady, time-dependent flow. These types of enclosure flows appear in thermal design problems and can present significant issues, like unpredictable distortion of a laser passing through the fluid [72].

Several benchmark calculations were performed on this model problem, including the determination of the critical Rayleigh number at which the steady flow becomes time-dependent. This requires the use of an eigensolver to perform a linear stability analysis of the steady solutions. Numerical experiments obtained the critical value of the Rayleigh number using the Arnoldi method with a Möbius (generalized Cayley) transformation [74, 72]. Since the computations were performed for meshes with the number of unknowns ranging from 23,940 to 708,292 in 2D and 330,000 to 20 million...
in 3D, the GMRES iterative method was employed to efficiently apply the Cayley transformation within P_ARPACK [50].

In this chapter, we will compare the GMRES iterative method and the GMRES fixed-polynomial operator in the computation of the critical Rayleigh number of this model problem for several large meshes. First we will briefly discuss the model problem as presented in [14] and the numerical methods used in the previous calculations [74]. Then we will use the same numerical methods and software to perform the comparisons for the 2D and 3D model problems.

4.1 Problem Definition

Consider the two-dimensional enclosed cavity in Figure 4.1 with width $W$, height $H$, and an aspect ratio $A = H/W$ of 8. The governing equations that describe the thermal convective flow in this cavity are the incompressible Navier-Stokes equations, conservation of mass, and the energy equation. Since this flow is accompanied by heat transfer, the fluid properties will often be a function of temperature. Therefore, the non-dimensional governing equations are written in terms of temperature $\theta$ as

\[
\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = -\nabla P + \sqrt{\frac{Pr}{Ra}} \nabla^2 \mathbf{u} + \hat{j} \theta \tag{4.1}
\]

\[
\nabla \cdot \mathbf{u} = 0 \tag{4.2}
\]

\[
\frac{\partial \theta}{\partial t} + \mathbf{u} \cdot \nabla \theta = \frac{1}{\sqrt{Ra \cdot Pr}} \nabla^2 \theta, \tag{4.3}
\]

where $\mathbf{u} = (u, v)$, $P$, and $\hat{j}$ are the velocity, the deviation from hydrostatic pressure, and unit vector in the $y$ direction, respectively [14]. The Boussinesq approximation is assumed for this model so only small temperature variations from the mean are allowed. The dimensionless parameters $Pr$ and $Ra$ are the Prandtl number and the Rayleigh number, respectively. The Prandtl number determines the efficacy of convection versus heat conduction in transferring energy from an area. Similar to the previous benchmark calculations, this will be set at $Pr = 0.71$ for all numerical
Figure 4.1: Geometry of the 2D thermal cavity with insulated horizontal walls and constant temperature vertical walls

calculations. The Rayleigh number is associated with the heat transfer within the fluid and was used as the free variable for the linear stability analysis.

The boundary conditions for the thermal cavity are simple. The walls are no-slip and no-penetration, which is equivalent to Dirichlet boundary conditions for the velocities

\[ u = v = 0. \]

The thermal boundary conditions for the insulated walls are adiabatic

\[ \frac{\partial \theta}{\partial y} \bigg|_{y=0} = 0 \quad \frac{\partial \theta}{\partial y} \bigg|_{y=H} = 0 \]

because they are impermeable to heat transfer. The thermal boundary conditions for
the vertical walls are constantly hot on the left and cold on the right

\[ \theta |_{x=0} = \frac{1}{2}, \quad \theta |_{x=W} = -\frac{1}{2}. \]

### 4.2 Numerical Methods

This section will provide a brief discussion of the numerical methods and software used to discretize the continuous problem, compute the steady states, and perform the linear stability analysis. A more detailed discussion can be found with the original calculations performed by Salinger, et al. [74, 72].

The finite element formulation used to approximate the continuous problem is the Galerkin least-squares (GLS) formulation [12, 33, 78, 88] with pressure stabilization. The pressure-stabilized Petrov-Galerkin (PSPG) formulation enhances the stability of the Galerkin method and allows for equal order interpolation of pressure and velocity. Upwinding terms for the momentum (4.1) and energy (4.3) were also employed to provide stability for highly convective flows. This is known as the streamline-upwind Petrov-Galerkin (SUPG) method and it is used in the two-dimensional calculations in Section 4.3, but not the three-dimensional calculations in Section 4.4.

