KRYLOV SUBSPACE METHODS WITH FIXED MEMORY REQUIREMENTS: NEARLY HERMITIAN LINEAR SYSTEMS AND SUBSPACE RECYCLING

A Dissertation
Submitted to
the Temple University Graduate Board

---

in Partial Fulfillment
of the Requirements for the Degree of
DOCTOR OF PHILOSOPHY

---

by
Kirk McLane Soodhalter
May, 2012

ABSTRACT

Krylov Subspace Methods with Fixed Memory Requirements: Nearly Hermitian Linear Systems and Subspace Recycling

Kirk McLane Soodhalter
DOCTOR OF PHILOSOPHY

Temple University, 2012

Professor Daniel B. Szyld, Chair

Krylov subspace iterative methods provide an effective tool for reducing the solution of large linear systems to a size for which a direct solver may be applied. However, the problems of limited storage and speed are still a concern. Therefore, in this dissertation work, we present iterative Krylov subspace algorithms for non-Hermitian systems which do have fixed memory requirements and have favorable convergence characteristics.

This dissertation describes three projects. The first project concerns short-term recurrence Krylov subspace methods for nearly-Hermitian linear systems. In 2008, Beckermann and Reichel introduced a short-term recurrence progressive

GMRES algorithm for nearly-Hermitian linear systems. However, we have found this method to be unstable. We document the instabilities and introduce a different fixed-memory algorithm to treat nearly-Hermitian problems. We present numerical experiments demonstrating that the performance of this algorithm is competitive.

The other two projects involve extending a strategy called Krylov subspace recycling, introduced by Parks and colleagues in 2005. This method requires more overhead than other subspace augmentation methods but offers the ability to recycle subspace information between cycles for a single linear system and recycle information between related linear systems.

In the first project, we extend subspace recycling to the block Krylov subspace setting. A block Krylov subspace is a generalization of Krylov subspace where a single starting vector is replaced with a block of linearly independent starting vectors. We then apply our method to a sequence of matrices arising in a Newton iteration applied to fluid density functional theory and present some numerical experiments.

In the second project, we extend the methods of subspace recycling to a family of linear systems differing only by multiples of the identity. These problems arise in the theory of quantum chromodynamics, a theory of the behavior of subatomic particles. We wish to build on the class of Krylov methods which allow the simultaneous solution of all shifted linear systems while generating only one subspace. However, the mechanics of subspace recycling complicates this situation and interferes with our ability to simultaneously solve all systems using these techniques. Therefore, we introduce an algorithm which avoids this complication and present some numerical experiments demonstrating its effectiveness.

ACKNOWLEDGEMENTS

I have worked incredibly hard, but I have also been quite fortunate to have been influenced and guided by good people along the way.

I have matured due to the guidance of my adviser, Daniel Szyld. I am quite lucky, since I ended up working with him by chance. Sinai Robins suggested I consider working with Daniel. I knew nothing of numerical analysis; but I approached him nonetheless, and thus began a great mentoring relationship, fruitful research collaboration, and a great friendship. Since then, Daniel has fostered my development as an independent researcher and taught me to be a good member of the research community. As much as he expected from me, Daniel always insisted that I live a balanced life and take breaks from my work to enjoy all that the city of Philadelphia has to offer. He has been the consummate adviser and a good friend.

I have also had the pleasure of working with Benjamin Seibold at Temple University. From him, I have learned much about how to think as an applied mathematician and to always be curious. His polite but adversarial style of questioning has made me a better presenter of my own ideas, and I have learned to use that same style when questioning others.

I have had the privilege to collaborate with Mark Embree, Josef Sifuentes, and Fei Xue. Working with Mark, Josef and Fei has been a great learning experience. I have learned it is okay to fail and have bad ideas. Successful research requires that one learns to accept failure and change tactics. They helped me gain the confidence to share my ideas and test them, knowing that failure will provide information for the path forward to success.

In the most recent summer, I was able to secure a summer internship at Sandia National Laboratories in Albuquerque, New Mexico under the direction of Michael Parks. Mike was a great mentor, and those experiences were priceless. I came away from working with Mike feeling more confident in my abilities.

At Tulane University, I was lucky to have a great mentor in Victor Moll. Victor became my advisor and directed my undergraduate research project, culminating in an honors thesis at Tulane. When I was applying to graduate schools, Victor recommended Temple University, feeling that it would be a good fit. I ended up at Temple, and I could not agree more with this assessment.

In middle school, Ann Chisholm recognized that I had some aptitude in mathematics and placed me in accelerated classes. In high school, Catherine Doxtater made calculus interesting. In her care, I realized for the first time that I enjoyed math and felt that I had a strong understanding of the material.

Finally, I have reaped the benefits of parents who have supported me in all my endeavors and equipped me with good values. They have sacrificed a great deal for me to take this path. I also want thank Jim Stout for his support. My large Greek family also supported me greatly when I was unsure if I wanted to continue.

DEDICATION

To my parents, Charles and Denise Soodhalter.

This thesis is also dedicated to Marvin Knopp
whom we lost unexpectedly this year.
Marvin took interest in my progress and offered his advice
and wisdom freely during my time at Temple.
Thank you for making time for conversation
and sharing some great stories with me.

TABLE OF CONTENTS

Page

ABSTRACT . . . . **ii**

ACKNOWLEDGEMENTS . . . . **iv**

DEDICATION . . . . **vi**

LIST OF FIGURES . . . . **x**

LIST OF TABLES . . . . **xvi**

LIST OF ALGORITHMS . . . . **xvii**

CHAPTER

1. INTRODUCTION **1**

2. NOTATION AND BACKGROUND **5**

2.1 Notation 5

2.2 Materials and Methods 6

2.3 Background 7

3. NEARLY-HERMITIAN LINEAR SYSTEMS **39**

3.1 Introduction 39
3.2 The Progressive GMRES Algorithm 40
3.2.1 Derivation of the Algorithm 40
3.2.2 Reinterpretation of Full Arnoldi 43
3.2.3 Instability of Progressive GMRES: Analysis and Illustration . 45
3.2.4 The Lippmann–Schwinger Equation 46
3.2.5 A class of simple examples 47
3.2.6 Analysis of PGMRES orthogonalization 51
3.3 The Schur Complement Method 56
3.3.1 Derivation 56
3.3.2 Schur Complement Method as a Preconditioner 65
3.3.3 Schur Complement Method Numerical Results 66
3.3.4 Simple example, revisited 67
3.3.5 Lippmann–Schwinger equation 70
3.3.6 A parallel experiment 71
3.3.7 Discussion and Conclusions 73
3.4 An Implementation of Block MINRES 75
3.4.1 Background 75
3.4.2 A Block Minimum Residual Method 78
3.4.3 Block MINRES for Hermitian Linear Systems 80
3.4.4 Implementation Details 83
3.4.5 Effectiveness of Block MINRES and Some Numerical Test Cases 86

4. SUBSPACE RECYCLING FOR SEQUENCES OF LINEAR SYSTEMS . . . 93

4.1 Introduction . . . 93

4.1.1 Block GMRES with Subspace Recycling . . . 94

4.1.2 Krylov Subspace Recycling for Families of Shifted Systems . . 94

4.2 Background . . . 95

4.2.1 GMRES with Subspace Recycling . . . 97

4.3 Block GMRES with Subspace Recycling . . . 101

4.3.1 Convergence Comparison with GCRODR for a Single Right-hand Side . . . 107

4.3.2 Recycled Subspace Selection Techniques . . . 110

4.3.3 Harmonic Ritz Vector Computation . . . 112

4.3.4 Experiments . . . 115

4.3.5 Conclusion . . . 118

4.4 GMRES with Subspace Recycling for Multiple Shifted Linear Systems 119

4.4.1 Introduction . . . 119

4.4.2 Background . . . 122

4.4.3 GCRODR For Shifted Systems . . . 126

4.4.4 An Approximate Collinearity Scheme . . . 139

4.4.5 Experiments . . . 148

4.4.6 Conclusion . . . 153

5. CONCLUSIONS . . . 156

BIBLIOGRAPHY . . . 158

LIST OF FIGURES

Figure Page

2.1 Computing the largest principal angle between the true Krylov subspace (as represented by a basis constructed using Householder Arnoldi) and the subspace spanned by the vectors generated by computing the Krylov basis and orthogonalizing. We use the `sherman5` matrix from Matrix Market [1] and the starting vector is the right-hand side is provided with the `sherman5` matrix. 20