The discretization of the continuous problem can be written as a system of ordinary differential equations

\[ F(\dot{x}, x, Ra) = 0, \quad (4.4) \]
\[ x(0) = x_0. \quad (4.5) \]

In this discussion, \( F : \mathbb{R}^{n+1} \rightarrow \mathbb{R}^n \) is a nonlinear function of the time-dependent terms, nodal unknowns, and the Rayleigh number, respectively. Given a Rayleigh number \( Ra \), the corresponding solution curve of (4.4) and (4.5) is called a trajectory. The steady state solution is any solution that satisfies (4.4) where \( \dot{x} = 0 \) or, equivalently, satisfies the condition \( x(t) = x_0 \) for all \( t \). Assume \( F \) is continuously differentiable and its Jacobian is \( J(\dot{x}, x, Ra) \in \mathbb{R}^{n \times n} \). Then the steady state solution can be computed
using Newton’s method, which determines a sequence of approximate solutions for

\[ F(0, x_k + s_k, Ra) = 0. \]

The update \( s_k \) to the current iterate \( x_k \) is obtained through the solution of the Newton equation

\[ J(0, x_k, Ra) s_k = -F(0, x_k, Ra). \]

Practical implementations of Newton’s method solve the Newton equation using an iterative solver. This is considered an *inexact* Newton’s method if the update \( s_k \) obeys the condition

\[ \| F(0, x_k, Ra) + J(0, x_k, Ra) s_k \| \leq \eta_k \| F(0, x_k, Ra) s_k \|, \]

where \( \eta_k \in [0, 1) \) is chosen to force the residual of (4.6) to be small. The numerical results presented in this section use the MPSalsa parallel finite element computational fluid dynamics (CFD) simulator [76]. MPSalsa computes the steady state solution to (4.4) and (4.5) using an inexact Newton’s method [77]. The Newton equation (4.6) is solved using GMRES from the Aztec [34] library of distributed memory parallel iterative methods.

Once the steady state solution of (4.4) and (4.5) is computed for a given Rayleigh number, a linear stability analysis can be performed. This analysis linearizes (4.4) about the steady state solution \((x_0, Ra)\) and produces a generalized eigenvalue problem

\[ J z = \lambda B z, \]

where \( J \) is the Jacobian matrix and \( B \) is the mass matrix. The eigenvalues of (4.8) with the largest real part (rightmost) provide information about the effect of small perturbations of \( x_0 \). If any eigenvalue has a real part greater than zero, then the steady state is *unstable*. If the rightmost eigenvalue has a zero real part and is a
single multiplicity, then the steady state is stable. It is unstable if the rightmost eigenvalue has a multiplicity larger than one and zero real part. The steady state is asymptotically stable if all the eigenvalues have negative real part.

Determining the critical Rayleigh number at which the steady flow becomes time-dependent can be accomplished by using continuation methods. Numerical continuation methods compute a set of steady state solutions to (4.4) while the Rayleigh number is varied. Following a steady state solution branch makes it easier to locate the critical Rayleigh number, which is the Rayleigh number where the real part of the rightmost eigenvalue passes through zero. This is also called a bifurcation point. In this thermal cavity problem, the steady solutions encounter a certain type of bifurcation called a Hopf bifurcation. This is where a steady solution becomes unstable to oscillatory modes resulting in the rightmost eigenvalues of (4.8) becoming a purely imaginary complex conjugate pair. The numerical results presented in this chapter use the LOCA library [73] to locate and track bifurcation points. The eigenvalues of (4.8) are computed using the Arnoldi method in P_ARPACK [50].

While the shift-invert spectral transformation

\[ \psi_{SI}(J; B) = (J - \sigma B)^{-1}B \]

is often used to compute the rightmost eigenvalues of (4.8), the generalized Cayley transformation

\[ \psi_C(J; B) = (J - \sigma B)^{-1}(J - \mu B) \quad (4.9) \]

is more effective in the detection of Hopf bifurcations [53]. In practice, shift-invert transformations use a real shift \( \sigma \in \mathbb{R} \), so eigenvalues of large imaginary part are often mapped to eigenvalues of \( \psi_{SI}(J; B) \) with small modulus. However, the generalized Cayley transform (4.9) has the property that the eigenvalues of (4.8) to left of the centerline

\[ \mathcal{L} = \{ \gamma : \text{Re}(\gamma) = \frac{1}{2}(\alpha + \mu) \} \]
are mapped inside the unit circle, while the eigenvalues to the right are mapped outside the unit circle. Hence, the proper selection of \( \alpha \) and \( \mu \) makes the generalized Cayley transform better for computing any eigenvalue that has a positive real part. This is beneficial for detecting Hopf bifurcations, especially when the rightmost eigenvalues have a large imaginary part.

The eigenvalue computations presented in this chapter utilize a Cayley transformation \((4.9)\), which requires a solution to the linear system

\[
(J - \sigma B)x = (J - \mu B)b
\]

at each step of Arnoldi’s method. This solution was computed iteratively using unrestarted, preconditioned GMRES from the Aztec library in the original stability computations. The preconditioner used with GMRES was an incomplete LU factorization (ILU) with no subdomain overlap, and a fill level of 2.0. We will make a comparison between using unrestarted, ILU-preconditioned GMRES and the ILU-preconditioned GMRES fixed-polynomial operator for solving \((4.10)\) within P_ARPACK to locate the critical Rayleigh number.

### 4.3 2D Numerical Results

In this section we will present the results obtained from computing the rightmost eigenvalues of the discretized 2D thermal cavity problem for a given Rayleigh number \( Ra \). The discretization we will consider has 10,877 nodes, resulting in a matrix of dimension 43,508. Comparing the performance of preconditioned, unrestarted GMRES and the preconditioned GMRES fixed-polynomial operator will be done through a two-dimensional parameter study of the GMRES iteration polynomial degree versus the dimension of the Krylov subspace. The Rayleigh number will be chosen to slightly exceed the critical Rayleigh number, so the rightmost eigenvalues will have a positive real part. A steady state solution will be computed for the chosen Rayleigh number and a linear stability analysis is performed.
An approximate spectrum of the discretized 2D thermal cavity problem for $Ra = 3.87 \times 10^5$ is illustrated in Figure 4.2. The rightmost eigenvalues have passed through zero, so the critical Rayleigh number has been exceeded. Furthermore, this spectrum indicates that the magnitude of the imaginary part of the rightmost eigenvalue is around 1000, which supports the employment of the Cayley transform for this problem. The pole ($\sigma$) and zero ($\mu$) of the Cayley transform (4.9) as well as the Rayleigh number used in the numerical experiments are documented in Table 4.1.

Table 4.1: Cayley transform parameters and Rayleigh number for 2D thermal cavity problem

<table>
<thead>
<tr>
<th>Dimension ($N$)</th>
<th>Cayley transform ($\sigma/\mu$)</th>
<th>Rayleigh number</th>
</tr>
</thead>
<tbody>
<tr>
<td>43,508</td>
<td>2000/−2000</td>
<td>$3.87 \times 10^5$</td>
</tr>
</tbody>
</table>
The numerical results for the discretized thermal cavity problem were obtained using the Computer Science Research Institute (CSRI) QED cluster at Sandia National Laboratories. QED has 33 compute nodes with the specifications given in Table 4.2. The degree of the iteration polynomial used by the GMRES iterative solver and GMRES fixed-polynomial operator is varied from 100 to 150 and the dimension of the Krylov subspace used in P_ARPACK ranges from 100 to 150. The Arnoldi method used within P_ARPACK is unrestarted, so the total number of matrix and preconditioner applications is equal to the polynomial degree times the dimension of the Krylov subspace.

First we will examine the relative residual of the rightmost eigenpair computed in this parameter study (Figure 4.3). The rightmost eigenvalue is obtained both directly by transforming the computed eigenvalue and through the Rayleigh quotient of the computed eigenvector. From these plots we can see that the rightmost complex eigenpair computed using either the GMRES iterative solver or the GMRES fixed-polynomial operator is about the same quality for any combination of polynomial degree and Krylov subspace dimension. Furthermore, neither approach for obtaining the rightmost eigenvalue has the advantage of improving the quality of the eigenpair.