2.2 The orthogonality residual ($\left\|\mathbf{I} - \mathbf{V}_j^*\mathbf{V}_j\right\|$ at the jth iteration) of three sets of bases of a Krylov subspace as the dimension j increases. The three methods are based on the Arnoldi procedure, but differ in how the orthogonalization step is performed (Gram-Schmidt, Modified Gram-Schmidt, Householder). We use the `sherman5` matrix from Matrix Market [1] and the starting vector is the right-hand side is provided with the `sherman5` matrix. 22

2.3 A comparison of two optimal Krylov subspace methods with nonoptimal counterparts. In the figure on the left, we compare the full orthogonalization method (FOM) with its truncated version. This implementation of the truncated version of FOM, called DIOM, is described in [75]. DIOM(20) means we orthogonalize the newest Krylov subspace vector against only the previous 20. In the figure on the right, we compare GMRES with restarted GMRES with cycle length 20.The system we are solving is the `sherman5` matrix from Matrix Market [1] preconditioned with `ILU(0)`. The right-hand side is provided with the `sherman5` matrix. 35

3.1 Comparison of PGMRES (solid line) to standard (modified Gram–Schmidt) GMRES (**dashed line**) applied to the one dimensional scattering problem for various values of the wave number κ. The left plot shows the relative residual norm for each method; the right plot shows the departure from orthonormality, along with the minimum singular value of the Arnoldi basis matrix $\mathbf{V}_j$ computed by the PGMRES recurrence (**gray line**). In exact arithmetic, the algorithms are identical; in finite precision, the performance of PGMRES degrades as κ increases. 48

3.2 Eigenvalues (●) of the matrix (3.14) in the complex plane, for $n = 200$ and $p = 6$. 50

3.3 PGMRES (**solid line**) and standard (modified Gram–Schmidt) GMRES (**dashed line**) as in Figure 3.1, applied to the simple example (3.14) with a vector $\mathbf{b} = [1, \ldots, 1]^{\mathrm{T}}$ and $n = 200$. The dotted line in each left plot shows the convergence bound (3.15). The gray line in each right plot shows the smallest singular value of the Arnoldi basis matrix $\mathbf{V}_j$ computed by PGMRES. In the well-conditioned top example ($\kappa(\mathbf{A}) = 32$), the instability develops gradually; in the ill-conditioned bottom example ($\kappa(\mathbf{A}) = 1.25 \times 10^6$), the instability is apparent at the fifth iteration. . . 52

3.4 Convergence of the Schur Complement Method for $\mathbf{Ax} = \mathbf{b}$, with coefficient matrix of dimension $n = 10^4$ given by the simple example (3.14) with a 2×2 identity added to the $\mathbf{Z}$ block, and parameters $\alpha = 0.125$, $\beta = 1$, and $p = 6$; the vector $\mathbf{b}$ is equal to $1/\sqrt{n}$ in all components. The gray line shows the upper bound γ_k in (3.32) on the residual norm. For context, the residual norms for full GMRES, PGMRES, and IDR(2) are also shown. 68

3.5 Comparison of the Schur Complement Method to GMRES (with Hermitian preconditioning) for the simple example (3.14) of dimension $n = 10^5$ with skew-Hermitian part of rank s. The eigenvalues of the Hermitian-preconditioned matrix fill the interval $[1 - 2i, 1 + 2i]$, so the condition number of this matrix is essentially fixed for all these experiments. . . . 69

3.6 Residual norms for the MINRES iterations for the discretized Lippmann–Schwinger integral equation for wave number $\kappa = 10$ on a grid of $n = 10^3$ quadrature points with refractive indices $m(x) \equiv -1$ (left) and $m(x) = -2 - \sin(x)$ (right). 71

3.7 Performance of block MINRES for different right-hand sides. In the figure on the left, the two right-hand sides are $\mathbf{b}_1 = \mathbf{e}_{n_1}^{(1)}$ and $\mathbf{b}_2 = \mathbf{1}$. In the figure on the right, $\mathbf{b}_1$ does not change, but $\hat{\mathbf{b}}_2 = \mathbf{e}_{n_1}^{(2)}$. 88

3.8 Magnitude of the components of different right-hand-sides in the eigenspace associated to the fifty eigenvectors associated to the smallest magnitude eigenvalues. 89

3.9 Performance of block MINRES after altering some components of the right-hand side $\mathbf{b}_2$. 90

4.1 Performance of restarted GMRES with different restart parameters for a two-dimensional convection diffusion problem $10\Delta u + \mathbf{a}^T(\nabla u) + u = f$ with a reaction term on the domain $\Omega = [0,1] \times [0,1]$, where Ω is discretized with a 500×500 grid, $\mathbf{a} \in \mathbb{C}^2$ is a vector of ones, and $f = f(x,y) = 1$ is a constant function. We assume the Dirichlet condition that the function is zero on the boundary. Restart parameters are 10, 50, 100, and 250. Full GMRES is also included for comparison. We preconditioned with `ILU(0)`. 96

4.2 The convergence curves of GMRES (**solid black curve**), GMRES(20) (**solid grey curve**), and GCRODR(18,2) (**dashed black curve**). Observe that the latter two methods have the same storage requirements. The coefficient matrix and right-hand side are from the `sherman5` system from Matrix Market [1]. We preconditioned with `ILU(0)`. We see that for this problem, it suffices to retain a two-dimensional subspace between restart cycles to achieve convergence almost as good as full GMRES. . . 101

4.3 Demonstration of the convergence of block GMRES (**dashed grey curve**) as compared to convergence of the methods used in Figure 4.2. For block GMRES, a second right-hand side was generated using Matlab's `rand()`. Convergence is measured using only the residual norm for the first right-hand side. One matrix-block-vector product is executed at each iteration. If the cost of one block-p matrix-vector product is equal to the cost of $\frac{2}{3}p$ single matrix-vector products, as in Figure 4.6, then block GMRES outperforms GMRES in this experiment. 104

4.4 Demonstration of the convergence of block GCRODR(10,2) (**gray dot-dashed curve**) as compared to GCRODR(10,2) (**black dashed curve**), block GMRES (**solid grey curve**), and GMRES (**solid black curve**). For block GMRES and block GCRODR, a second right-hand side was generated using Matlab's `rand()`. Convergence is measured using only the residual norm for the first right-hand side. One matrix-block-vector product is executed at each iteration. Here we see that block GCRODR(10,2) converges with the same number of iterations as GMRES but uses much less storage. 111

4.5 Comparison of the time taken to perform $p \times 10^5$ matrix-vector products for the sparse `sherman5` matrix for the single matrix-vector product (**grey**) and the block size p matrix-vector product (**black**). Different block sizes were tested. The initial target vector was generated using Matlab's `rand()` function, and the matrix simply acted upon this vector repeatedly. 116

4.6 In the figure on the left, we compare the time taken to perform 10^5 single matrix-vector products (**grey**) with how much time is taken to perform the same number of block matrix-vector products for different block sizes p (**black**). The figure on the right shows the ratio of time taken for the block matrix- vector products over the time taken for single matrix-vector products (**black solid line**). As predicted, the block matrix-vector product is not p times as expensive. The **dashed line** shows the ratio if a block p matrix-vector product was $\frac{2}{3}p$ times as expensive as a single matrix-vector product. 117

4.7 Performance of block GCRODR(100,50) on the `sherman5` matrix from Matrix Market [1] preconditioned with `ILU(0)`as we increase the number of right-hand sides. The first right-hand side is packaged with the matrix. The other right-hand sides are generated using Matlab's `rand()` function. Convergence is measured by computing the residual norm associated to the first right-hand side. Lightness of the convergence curves increases as we add more right-hand sides. 118

4.8 In the left-hand figure, we see the matrix-vector product count for block GCRODR solving linear systems involving the Jacobian for 7 sequential Newton steps of fluid DFT problem from Tramonto software package. ILU(0) preconditioning was used. The figure on the right shows the total matrix-vector product count for various recycle space dimensions Effectively, as we move to the right in this figure, we reach a point at which no recycled subspace information is ever discarded in the experiment. . . 119

4.9 The simultaneous solution of a base system and four shifted systems using the shifted GMRES method. The base system $\mathbf{A}$ is constructed from the matrix `conf5.0-00l4x4-1000` from Matrix Market [1], which we call $\mathbf{D}$. We form $\mathbf{A} = \mathbf{D} - (1/\kappa_c + .001)\mathbf{I}$ for a value κ_c provided with the matrix. The matrix $\mathbf{A}$ is real-positive. GMRES(30) is used to solve the base system, and the shifted GMRES method produces approximations for the four shifted systems. The shifts $\{\sigma_i\}_{i=1}^4$ are listed in the convergence plot. 125

4.10 The performance of Algorithm 4.4.1 for a collection of shifts taken from the interval $[10^{-8}, 10^8]$. For this figure, both axes are on a logarithmic scale. We compute the largest principle angle between $\mathcal{R}(\mathbf{C})$ and $\mathcal{R}((\mathbf{A}+\sigma\mathbf{I})\mathbf{U}^{(\sigma)})$ to examine the accuracy. 137