Table 4.2: Specification for a single QED compute node used to obtain results for thermal cavity problem

<table>
<thead>
<tr>
<th>Specification</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processors</td>
<td>2</td>
</tr>
<tr>
<td>CPU</td>
<td>Intel® Xeon™ 2.80GHz</td>
</tr>
<tr>
<td>L2 Cache</td>
<td>512 KB</td>
</tr>
<tr>
<td>RAM</td>
<td>2GB (shared)</td>
</tr>
</tbody>
</table>
Figure 4.3: Parameter study of polynomial degree versus Krylov subspace dimension for computing the rightmost eigenpair of the 2D thermal cavity problem ($N = 43,508$); relative residuals (log) are compared for the rightmost eigenvalues obtained using a direct transformation (top) or the Rayleigh quotient (bottom) by the GMRES iterative solver (left) and GMRES fixed-polynomial operator (right).
The time required to compute the rightmost eigenpair is noteworthy, especially since the quality of the eigenpair is comparable for the 2D thermal cavity problem. However, as a result of its software design, each run of MPSalsa requires one nonlinear iteration to obtain the steady state solution before performing the linear stability analysis in LOCA. This introduces variability into the total wall-clock time reported by the linear stability analysis process. Thus, we will compare the total wall-clock time for using the GMRES iterative method in Aztec versus applying the GMRES iteration polynomial within P_ARPACK to compute the rightmost eigenpair. For the GMRES fixed-polynomial operator, the time required to construct the GMRES iteration polynomial will also be included in its total wall-clock time.

Figure 4.4: Relative linear solver time (GMRES iterative solver / GMRES fixed-polynomial operator) for computing the rightmost eigenpair of the 2D thermal cavity problem (N = 43,508)

The wall-clock time required to compute the rightmost eigenpair for the 2D thermal cavity problem for any given polynomial degree and Krylov subspace dimension can be found in Figure 4.4. From this illustration it appears that by using the GMRES
fixed-polynomial operator with P_ARPACK we can obtain the same quality solution in less time than required by P_ARPACK with the GMRES iterative method. In fact, the gap in wall-clock time between using the GMRES fixed-polynomial operator and the GMRES iterative method slightly increases with the degree of the polynomial or dimension of the Krylov subspace.

A reason for the disparity in the wall-clock time was provided during the cost comparison of the GMRES iterative method and GMRES fixed-polynomial operator (see Section 2.5). These parameter studies hold the number of applications of the matrix and preconditioner constant to gauge the effectiveness of the fixed-polynomial operator constructed under these constraints. So the only difference between the GMRES fixed-polynomial operator and a GMRES iterative solver for these experiments is the computation of the orthogonalization coefficients and the solution of the least squares problem. The former of these computations requires synchronization and communication for each coefficient, the number of which grows quadratically with the degree of the polynomial. The penalty for computing these coefficients is dependent upon the system architecture and parallel distribution of the eigenproblem.

To further investigate the disparity in the wall-clock times between the GMRES iterative method and the GMRES fixed-polynomial operator we will look at the average wall-clock time of the linear system solves. Table 4.3 provides the details of the average time required for a linear system solve, given a fixed iteration polynomial degree. The average operator and preconditioner time for the linear system solve are also given, as well as the percentage of the solve time that is not spent applying the operator or preconditioner. From this information we see that, as expected, the GMRES iterative method spends a larger percentage of the time performing functions that do not include applying the operator or preconditioner. However, for both the GMRES iterative method and GMRES fixed-polynomial operator this percentage grows with the degree of the iteration polynomial at about the same rate.

The numerical results obtained for the 2D thermal cavity problem indicate that
Table 4.3: Average wall-clock times for the linear system solve of the 2D thermal cavity problem (43,508) for a given iteration polynomial degree using the GMRES fixed-polynomial operator (fpo) and the unrestarted GMRES iterative method (iterative).