4.11 The performance of Algorithm 4.4.1 for collections of shifts. Different lines depict the algorithm's performance for different numbers of shifts. Line darkness increases when more shifts are used (meaning the algorithm requires more iterations). We compute the largest principle angle between $\mathcal{R}(\mathbf{C})$ and $\mathcal{R}((\mathbf{A}+\sigma\mathbf{I})\mathbf{U}^{(\sigma)})$ to examine the accuracy. In the figures in the left column, shifts come from the interval $[10-\gamma, 10+\gamma]$. In the right column, shifts come from the interval $[10^5-\gamma, 10^5+\gamma]$. Observe that the Algorithm 4.4.1 performs well for larger values of γ when the center is located at 10^5. 138

4.12 The performance of GCRODR for shifted systems. We performed simple tests on a sequence of four 1000×1000 bidiagonal matrices. The first matrix is the bidiagonal matrix used in [19]. The other systems are constructed by applying $\mathcal{O}(1)$ bidiagonal random perturbations to the first system using the `sprand()` Matlab function. The four shifts are 10^{-2}, 10^{-1}, 1, and 10. For the shifted systems, the residuals are only computed at the end of each cycle, with residual norms represented by the **circle**, **triangle**, **square**, and **cross**, respectively. The curves originating from these symbols are the convergence curves for the GCRODR iterations executed for each shift system in serial at the end of the process. 149

4.13 An experiment for the same sequence of systems and shifts as in Figure 4.12. Here, after the base system converges, we choose a new base system and apply the algorithm recursively on a smaller family of systems. We observe that this strategy does yield an 11 iteration improvement. 150

4.14 Convergence curves when we apply GCRODR to each shifted bidiagonal system in serial. 151

4.15 The performance of GCRODR for shifted systems on a sequence of seven small Wilson fermion matrices where for each matrix, we solve a linear system with the base system and ones associated to the shifts, .001 , .002, .003, $-.6$, and $-.5$. In the left figure, we demonstrate the performance of GCRODR(50,50) for shifted systems as compared to repeated applications of GCRODR(50,50). In the figure on the right, we compare performance for different size recycled subspaces. 152

4.16 Convergence curves from the same experiment as in Figure 4.15. Observe that two of the shifted systems (corresponding to the negative shifts) require more work than the others for each base system **including** the first system, in which we started with no recycled subspace. These are the two shifts for which shifted GMRES (or shifted GMRES-DR) would fail. Notice that in this case, we are still able to converge by applying GCRODR at the end. 154

LIST OF TABLES

Table Page

3.1 MATLAB's implementation of GMRES, PGMRES, van Gijzen's implementation of IDR(2), and the Schur Complement Method are applied to the discretized Lippmann–Schwinger integral operator for various wave numbers, using a discretization of $n = 10^3$ points and a right-hand side vector randomly generated by MATLAB's `randn` function. The timings are in seconds; a $*$ indicates that the method did not reduce the relative residual norm to 10^{-10} within 1000 iterations. Experiments were run on `Macbooki5`. 72

3.2 Run time and matrix-vector product count for the parallelized Schur Complement Method, IDR(2) and GMRES(11), and GMRES(100) on the two-dimensional Bratu problem from [8], but discretized on a grid of 500×500 points. The convergence tolerance of the relative residual, $\|\mathbf{b} - \mathbf{A}\widetilde{\mathbf{x}}\|$, was 10^{-7}. All methods were preconditioned with an incomplete Cholesky factorization of the Hermitian part with drop tolerance $10^{-2.}$. For the Schur Complement method, there were three systems to solve, and they were done in parallel on three processors. 74

LIST OF ALGORITHMS

Algorithm Page

2.3.1 Modified Gram-Schmidt Arnoldi Procedure 23

2.3.2 The Full Orthogonalization Method 26

2.3.3 The GMRES Method 30

2.3.4 Hermitian Lanczos Algorithm 32

3.3.1 The Schur Complement Method 64

3.4.1 Ruhe's Block Arnoldi Method 77

3.4.2 Block MINRES (Ruhe Implementation) 92

4.3.1 A step of the block Arnoldi algorithm 102

4.3.2 The Block GCRODR Algorithm 108

4.4.1 Computing a Family of Recycled Spaces for Multiple Shifted Linear Systems 136

4.4.2 Schematic of Shifted GCRODR with an Approximate Collinearity Condition 141

CHAPTER 1
INTRODUCTION

This thesis concerns methods for solving a linear system or a sequence of systems all of the form

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{1.1}$$

where $\mathbf{A} \in \mathbb{C}^{n\times n}$ is large and $\mathbf{A}$ is *sparse*, meaning only a small percentage of its entries are nonzero. This is an important class of problems arising in a wide variety of applied mathematics applications including the numerical solution of partial differential equations (PDE), see, e.g., [9], Newton iterations for nonlinear problems [20], and optimization problems, see, e.g., [52]. In Chapter 2, we present a review of Krylov subspace methods, providing context for the work presented in the later chapters.

The work in Chapter 3 concerns short-term recurrence Krylov subspace methods for nearly-Hermitian linear systems. This chapter is based on work done in collaboration with Mark Embree of Rice University, Josef Sifuentes of New York University's Courant Institute, and Fei Xue of Temple University. We had many fruitful discussions which improved the quality of the analysis.

A coefficient matrix is called nearly-Hermitian when it can be split into the sum of a Hermitian matrix and a low-rank skew-Hermitian matrix. In 2008, Beckermann and Reichel [8] introduced the Progressive GMRES algorithm, a residual minimizing short-term recurrence Krylov subspace method for solving such systems. We present numerical experiments showing that this algorithm is unstable for both simple contrived systems and some real world examples. Our analysis identifies some causes of this instability. We show that the underlying short-term recurrence actually produces dependent basis vectors, and this deficiency has previously been shown to cause stagnation in GMRES algorithms. As a work-around, we introduce a different short-term recurrence method for the same class of matrices using Lanczos-based iterations involving the Hermitian part of the system. We call this the Schur complement method, and it can be used either as a solver or a preconditioner. We present convergence bounds for this new algorithm and demonstrate its competitiveness against existing fixed-storage methods. The Schur complement method also has the added benefit of being amenable to parallelization.

The work Chapter 4 involves extending a strategy called Krylov subspace recycling, introduced by Parks and colleagues in 2005 [68]. This method requires more overhead than other subspace augmentation methods but offers the ability to recycle subspace information between cycles for a single linear system and recycle information between related linear systems.

In Section 4.3, we extend subspace recycling to the block Krylov subspace setting. This work was completed under the guidance of Michael Parks of Sandia National Laboratories in Albuquerque, NM.

A block Krylov subspace is a generalization of Krylov subspace where a single starting vector is replaced with a block of linearly independent starting vectors. This effectively generates a richer subspace with fewer applications of the operator at the expense of quicker filling of available memory. For solving linear systems, block Krylov subspaces can be used to solve systems with multiple right-hand sides. They can also be used to solve a system with one right-hand side through the introduction of random right-hand sides used solely to accelerate convergence since the richer information in the subspace may lead to faster convergence. This richer subspace is also attractive from the subspace recycling standpoint. If, for example, we recycle approximate subspace information, the block Krylov subspace may offer better eigenspace approximations. Therefore, we derive a version of this algorithm for use in the block Krylov setting for both systems with multiple physical right-hand sides and for acceleration of the solver through the introduction of random right-hand sides. We call this method block GCRODR (block GMRES with recycling). We discuss both theory and implementation details, and we have codes in both Matlab [55] and in the Trilinos C++ framework [2]. We then demonstrate this method's effectiveness as a solver embedded in a Newton iteration arising in fluid density functional theory.

In Section 4.4, we extend the methods of subspace recycling to a family of linear systems differing only by multiples of the identity. The work in this section has benefited from many fruitful discussions with Fei Xue of Temple University.

These problems arise in applications such as the theory of quantum chromodynamics, a theory of the behavior of subatomic particles. We wish to build on the class of Krylov methods which allow the simultaneous solution of all shifted linear

systems while generating only one subspace. However, the mechanics of subspace recycling complicates this situation and interferes with our ability to simultaneously solve all systems using these techniques. We discuss the difficulties of solving all shifted systems simultaneously while also employing a recycling scheme and fixing storage requirements. We show that, though we cannot directly solve all systems while satisfying these criteria, we can solve the base system and produce an improved approximation to the solution of the shifted systems at little additional cost. Then, once the primary system has converged, we solve the shifted systems to tolerance. The method is robust enough to be applied to sequences of systems where the base system changes slowly and the shifts differ for each base system. We have good preliminary results when solving a sequence of five families of shifted systems arising from quantum chromodynamics simulations.