<table>
<thead>
<tr>
<th>Polynomial degree</th>
<th>Linear solver</th>
<th>Total solve time (sec.)</th>
<th>Operator time (sec.)</th>
<th>Preconditioner time (sec.)</th>
<th>% Solve time no Op/Prec</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>fpo</td>
<td>4.46</td>
<td>0.62</td>
<td>1.81</td>
<td>46%</td>
</tr>
<tr>
<td></td>
<td>iterative</td>
<td>5.62</td>
<td>0.63</td>
<td>1.84</td>
<td>56%</td>
</tr>
<tr>
<td>110</td>
<td>fpo</td>
<td>5.10</td>
<td>0.68</td>
<td>1.99</td>
<td>48%</td>
</tr>
<tr>
<td></td>
<td>iterative</td>
<td>6.55</td>
<td>0.70</td>
<td>2.02</td>
<td>58%</td>
</tr>
<tr>
<td>120</td>
<td>fpo</td>
<td>5.78</td>
<td>0.74</td>
<td>2.17</td>
<td>50%</td>
</tr>
<tr>
<td></td>
<td>iterative</td>
<td>7.50</td>
<td>0.76</td>
<td>2.21</td>
<td>60%</td>
</tr>
<tr>
<td>130</td>
<td>fpo</td>
<td>6.52</td>
<td>0.80</td>
<td>2.35</td>
<td>52%</td>
</tr>
<tr>
<td></td>
<td>iterative</td>
<td>8.57</td>
<td>0.82</td>
<td>2.39</td>
<td>63%</td>
</tr>
<tr>
<td>140</td>
<td>fpo</td>
<td>7.27</td>
<td>0.86</td>
<td>2.53</td>
<td>53%</td>
</tr>
<tr>
<td></td>
<td>iterative</td>
<td>9.66</td>
<td>0.89</td>
<td>2.57</td>
<td>64%</td>
</tr>
<tr>
<td>150</td>
<td>fpo</td>
<td>8.07</td>
<td>0.92</td>
<td>2.71</td>
<td>55%</td>
</tr>
<tr>
<td></td>
<td>iterative</td>
<td>10.77</td>
<td>0.95</td>
<td>2.76</td>
<td>66%</td>
</tr>
</tbody>
</table>

using the GMRES fixed-polynomial operator with PARPACK can provide a modest reduction of the computational time necessary to perform a linear stability analysis. However, these results were obtained for a moderate sized problem run on one dual-processor node of a computing cluster. The trends indicated by this 2D model problem would be more conclusive if they were supported by a larger thermal cavity problem. Therefore, in the next section, we will consider a 3D thermal cavity problem with over a million variables.
4.4 3D Numerical Results

At the Third MIT Conference on Computational Fluid and Solid Dynamics, Salinger presented numerical results from a computational stability study of the 3D thermal cavity problem [72]. The 3D model results from taking the 2D model presented in Section 4.1 and extruding it by a depth of 1.0. On the front and back surfaces, the walls are no-slip and no-penetration as well as adiabatic to ensure variation in the third dimension of the solution. The same numerical methods and software (see Section 4.2) are used to compute the steady state solution and perform the linear stability analysis for both the two and three-dimensional model. The only difference is that the upwinding terms (SUPG) are absent in the discretization of the 3D model.

![Approximate rightmost part of the spectrum of discretized 3D thermal cavity problem for $Ra = 5.57 \times 10^5$](image)

Figure 4.5: Approximate rightmost part of the spectrum of discretized 3D thermal cavity problem for $Ra = 5.57 \times 10^5$

In this section we will present the results obtained from computing the rightmost eigenvalues of the 3D thermal cavity problem for a given Rayleigh number. The discretization we will consider has 219,373 nodes, resulting in a matrix of dimension
1,096,865. The Rayleigh number is chosen to be $5.57 \times 10^5$, which slightly exceeds the critical Rayleigh number. This is illustrated by the approximate spectrum in Figure 4.5. The imaginary part of the rightmost eigenvalue still has a magnitude of around 1000, so the Cayley transform is utilized for solving the eigenproblem (4.8). The pole ($\sigma$) and zero ($\mu$) of the Cayley transform (4.9) as well as the Rayleigh number used in the numerical experiments are documented in Table 4.4.