CHAPTER 2
NOTATION AND BACKGROUND

2.1 Notation

We begin with some notation. Boldface upper-case letters will be used to denote matrices, including block vectors. Boldface lower-case letters will denote column vectors. In the case that we need to refer to a column of a matrix, we will denote this with the lower-case version of the letter used to denote the matrix, with a superscript number indicating to which column we refer. For example, $\mathbf{a}^{(i)}$ refers to the ith column of the matrix $\mathbf{A}$. As an exception to this rule, we will denote the ith Cartesian basis vector from $\mathbb{C}^j$ as $\mathbf{e}_j^{(i)}$. Let $(\cdot)_{1:m}$ denote a matrix or vector composed of the first m rows of the argument. We denote the Euclidean norm by $\|\cdot\|$ and the Frobenius norm by $\|\cdot\|_F$. Real constants will be denoted by lower-case Greek letters, and integers will be denoted by the letters i, j, k, ℓ, m, and n, or subscripted versions of them when needed. For a square, nonsingular matrix $\mathbf{A}$, we will denote the condition number associated to the 2-norm, $\kappa(\mathbf{A}) = \|\mathbf{A}\| \, \|\mathbf{A}^{-1}\|$. For a matrix $\mathbf{V}$ which is rectangular, we define $\kappa(\mathbf{V})$ as the ratio of the largest and smallest singular values. We denote the range of $\mathbf{A}$ by $\mathcal{R}(\mathbf{A})$. When identifying an equation as a

QR-factorization, we will use the convention that the right-hand side of the equation is the QR-factorization of the left-hand side of the equation. For example, if we identify $\mathbf{B} = \mathbf{GS}$ as a QR-factorization, then the matrix $\mathbf{G}$ has orthonormal columns and $\mathbf{S}$ is upper triangular. Similarly, if we identify an equation as an LU-factorization, then the right-hand side of the equation is the LU-factorization of the left-hand side. We use an upper-case calligraphic script letter to identify a linear operator independent of any matrix representation with respect to a particular basis.

2.2 Materials and Methods

In our work, we used different computers to execute performance testing experiments. Many experiments were done on a Macbook Pro with a 2.3 GHz Intel Core i5 processor and 8 GB of 1333 MHz DDR3 main memory. The operating system is Mac OS X version 10.7.3, and the Matlab version is R2011b 64-bit. We will denote this computer `Macbooki5`. Some parallel experiments were performed on an Oracle Sun Fire X4600 M2 x64 server with eight AMD dual core Opteron 64-bit processors and 128 gigabytes of RAM. The operating system is the 64-bit version of SuSE Linux 11. All computations were done in the 64-bit version of Matlab R2010a. We will denote this machine `Euler`.

At times, we need to generate matrices or vectors by applying random perturbations to an existing matrix or vector. In all our Matlab experiments using random perturbations, we use the `mt19937ar` random number generator and seed value zero.

2.3 Background

There are many direct methods which have been developed for solving linear systems such as (1.1). Direct methods involve obtaining a factorization of the coefficient matrix such that solving the linear systems involving the factors can be solved efficiently and stably. One such example is the LU-factorization. For a given matrix $\mathbf{A}$, this factorization encapsulates the process of Gaussian elimination as the product of two matrices $\mathbf{A} = \mathbf{LU}$ where $\mathbf{A}$ is transformed into the upper- triangular matrix $\mathbf{U}$ through a series of row operations. The matrix $\mathbf{L}$ is a lower-triangular matrix with unit diagonal elements, and elements of the lower-triangular $\mathbf{L}^{-1}$ are the coefficients from the row operations used to transform $\mathbf{A}$ into $\mathbf{U}$. We can then rewrite the equation (1.1) as

$$\mathbf{LUx} = \mathbf{b},$$

and this can be solved by first solving a linear system involving $\mathbf{L}$ and then solving a linear system involving $\mathbf{U}$. Since $\mathbf{L}$ is lower-triangular and $\mathbf{U}$ is upper-triangular, these systems can be solved efficiently and stably by forward and back substitution, respectively. See, e.g., [94, Chapters 20-22], for details on the computation of the LU-factorization as well as practical implementation issues. The LU- factorization is just one example of a matrix factorization important in numerical linear algebra. There are many others, such as the QR-factorization and the singular value decomposition.

For any square matrix these factorizations exist, in theory (where we may need to permute the rows of $\mathbf{A}$ to obtain an LU-factorization). However, in practice, their computation becomes impractical for matrices of large enough size. For example, the LU-factorization requires $\mathcal{O}(n^3)$ operations for a dense matrix. For a large enough dimension n, the factorization becomes prohibitively expensive to compute. If a matrix is sparse, there is highly developed technology for obtaining a sparse LU-factorization of a sparse matrix at a cost of $\mathcal{O}(\ell)$ where ℓ is the number of non-zeros in the matrix [22, 35, 70]. This is achieved by computing a permutation of the rows and columns resulting in a matrix which can be factored for less expense while maintaining sparsity of the factors. These methods, though, do not scale well for large systems, e.g., ones arising from fully 3-dimensional problems. However, sometimes computing this permutation has significant cost. Furthermore, in some cases, we may not even be able to hold the entire matrix in main memory, or we may only possess a procedure which performs a matrix-vector product. Therefore, for large, sparse linear systems, we may need to employ iterative methods which produce a series of approximations that will converge to the desired solution. Furthermore, methods which are matrix-free are desirable, where we say a method is matrix-free if it requires only the matrix-vector product. In many cases, such methods can have much better scaling properties.

Krylov subspace methods are a class of iterative methods which have been widely and successfully used in the solution of large, sparse linear systems. These methods do not require the full matrix to perform computations. They only require that we have a procedure which performs the matrix-vector product. For a coefficient matrix $\mathbf{A}$ and a starting vector $\mathbf{u}$, the jth Krylov subspace is defined as

$$\mathcal{K}_j(\mathbf{A},\mathbf{u}) = \operatorname{span}\left\{\mathbf{u}, \mathbf{A}\mathbf{u}, \mathbf{A}^2\mathbf{u}, \ldots, \mathbf{A}^{j-1}\mathbf{u}\right\}, \tag{2.1}$$

i.e., an element $\mathbf{w} \in \mathcal{K}_j(\mathbf{A},\mathbf{u})$ has the form $\mathbf{w} = p(\mathbf{A})\mathbf{u}$ where $p(x)$ is a polynomial of degree less than j. In a prototypical Krylov subspace method for solving

(1.1), we begin with an initial approximation to the solution $\mathbf{x}_0$ and initial residual $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$. At the jth iteration, we construct an approximation $\mathbf{x}_j = \mathbf{x}_0 + \mathbf{t}_j$. We choose $\mathbf{t}_j \in \mathcal{K}_j(\mathbf{A}, \mathbf{r}_0)$ such that the jth residual $\mathbf{r}_j = \mathbf{b} - \mathbf{A}\mathbf{x}_j$ satisfies some constraint. For example, in the full orthogonalization method (FOM) and conjugate gradient method (CG) [45], we choose $\mathbf{t}_j$ such that $\mathbf{r}_j \perp \mathcal{K}(\mathbf{A}, \mathbf{r}_0)$, and in minimum residual methods, such as the generalized minimum residual method (GMRES) of Saad and Schultz [76], Generalized Conjugate Residual (GCR) of Elman, Eisenstat, and Schultz [24], and MINRES of Paige and Saunders [64] we choose $\mathbf{t}_j$ such that $\mathbf{r}_j \perp \mathbf{A}\mathcal{K}(\mathbf{A}, \mathbf{r}_0)$. These types of conditions on the residual are called *Petrov-Galerkin* conditions, and they belong to a wider class of *residual projection methods.*

We emphasize here the difference between the imposition of a Petrov-Galerkin condition and an actual implementation of an algorithm. GMRES, GCR, and MINRES are all implementations which impose the condition that $\mathbf{r}_j \perp \mathbf{A}\mathcal{K}(\mathbf{A}, \mathbf{r}_0)$. However, MINRES only works for Hermitian matrices. GMRES and GCR are equivalent in exact arithmetic, but these two implementations can produce different sequences of iterations in practice. In the general case, where the dimension of $\mathcal{K}_j(\mathbf{A}, \mathbf{r}_0)$ is j for all $j \leq n$, observe that since we force the residual at each step to be orthogonal to subspaces which grow by one dimension at each iteration, at iteration $j = n$, the residual is orthogonal to $\mathbb{C}^n$, i.e., $\mathbf{r}_j = \mathbf{0}$, and we will have converged. However, we wish to solve linear systems with large, sparse coefficient matrices. Therefore, we do not want to perform n iterations of a Krylov methods to achieve convergence.

To understand the motivation for selecting iterates from a Krylov subspace we must discuss some properties of the coefficient matrix $\mathbf{A}$. First, observe that for a given initial approximation $\mathbf{x}_0$, there is some correction $\mathbf{t}$ such that we can rewrite (1.1) as

$$\begin{aligned} A(\mathbf{x}_0 + \mathbf{t}) &= \mathbf{b} \\ \mathbf{At} &= \mathbf{b} - \mathbf{Ax}_0 \\ \mathbf{At} &= \mathbf{r}_0. \end{aligned} \tag{2.2}$$

This means that, given an initial approximation, we can restate the problem (1.1) with initial approximation $\mathbf{x}_0$ as one in which we solve (2.2) for $\mathbf{t}$ where our initial approximation is $\mathbf{t}_0 = \mathbf{0}$. Therefore, without loss of generality, we simply can consider the case where our initial approximation $\mathbf{x}_0 = \mathbf{0}$.

At the jth step of our prototypical Krylov subspace method, we construct $\mathbf{x}_j \in \mathcal{K}_j(\mathbf{A}, \mathbf{b})$ which means $\mathbf{x}_j = p_j(\mathbf{A})\mathbf{b}$ where $p_j(x)$ is a polynomial of degree less than j. It is a well-known consequence of the Caley-Hamilton theorem that any matrix $\mathbf{A}^{-1}$ can be written as a polynomial of $\mathbf{A}$. Rather than appealing to the Cayley-Hamilton theorem, we offer an alternate proof of this fact that puts Krylov subspace methods into better context. This proof is inspired by the proof of Theorem 2.5 in [48], and it is constructive, i.e., we will construct this polynomial explicitly. If the matrix is diagonalizable, the proof is quite simple, relying on Lagrange interpolation. However, if we wish to prove this fact for defective matrices, we must work harder.

Before we state the theorem and proof, need some intermediate lemmas. With these lemmas, we will show that for an elementary Jordan matrix, we can construct a polynomial mapping the Jordan matrix to its inverse. We present a quartet of intermediate lemmas which build to proving this fact.

Lemma 2.1: Let $\mathcal{T}_i : \mathcal{C}^\infty(\mathbb{C}) \to \mathcal{C}^\infty(\mathbb{C})$ be the linear operator defined (for $i \geq 1$) by the operation

$$\mathcal{T}_i = \left(\frac{x}{i!} \frac{\mathrm{d}^i}{\mathrm{d}x} + \frac{1}{(i-1)!} \frac{\mathrm{d}^{(i-1)}}{\mathrm{d}x} \right), \tag{2.3}$$

then we have that

$$\mathcal{T}_i x^k = \frac{\mathrm{d}^i}{\mathrm{d}x} \frac{x^{k+1}}{i!}. \tag{2.4}$$

Proof. Observe that

$$\begin{aligned} \frac{\mathrm{d}^i}{\mathrm{d}x} x^k &= \left(\prod_{\ell=0}^{i-1} (k-\ell) \right) x^{(k-i)} \\ \frac{\mathrm{d}^{i-1}}{\mathrm{d}x} x^k &= \left(\prod_{\ell=0}^{i-2} (k-\ell) \right) x^{(k-(i-1))}. \end{aligned}$$

Now we can compute

$$\begin{aligned} \mathcal{T}_i x^k &= x \frac{\left(\prod_{\ell=0}^{i-1} (k-\ell) \right) x^{(k-i)}}{i!} + \frac{\left(\prod_{\ell=0}^{i-2} (k-\ell) \right) x^{(k-(i-1))}}{(i-1)!} \\ &= \frac{\left(\prod_{\ell=0}^{i-1} (k-\ell) \right) x^{(k-(i-1))}}{i!} + \frac{\left(\prod_{\ell=0}^{i-2} (k-\ell) \right) x^{(k-(i-1))}}{(i-1)!} \\ &= \frac{[k-(i-1)] \left(\prod_{\ell=0}^{i-2} (k-\ell) \right) x^{(k-(i-1))}}{i!} + \frac{\left(\prod_{\ell=0}^{i-2} (k-\ell) \right) x^{(k-(i-1))}}{(i-1)!} \\ &= \frac{[k-(i-1)] \left(\prod_{\ell=0}^{i-2} (k-\ell) \right) x^{(k-(i-1))}}{i!} + \frac{i \left(\prod_{\ell=0}^{i-2} (k-\ell) \right) x^{(k-(i-1))}}{i!} \\ &= \frac{[k+1] \left(\prod_{\ell=0}^{i-2} (k-\ell) \right) x^{(k-(i-1))}}{i!} \\ &= \frac{\mathrm{d}^i}{\mathrm{d}x} \frac{x^{k+1}}{i!}. \quad \square \end{aligned}$$

To be able to apply a polynomial to a Jordan matrix, we must be able to describe the structure of an arbitrary polynomial of a Jordan matrix. Fortunately, this can be done by describing the powers of a Jordan matrix.

Lemma 2.2: Let $p(x)$ be a polynomial of degree $d \geq 0$. Let $\mathbf{J}_\ell \in \mathbb{C}^{\ell \times \ell}$ be the Jordan matrix with eigenvalue λ,

$$\mathbf{J}_\ell = \begin{bmatrix} \lambda & 1 & 0 & \cdots & 0 \\ 0 & \lambda & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \lambda & 1 \\ 0 & 0 & 0 & 0 & \lambda \end{bmatrix}. \tag{2.5}$$

Then we have for any upper triangular entry of $(p(\mathbf{J}_\ell))$ the formula

$$(p(\mathbf{J}_\ell))_{(j_1,j_2)} = \frac{p^{(j_1-j_2)}(\lambda)}{(j_1-j_2)!}.$$

which gives us the full matrix structure,

$$p(\mathbf{J}_\ell) = \begin{bmatrix} p(\lambda) & p'(\lambda) & \frac{p''(\lambda)}{2} & \cdots & \frac{p^{(n-2)}(\lambda)}{(n-2)!} & \frac{p^{(n-1)}(\lambda)}{(n-1)!} \\ 0 & p(\lambda) & p'(\lambda) & \cdots & \frac{p^{(n-3)}(\lambda)}{(n-3)!} & \frac{p^{(n-2)}(\lambda)}{(n-2)!} \\ 0 & 0 & p(\lambda) & \cdots & \frac{p^{(n-4)}(\lambda)}{(n-4)!} & \frac{p^{(n-3)}(\lambda)}{(n-3)!} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & p(\lambda) & p'(\lambda) \\ 0 & 0 & 0 & \cdots & 0 & p(\lambda) \end{bmatrix}.$$

Proof. It suffices to prove this fact for a basis of the space of polynomials. Therefore, we will show this fact to be true for the canonical polynomial basis $\{1, x, x^2, \ldots\}$. We will prove by induction on the degree j of the basis polynomial. Let $p_j(x) = x^j$. This lemma is trivially true for $p_0(\mathbf{J}_\ell) = \mathbf{I}$. Now, suppose the lemma is true for

$j = 1 \dots k-1$. We can write $p_k(\mathbf{J}_\ell) = \mathbf{J}_\ell p_{k-1}(\mathbf{J}_\ell)$ and

$$\mathbf{J}_\ell p_{k-1}(\mathbf{J}_\ell) = \begin{bmatrix} \lambda & 1 & 0 & \cdots & 0 \\ 0 & \lambda & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \lambda & 1 \\ 0 & 0 & 0 & 0 & \lambda \end{bmatrix} \begin{bmatrix} p_{k-1}(\lambda) & p'_{k-1}(\lambda) & \frac{p''_{k-1}(\lambda)}{2} & \cdots & \frac{p_{k-1}^{(\ell-2)}(\lambda)}{(\ell-2)!} & \frac{p_{k-1}^{(\ell-1)}(\lambda)}{(\ell-1)!} \\ 0 & p_{k-1}(\lambda) & p'_{k-1}(\lambda) & \cdots & \frac{p_{k-1}^{(\ell-3)}(\lambda)}{(\ell-3)!} & \frac{p_{k-1}^{(\ell-2)}(\lambda)}{(\ell-2)!} \\ 0 & 0 & p_{k-1}(\lambda) & \cdots & \frac{p_{k-1}^{(\ell-4)}(\lambda)}{(\ell-4)!} & \frac{p_{k-1}^{(\ell-3)}(\lambda)}{(\ell-3)!} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & p_{k-1}(\lambda) & p'_{k-1}(\lambda) \\ 0 & 0 & 0 & \cdots & 0 & p_{k-1}(\lambda) \end{bmatrix}$$

This matrix is upper triangular, since it is the product of two upper triangular matrices. There are three possible cases for nonzero entries of $\mathbf{J}_\ell p_{k-1}(\mathbf{J}_\ell)$.