Table 4.4: Cayley transform parameters and Rayleigh number for 3D thermal cavity problem

<table>
<thead>
<tr>
<th>Dimension ($N$)</th>
<th>Cayley transform ($\sigma/\mu$)</th>
<th>Rayleigh number</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,096,865</td>
<td>$800/ -800$</td>
<td>$5.57 \times 10^5$</td>
</tr>
</tbody>
</table>

Twenty nodes, or forty processors, on the QED cluster (see Table 4.2) were used to perform the numerical computations for the 3D thermal cavity problem. The degree of the iteration polynomial used by the GMRES iterative solver and GMRES fixed-polynomial operator is varied from 100 to 150, while the dimension of the Krylov subspace used in P_ARPACK is varied from 130 to 150. Again, the Arnoldi method used by P_ARPACK is unrestarted, so the total number of matrix and preconditioner applications is equal to the polynomial degree times the dimension of the Krylov subspace.

We will first examine the relative residual of the rightmost eigenpair computed in this parameter study. The rightmost eigenvalue is obtained both directly from transforming the computed eigenvalue and through the Rayleigh quotient of the computed eigenvector. From the plots in Figure 4.6 we can see that the rightmost complex eigenpair computed using either the GMRES iterative solver or the GMRES fixed-polynomial operator is about the same quality except for when the polynomial is of degree 120. In this case, the GMRES iterative solver computes an eigenpair that is only slightly better than the eigenpair computed by the GMRES fixed-polynomial op-
Figure 4.6: Parameter study of polynomial degree versus Krylov subspace dimension for computing the rightmost eigenpair of the 3D thermal cavity problem \((N = 1,096,865)\); relative residuals (log) are compared for the rightmost eigenvalues obtained using a direct transformation (left) or the Rayleigh quotient (right) by the GMRES fixed-polynomial operator (top) and GMRES iterative solver (bottom).

Even though the plots make the residuals of these eigenpairs look drastically different, the difference between the two is less than a half of an order of magnitude.

Similar to the 2D thermal cavity problem, the comparable quality of the rightmost eigenpair makes the time required to compute it noteworthy. The ratio of wall-clock time required in using the GMRES iterative method versus the GMRES fixed-polynomial operator for any given polynomial degree and Krylov subspace dimension can be found in Figure 4.7. This illustrates that by using the GMRES fixed-polynomial operator with \(P\_ARPACK\) we can obtain the same quality solution in less time that required by \(P\_ARPACK\) with the GMRES iterative method. The ratio, or gap between the wall-clock times, slightly increases with the degree of the polynomial or dimension of the Krylov subspace.

Compared to the two-dimensional results, the relative gap in the wall-clock times between the GMRES iterative method and the GMRES fixed-polynomial operator is lower for any polynomial degree or Krylov subspace dimension. This encourages further investigation of the average wall-clock time required for the linear system solves.
Figure 4.7: Relative linear solver time (GMRES iterative solver / GMRES fixed-polynomial operator) for computing the rightmost eigenpair of the 3D thermal cavity problem \((N = 1,096,865)\)

and what portion of that time is spent applying the operator or preconditioner. This information, provided in Table 4.5, indicates that applying the preconditioner and operator takes relatively much more time for the 3D thermal cavity problem than the 2D problem. For the 2D thermal cavity problem 8 – 11% of the total solution time for the GMRES iterative method was spent applying the operator and 26 – 33% of the time was spent applying the preconditioner. For the 3D thermal cavity problem this increased to 13 – 16% for the operator and 38 – 44% for the preconditioner. In the previous discussion about the differences between the GMRES iterative method and the GMRES fixed-polynomial-operator it was concluded that the computation of orthogonalization coefficients and the solution of the least squares problem were the only computational differences. If the relative cost of applying the operator or preconditioner increases, then the potential for saving time by precomputing the coefficients of the fixed-polynomial operator diminishes. This explains why the performance gains
Table 4.5: Average wall-clock times for the linear system solve of the 3D thermal cavity problem \((N = 1,096,865)\) for a given iteration polynomial degree using the GMRES fixed-polynomial operator (fpo) and the unrestarted GMRES iterative method (iterative).