1. By straightforward computation, we see that all diagonal entries of $\mathbf{J}_\ell p_{k-1}(\mathbf{J}_\ell)$ are of the form $\lambda p_{k-1}(\lambda)$ which is $p_k(\lambda)$.
2. Suppose $p_{k-1}(\mathbf{J}_\ell)$ is a truly banded, upper triangular matrix, such that the jth superdiagonal is the last nonzero diagonal of the matrix. All entries of this superdiagonal are the value $\frac{p^{(j)}(\lambda)}{j!}$. Observe that in this case, after multiplication on the left by $\mathbf{J}_\ell$, the entries of the $(j+1)$st superdiagonal of $\mathbf{J}_\ell p_{k-1}(\mathbf{J}_\ell)$ will be copies of the entries of the jth subdiagonal of $p_{k-1}(\mathbf{J}_\ell)$.
3. For any upper triangular entry $(\mathbf{J}_\ell p_{k-1}(\mathbf{J}_\ell))_{(j_1,j_2)}$ not discussed in cases 1 or 2, we can explicitly compute its value by applying Lemma 2.1. Studying the *real* dot product between the row j_1 of $\mathbf{J}_\ell$ and column j_2 of $p_{k-1}(\mathbf{J}_\ell)$, we see that $(\mathbf{J}_\ell p_{k-1}(\mathbf{J}_\ell))_{(j_1,j_2)} = \lambda p_{k-1}^{(j_1-j_2)}(\lambda) + p_{k-1}^{(j_1-j_2-1)}(\lambda)$. However, since $p_{k-1}(x) = x^{k-1}$, by Lemma 2.1, this is equivalent to

$$(\mathbf{J}_\ell p_{k-1}(\mathbf{J}_\ell))_{(j_1,j_2)} = \left(\mathcal{T}_{(j_1-j_2)} x^{k-1}\right)_{x=\lambda} = \left(\frac{\mathrm{d}^{(j_1-j_2)}}{\mathrm{d}x} \frac{x^k}{(j_1-j_2)!}\right)_{x=\lambda}.$$

Thus, the lemma is proven for the canonical basis of polynomials, and it can be applied to arbitrary polynomials in $\mathbf{J}_\ell$. □

Next we need to construct the inverse of a Jordan matrix of arbitrary size. Due to its special structure, this turns out to be relatively straightforward.

Lemma 2.3: Let $\mathbf{J}_\ell \in \mathbb{C}^{\ell\times\ell}$ be a Jordan matrix as previously defined. Then $\mathbf{J}_\ell$ has the inverse

$$\mathbf{J}_\ell^{-1} = \begin{bmatrix} \frac{1}{\lambda} & \frac{-1}{\lambda^2} & \frac{1}{\lambda^3} & \cdots & \frac{(-1)^{\ell+1}}{\lambda^\ell} \\ 0 & \frac{1}{\lambda} & \frac{-1}{\lambda^2} & \cdots & \frac{(-1)^\ell}{\lambda^{\ell-1}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \frac{1}{\lambda} & \frac{-1}{\lambda^2} \\ 0 & 0 & 0 & 0 & \frac{1}{\lambda} \end{bmatrix} \tag{2.6}$$

Proof. This can be proved by inducting on the matrix dimension j. We can easily invert the scalar matrix $\mathbf{J}_1 = \lambda$ with $\mathbf{J}_1^{-1} = \frac{1}{\lambda}$, satisfying the lemma for $j = 1$. Now, we assume that the lemma is true for $j = 1 \ldots \ell - 1$. Observe that $\mathbf{J}_{\ell-1}$ is contained in the upper left-hand corner of $\mathbf{J}_\ell$, i.e.,

$$\mathbf{J}_\ell = \begin{bmatrix} \mathbf{J}_{\ell-1} & \mathbf{e} \\ \mathbf{0} & \lambda \end{bmatrix},$$

where $\mathbf{e} = \mathbf{e}_{\ell-1}^{(\ell-1)}$. If we let let $\mathbf{h}_\lambda = \begin{bmatrix} 0 & \cdots & 0 & 1 & \lambda - 1 \end{bmatrix}^T \in \mathbb{C}^\ell$, then we can write $\mathbf{J}_\ell$ as the rank one modification of an easily invertible matrix,

$$\mathbf{J}_\ell = \begin{bmatrix} \mathbf{J}_{\ell-1} & \\ & 1 \end{bmatrix} + \mathbf{h}_\lambda \mathbf{e}^*. \tag{2.7}$$

We can now invert $\mathbf{J}_\ell$ by applying the Sherman-Morrison-Woodbury identity [77, 101] for inverting low-rank modifications of matrices with known inverses. We have

$$\mathbf{J}_\ell^{-1} = \begin{bmatrix} \mathbf{J}_{\ell-1}^{-1} & \\ & 1 \end{bmatrix} - \frac{\begin{bmatrix} \mathbf{J}_{\ell-1}^{-1} & \\ & 1 \end{bmatrix} \mathbf{h}_\lambda \mathbf{e}^* \begin{bmatrix} \mathbf{J}_{\ell-1}^{-1} & \\ & 1 \end{bmatrix}}{1 + \mathbf{e}^* \begin{bmatrix} \mathbf{J}_{\ell-1}^{-1} & \\ & 1 \end{bmatrix} \mathbf{h}_\lambda}. \tag{2.8}$$

Now notice that we can simplify (2.8), since

$$\begin{bmatrix} \mathbf{J}_{\ell-1}^{-1} & \\ & 1 \end{bmatrix} \mathbf{h}_\lambda = \begin{bmatrix} \frac{(-1)^\ell}{\lambda^{\ell-1}} & \\ \frac{(-1)^{\ell-1}}{\lambda^{\ell-2}} & \vdots \\ \frac{-1}{\lambda^2} & \\ \frac{1}{\lambda} & \\ 0 & \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} (\lambda - 1) = \begin{bmatrix} \frac{(-1)^\ell}{\lambda^{\ell-1}} \\ \frac{(-1)^{\ell-1}}{\lambda^{\ell-2}} \\ \vdots \\ \frac{-1}{\lambda^2} \\ \frac{1}{\lambda} \\ \lambda - 1 \end{bmatrix} \quad \text{and} \quad \mathbf{e}^* \begin{bmatrix} \mathbf{J}_{\ell-1}^{-1} & \\ & 1 \end{bmatrix} = \mathbf{e}^*.$$

Now we can rewrite (2.8),

$$\mathbf{J}_\ell^{-1} = \begin{bmatrix} \mathbf{J}_{\ell-1}^{-1} & \\ & 1 \end{bmatrix} - \frac{1}{\lambda} \begin{bmatrix} \frac{(-1)^\ell}{\lambda^{\ell-1}} \\ \frac{(-1)^{\ell-1}}{\lambda^{\ell-2}} \\ \vdots \\ \frac{-1}{\lambda^2} \\ \frac{1}{\lambda} \\ \lambda - 1 \end{bmatrix} \mathbf{e}^* = \begin{bmatrix} \mathbf{J}_{\ell-1}^{-1} & \\ & 1 \end{bmatrix} + \begin{bmatrix} \frac{(-1)^{\ell+1}}{\lambda^\ell} \\ \frac{(-1)^\ell}{\lambda^{\ell-1}} \\ \vdots \\ \frac{1}{\lambda^3} \\ \frac{-1}{\lambda^2} \\ \frac{1-\lambda}{\lambda} \end{bmatrix} \mathbf{e}^*.$$

This last step has the effect of adding the column vector $\begin{bmatrix} \frac{(-1)^{\ell+1}}{\lambda^\ell} & \frac{(-1)^\ell}{\lambda^{\ell-1}} & \cdots & \frac{1}{\lambda^3} & \frac{-1}{\lambda^2} & \frac{1-\lambda}{\lambda} \end{bmatrix}^T$ to the last column of $\begin{bmatrix} \mathbf{J}_{\ell-1}^{-1} & \\ & 1 \end{bmatrix}$ which yields the identity (2.6). □

The final lemma we need states that the inverse of a Jordan matrix can be written as a polynomial in the matrix. This proof is constructive, using Hermite interpolation polynomials.

Lemma 2.4: Let $\mathbf{J} \in \mathbb{C}^{\ell \times \ell}$ be the Jordan matrix with eigenvalue λ defined as in (2.5). Then there exists a polynomial $p(x)$ of degree ℓ such that $p(\mathbf{J}) = \mathbf{J}^{-1}$.