<table>
<thead>
<tr>
<th>Polynomial degree</th>
<th>Linear solver</th>
<th>Total solve time (sec.)</th>
<th>Operator time (sec.)</th>
<th>Preconditioner time (sec.)</th>
<th>% Solve time no Op/Prec</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>fpo</td>
<td>13.59</td>
<td>2.62</td>
<td>7.00</td>
<td>29%</td>
</tr>
<tr>
<td></td>
<td>iterative</td>
<td>15.47</td>
<td>2.46</td>
<td>6.87</td>
<td>40%</td>
</tr>
<tr>
<td>110</td>
<td>fpo</td>
<td>15.06</td>
<td>2.82</td>
<td>7.58</td>
<td>31%</td>
</tr>
<tr>
<td></td>
<td>iterative</td>
<td>17.59</td>
<td>2.75</td>
<td>7.70</td>
<td>41%</td>
</tr>
<tr>
<td>120</td>
<td>fpo</td>
<td>16.08</td>
<td>3.00</td>
<td>7.88</td>
<td>32%</td>
</tr>
<tr>
<td></td>
<td>iterative</td>
<td>19.82</td>
<td>2.95</td>
<td>8.23</td>
<td>44%</td>
</tr>
<tr>
<td>130</td>
<td>fpo</td>
<td>17.88</td>
<td>3.25</td>
<td>8.54</td>
<td>34%</td>
</tr>
<tr>
<td></td>
<td>iterative</td>
<td>22.12</td>
<td>3.19</td>
<td>8.91</td>
<td>45%</td>
</tr>
<tr>
<td>140</td>
<td>fpo</td>
<td>19.72</td>
<td>3.49</td>
<td>9.19</td>
<td>36%</td>
</tr>
<tr>
<td></td>
<td>iterative</td>
<td>24.60</td>
<td>3.44</td>
<td>9.56</td>
<td>47%</td>
</tr>
<tr>
<td>150</td>
<td>fpo</td>
<td>21.59</td>
<td>3.77</td>
<td>9.78</td>
<td>37%</td>
</tr>
<tr>
<td></td>
<td>iterative</td>
<td>27.13</td>
<td>3.68</td>
<td>10.24</td>
<td>49%</td>
</tr>
</tbody>
</table>

obtained with the 2D thermal cavity problem were not observed with the 3D thermal cavity problem.
Chapter 5

Conclusions

This dissertation presented an alternative approach for approximating the spectral transformations often used to accelerate the convergence of the Implicitly Restarted Arnoldi (IRA) method to the rightmost eigenvalues of a non-Hermitian matrix. These approximate spectral transformations, or fixed-polynomial operators, are constructed using the iteration polynomials from three common iterative methods for non-Hermitian linear systems: GMRES, BiCGSTAB, and TFQMR. Considerable testing of all three fixed-polynomial operators was performed on several realistic eigenvalue problems from science and engineering. The numerical results in Chapter 3 and Chapter 4 conclusively find that fixed-polynomial operators are a competitive preconditioning technique for IRA, often outperforming iterative solvers. Furthermore, IRA with a fixed-polynomial operator is shown to be competitive with other popular eigensolvers like Jacobi-Davidson and LOBPCG.

Details of the three fixed-polynomial operators and their implementations were presented in Chapter 2. This includes a novel derivation of the BiCGSTAB and TFQMR iteration polynomials. The discussion of implementation details illustrates the significant impact of precomputing these operators on the simplicity of use and reduction of computational expense through avoidance of inner-products. Approximate eigenpairs of the original problem can be easily and directly reconstructed from
the computed eigenpairs of the fixed-polynomial operator. The accuracy of eigenpairs computed in this way are dependent upon the norm of the residual polynomial, \( \| \psi(A_{\sigma}M_{\sigma}^{-1}) \| \) or the maximum magnitude of the residual polynomial over the spectrum of \( A_{\sigma}M_{\sigma}^{-1} \) if it is diagonalizable. These residual bounds demonstrate the ability of this approach to compute approximate eigenpairs of the original eigenproblem from a fixed-polynomial operator, while avoiding the restrictive accuracy requirements necessitated by using a spectral transformation with an iterative solver.