Proof. Let $p(x)$ be the Hermite interpolation polynomial such that

$$p^{(i)}(\lambda) = \frac{(-1)^i}{\lambda^{i+1}} i!,$$

for $i = 0 \ldots \ell - 1$. By applying Lemmas 2.2 and 2.3, we see that $p(\mathbf{J}) = \mathbf{J}^{-1}$. □

We now have the tools we need to prove our main theorem.

Theorem 2.5: Let $\mathbf{A} \in \mathbb{C}^{n \times n}$ be a nonsingular matrix, possibly with eigenvalues of multiplicity greater than one. Then $\mathbf{A}^{-1} = p(\mathbf{A})$ for some polynomial $p(x)$ with degree less than or equal to n. Furthermore, the polynomial $r(x) = 1 - xp(x)$ has roots at all the eigenvalues of $\mathbf{A}$.

Proof. Let $\mathbf{A} = \mathbf{P}^{-1}\mathbf{J}\mathbf{P}$ be the Jordan decomposition of $\mathbf{A}$ where $\mathbf{P}$ is the invertible matrix with the Jordan vectors of $\mathbf{A}$ as columns and $\mathbf{J} = \mathrm{diag}(\mathbf{J}_1, \mathbf{J}_2, \ldots, \mathbf{J}_k)$ is the block diagonal matrix with Jordan block $\mathbf{J}_i$ corresponding to distinct eigenvalue λ_i for $1 \leq i \leq k$. Let $p(x)$ be the unique Hermite interpolation polynomial defined as follows. For every nondefective eigenvalue λ_{j_1}, define $p(\lambda_{j_1}) = 1/\lambda_{j_1}$. For a defective eigenvalue λ_{j_2}, define $p^{(i)}(\lambda_{j_2}) = \frac{(-1)^i i!}{\lambda_{j_2}^{i+1}}$ for $0 \leq i \leq \ell_{j_2} - 1$ where ℓ_{j_2} is the algebraic multiplicity of λ_{j_2}. It is clear from Lemma 2.4 that this construction yields a polynomial such that $p(\mathbf{J}_{j_2}) = \mathbf{J}_{j_2}^{-1}$. Thus, we have

$$\begin{aligned} p(\mathbf{A}) &= p(\mathbf{P}^{-1}\mathbf{J}\mathbf{P}) \\ &= \mathbf{P}^{-1}p(\mathbf{J})\mathbf{P} \\ &= \mathbf{P}^{-1}\mathbf{J}^{-1}\mathbf{P} \qquad (2.9) \\ &= \mathbf{A}^{-1} \end{aligned}$$

where (2.9) follows from the fact that by the construction of $p(x)$, we have

$$\begin{aligned} p(\mathbf{J}) &= \mathrm{diag}\left\{p(\mathbf{J}_1), p(\mathbf{J}_2), \ldots, p(\mathbf{J}_\ell)\right\} \\ &= \mathrm{diag}\left\{\mathbf{J}_1^{-1}, \mathbf{J}_2^{-1}, \ldots, \mathbf{J}_\ell^{-1}\right\}. \end{aligned}$$

Furthermore, it is clear that the roots of $r(x) = 1 - xp(x)$ are the eigenvalues of $\mathbf{A}$. $\square$

This proof illuminates the connection between Krylov subspace methods and polynomial interpolation. At iteration j of the prototypical Krylov subspace method, we produce an approximation of the form $\mathbf{x}_j = p_j(\mathbf{A})\mathbf{b}$. An effective Krylov subspace method selects $\mathbf{x}_j$ such that $p_j(\mathbf{A})$ is a good low-degree polynomial approximation to $p(\mathbf{A}) = \mathbf{A}^{-1}$. Consider the situation in which at step j of our method we compute j eigenvalue approximations for $\mathbf{A}$ using Krylov subspace information. Denote these approximations $\{\theta_i\}_{i=1}^{j}$. Then, we can consider constructing $p_j(x)$ such that $p_j(\theta_i) = 1/\theta_i$ for all $i \leq j$. Equivalently, we can construct the degree j polynomial

$$r_j(x) = 1 - xp_j(x) \tag{2.10}$$

as the polynomial satisfying $r_j(\theta_i) = 0$ for all $i \leq j$. At step j, the residual $\mathbf{r}_j = \mathbf{b} - \mathbf{A}\mathbf{x}_j = r_j(\mathbf{A})\mathbf{b}$. We would like to achieve large reduction of the residual norm $\|\mathbf{r}_j\|$ for $j \ll n$. Observe that

$$\begin{aligned} \|\mathbf{r}_j\| &= \|r_j(\mathbf{A})\mathbf{b}\| \\ &\leq \|r_j(\mathbf{A})\| \, \|\mathbf{b}\| \\ &\leq \|\mathbf{X}\| \left\|\mathbf{X}^{-1}\right\| \|r_j(\mathbf{D})\| \, \|\mathbf{b}\| \\ &\leq \kappa(\mathbf{X}) \max_{1\leq i\leq n} |r_j(\lambda_i)| \, \|\mathbf{b}\| \, . \end{aligned} \tag{2.11}$$

This reveals the deep connection between Krylov subspace methods, polynomial interpolation, and eigenvalue approximation. Observe that this inequality indicates that the convergence of a Krylov subspace method is effected by how well the roots of the residual polynomial approximate eigenvalues of the matrix. This is not surprising. More interesting, though, is the fact that the condition number of the eigenvector matrix can also effect convergence. This indicates that there are limits to the polynomial interpolation analogy when discussing Krylov subspace methods. In fact, for one method (GMRES) it has been shown that for a given set of eigenvalues and any monotonically nonincreasing set of residual norms, one can construct a non-Hermitian matrix with those eigenvalues for which GMRES produces that sequence of residual norms [39]. Though eigenvalues are important for convergence, they by no means tell the whole story for non-Hermitian systems.

For Hermitian systems, though, we are more fortunate. Recall that the eigenvectors of the Hermitian matrix are mutually orthogonal. Thus, $\mathbf{X}$ is a unitary matrix, and $\kappa(\mathbf{X}) = 1$, and the bound on the residual norm (2.11) becomes

$$\left\|\mathbf{r}_j\right\| \leq \max_{1\leq i\leq n} \left|r_j(\lambda_i)\right| \left\|\mathbf{b}\right\| . \tag{2.12}$$

The convergence of a Krylov method applied to a Hermitian linear system is completely determined by its eigenvalues. In fact, there is a rich theory connecting Krylov methods for Hermitian linear systems to the theory of orthogonal polynomials and interpolation; see, e.g., [28], for further reading.

In practical applications, the Krylov basis

$$\left\{\mathbf{b}, \mathbf{A}\mathbf{b}, \mathbf{A}^2\mathbf{b}, \ldots, \mathbf{A}^{j-1}\mathbf{b}\right\} \tag{2.13}$$

of $\mathcal{K}_j(\mathbf{A}, \mathbf{b})$ is never generated; it is merely a theoretical tool. This basis quickly becomes highly dependent in the direction of a dominant eigenvector of $\mathbf{A}$. This is easily demonstrated. For illustration, we assume that $\mathbf{A}$ is diagonalizable and we also assume that $\mathbf{A}$ has a unique dominant eigenvector. For $1 \leq i \leq n$, let $(\mathbf{w}_i, \lambda_i)$

denote the ith eigenpair of $\mathbf{A}$, where the eigenvalues are indexed in descending order of their magnitude. Then, for any $\mathbf{u} \in \mathbb{C}^n$, we can write its eigendecomposition

$$\mathbf{u} = \sum_{i=1}^{n} a_i \mathbf{w}_i.$$

We have that

$$\mathbf{A}^k \mathbf{u} = \sum_{i=1}^{n} a_i \lambda_i^k \mathbf{w}_i,$$

and if we divide both sides by λ_1^k, then we get

$$\frac{1}{\lambda_1^k} \mathbf{A}^k \mathbf{u} = a_1 \mathbf{w}_1 + \sum_{i=2}^{n} a_i \left(\frac{\lambda_i}{\lambda_1} \right)^k \mathbf{w}_i.$$

Since, λ_1 is the unique eigenvalue with largest magnitude of $\mathbf{A}$, we have $\left(\frac{\lambda_i}{\lambda_1}\right)^k \to 0$ as $k \to \infty$ for all $i > 1$. Thus, we see that computing powers of $\mathbf{A}$ to construct the Krylov basis will yield vectors increasingly pointing in the direction of the dominant eigenvector. This is the underlying principle of eigenvector computation methods such as the power method and the inverse power method, which use this convergence to obtain eigenvector approximations. However, in this case, the convergence yields a nearly-dependent basis. In exact arithmetic, this would not be problematic. However, numerically, this will result in dependence of the Krylov subspace basis which has been shown to cause convergence failure, see, e.g., [40]. Under these circumstances, the optimal basis we can maintain is an orthonormal basis. In fact, it has been shown that Krylov subspace methods using non-optimal, i.e., non-orthonormal, bases exhibit delayed convergence [83].