The numerical results presented in Chapter 3 enable us to conclude that using fixed-polynomial operators with IRA produces a method competitive with other popular eigensolvers and preconditioning techniques. Using IRA with a GMRES fixed-polynomial operator compares favorably against Jacobi-Davidson and LOBPCG on small-scale Hermitian problems. However, the accuracy of IRA with this preconditioning scheme suffers for eigenvalues further away from the shift. Considerable testing of the three fixed-polynomial operators indicate that the GMRES method constructs a lower-degree iteration polynomial that is a more consistent and effective preconditioner for non-Hermitian eigenvalue problems than either BiCGSTAB or TFQMR. Numerical results from the linear stability analysis of a wind-driven barotropic ocean circulation model show that if multiple eigenvalues are required, then IRA with a GMRES fixed-polynomial operator can be more accurate than Jacobi-Davidson for a fixed computational cost.

The performance of the GMRES fixed-polynomial as a preconditioning technique for IRA on large-scale non-Hermitian eigenproblems was the subject of Chapter 4. The eigenproblems presented in this chapter came from the linear stability analysis of a previously studied thermally-driven cavity problem. The two and three-dimensional model problems were both considered and their discretizations resulted in matrices of dimension 43,508 and 1,096,865, respectively. We compared the performance of using the GMRES iterative method versus the GMRES fixed-polynomial operator to approximate a Cayley transformation, which is required to efficiently compute of the
rightmost eigenvalues of these matrices. These computations were performed on a parallel distributed memory machine using mature software packages for modeling the fluid dynamics (MPSalsa), performing bifurcation tracking (LOCA), computing the eigenvalues (P_ARPACK), and solving the linear systems (Aztec). The numerical results obtained for the two and three-dimensional thermal cavity problem indicate that using the GMRES fixed-polynomial operator with P_ARPACK can provide a significant reduction of the computational time necessary to perform a linear stability analysis.

5.1 Future Directions

These results provide a compelling argument for the use of fixed-polynomial operators in approximating the spectral transformations often used to accelerate the convergence of the Implicitly Restarted Arnoldi method. However, several questions about this preconditioning technique remain:

- The feasibility of a convergence theory for Krylov subspace methods applied to this fixed-polynomial operator is uncertain. An attempt to develop a convergence theory will require a subspace approximation theory approach [7] for IRA.

- It is uncertain if this preconditioning scheme can aid IRA in computing interior eigenvalues. The current results are only encouraging for the computation of the rightmost eigenvalues. An error bound should be found for the rightmost eigenvalues of the fixed-polynomial as approximations to those of the spectral transformation.

- Current heuristics for the accuracy of a computed eigenpair include a bound that is dependent upon \( \|p(A_\sigma M_\sigma^{-1})\| \). Unfortunately, the iterative method used to construct \( p \) can only offer a bound for \( \|p(A_\sigma M_\sigma^{-1})v_1\| \), where \( v_1 \) is the ini-
tial vector. Alternative derivations for an accuracy bound on the computed eigenpair are currently being considered which take advantage of the optimality condition provided by the Rayleigh quotient. Another avenue of investigation would be to derive a probabilistic bound [39].
Bibliography


[35] Z. Jia. The convergence of generalized lanczos methods for large unsymmetric


[53] K. Meerbergen, A. Spence, and D. Roose. Shift-invert and Cayley transforms for
detection of the rightmost eigenvalues of nonsymmetric matrices. *BIT*, 34:409–

[54] Karl Meerbergen and Alastair Spence. A spectral transformation for finding
complex eigenvalues of large sparse nonsymmetric matrices. Report TW 219,
Department of Computer Science, Katheolieke Universiteit Leuven, Belgium,
1994.


[56] Ronald B. Morgan and Min Zeng. Harmonic projection methods for large non–

[57] Noel M. Nachtigal, Lothar Reichel, and Lloyd N. Trefethen. A hybrid GMRES

[58] Balasubramanya T. Nadiga and Benjamin P. Luce. Global bifurcation of


[60] C. C. Paige and M. A. Saunders. LSQR: An algorithm for sparse linear equations