How do we generate an orthonormal basis of the Krylov subspace $\mathcal{K}_j(\mathbf{A}, \mathbf{b})$? In exact arithmetic, we could construct the Krylov basis (2.13) and then orthogonalize the resulting vectors. However, numerically, this is an unstable process. As j increases, the Krylov basis becomes increasingly dependent, and we begin to lose subspace information due to floating-point errors. We demonstrate this in Figure 2.1 by constructing the Krylov basis of $\mathcal{K}_{50}(\mathbf{A}, \mathbf{b})$, orthogonalizing this basis, and measuring its accuracy. Here, accuracy is taken to mean the size of the largest principle angle, see e.g., [36, Section 12.4], between the subspace spanned by this basis and a different orthonormal basis for $\mathcal{K}_{50}(\mathbf{A}, \mathbf{b})$, computed using a much more accurate method (Householder Arnoldi [75, Section 6.3]). Fortunately, Arnoldi proposed a stable alternative in 1951 [5].

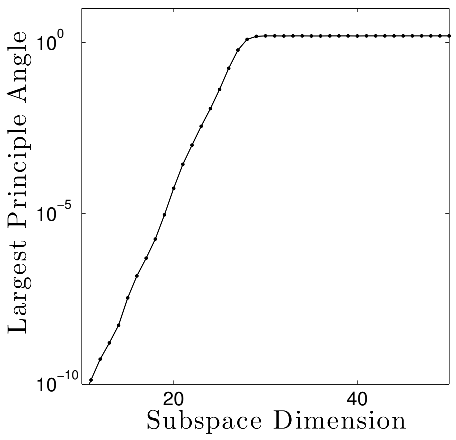


Figure 2.1. Computing the largest principal angle between the true Krylov subspace (as represented by a basis constructed using Householder Arnoldi) and the subspace spanned by the vectors generated by computing the Krylov basis and orthogonalizing. We use the `sherman5` matrix from Matrix Market [1] and the starting vector is the right-hand side is provided with the `sherman5` matrix.

In this discussion, let us assume that the dimension of $\mathcal{K}_j(\mathbf{A}, \mathbf{b})$ is j for all $1 \leq j \leq n$. For a given coefficient matrix $\mathbf{A}$ and a starting vector $\mathbf{b}$, the Arnoldi procedure is an iterative process used to generate orthonormal basis vectors of the Krylov space generated by $\mathbf{A}$ and $\mathbf{b}$. The method begins by setting $\mathbf{v}_1 = \mathbf{b}/\left\|\mathbf{b}\right\|$. At step j of the process, assume we have already constructed $\{\mathbf{v}_1, \ldots, \mathbf{v}_j\}$, an orthonormal basis for $\mathcal{K}_j(\mathbf{A}, \mathbf{b})$. We can collect the orthonormal Arnoldi basis vectors as columns of a matrix

$$\mathbf{V}_{j+1} = \begin{bmatrix} \mathbf{v}_1, & \mathbf{v}_2, & \ldots, & \mathbf{v}_{j+1} \end{bmatrix}.$$

We compute $\mathbf{w}_j = \mathbf{A}\mathbf{v}_j$. It can be shown that $\mathbf{w}_j \in \mathcal{K}_{j+1}(\mathbf{A}, \mathbf{b}) \setminus \mathcal{K}_j(\mathbf{A}, \mathbf{b})$. We must then orthogonalize $\mathbf{w}_j$ against $\mathcal{K}_j(\mathbf{A}, \mathbf{b})$. Numerically, orthogonalization via the classical Gram-Schmidt process will produce basis vectors which become less orthogonal at each iteration. Alternative procedures which are better able to produce orthonormal basis vectors have been proposed; see, e.g., the modified Gram-Schmidt process [94, Chapter 8], Householder orthogonalization [94, Chapter 10], and classical Gram-Schmidt with selective reorthogonalization [18]. One way to measure the quality of an orthogonalization is by computing the *orthogonality residual*, i.e., compute the size of $\left\|\mathbf{I} - \mathbf{V}_j^*\mathbf{V}_j\right\|$. Since $\mathbf{V}_j^*$ is the left inverse of $\mathbf{V}_j$, the orthogonality residual should be zero in exact arithmetic. In Figure 2.2, we present a comparison of orthogonality residuals of orthonormal bases for $\mathcal{K}_{50}(\mathbf{A}, \mathbf{b})$, for a particular matrix and starting vector, constructed using classical Gram-Schmidt, Modified Gram-Schmidt, and Householder orthogonalization.

We briefly describe orthogonalization via the modified Gram-Schmidt process. We wish to orthogonalize $\mathbf{w}_j$ with respect to $\mathcal{K}_j(\mathbf{A}, \mathbf{b})$. Let $\mathcal{P}_{\mathcal{K}_j}$ be the orthogonal

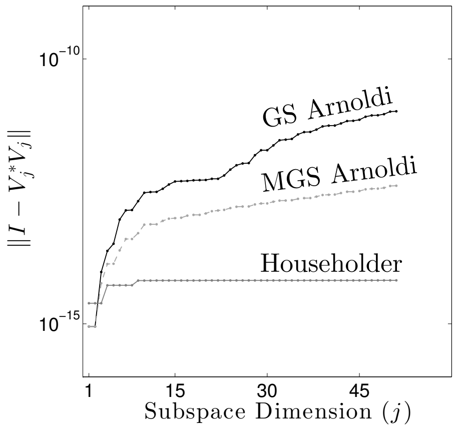


Figure 2.2. The orthogonality residual ($\left\|\mathbf{I} - \mathbf{V}_j^* \mathbf{V}_j\right\|$ at the jth iteration) of three sets of bases of a Krylov subspace as the dimension j increases. The three methods are based on the Arnoldi procedure, but differ in how the orthogonalization step is performed (Gram-Schmidt, Modified Gram- Schmidt, Householder). We use the `sherman5` matrix from Matrix Market [1] and the starting vector is the right-hand side is provided with the `sherman5` matrix.

projector onto $\mathcal{K}_j(\mathbf{A}, \mathbf{b})$. In the classical Gram-Schmidt process, we would numerically compute the orthogonalization in one step,

$$\mathbf{w}_j \leftarrow \mathbf{w}_j - \mathcal{P}_{\mathcal{K}_j} \mathbf{w}_j.$$

However, in order to obtain a better numerical orthogonalization of this new vector, in the modified Gram-Schmidt process we orthogonalize $\mathbf{w}_j$ against each $\mathbf{v}_i$ individually. We present the modified Gram-Schmidt version of the Arnoldi procedure as Algorithm 2.3.1.

Observe that we have

$$\mathbf{A}\mathbf{v}_j = h_{j+1,j}\mathbf{v}_{j+1} + h_{j,j}\mathbf{v}_j + \cdots + h_{1,j}\mathbf{v}_1. \tag{2.14}$$

The coefficients $h_{\ell,i}$ can be stored as entries of $\overline{\mathbf{H}}_j \in \mathbb{C}^{j+1\times j}$, a matrix with zero entries below the first lower- subdiagonal. A matrix with this structure is called *upper Hessenberg*. This allows us to express the first j steps of the Arnoldi procedure in terms of the *Arnoldi relation*,

$$\mathbf{A}\mathbf{V}_j = \mathbf{V}_{j+1}\overline{\mathbf{H}}_j \tag{2.15}$$

which expressed (2.14) for the entire basis. If we premultiply both sides of (2.15) by $\mathbf{V}_j^*$, we obtain

$$\mathbf{V}_j^*\mathbf{A}\mathbf{V}_j = \mathbf{H}_j$$

where $\mathbf{H}_j = \left(\overline{\mathbf{H}}_j\right)_{1:j} \in \mathbb{C}^{j\times j}$. Since $\mathcal{K}_j(\mathbf{A}, \mathbf{b})$ is a subspace of dimension j, it is isomorphic with $\mathbb{C}^j$ under the mapping from $\mathbb{C}^j$ to $\mathcal{K}_j(\mathbf{A}, \mathbf{b})$

$$\mathbf{y} \longrightarrow \mathbf{V}_j\mathbf{y} \quad \text{for all} \quad \mathbf{y} \in \mathbb{C}^j.$$

Algorithm 2.3.1: Modified Gram-Schmidt Arnoldi Procedure

```
Input  : $\mathbf{A} \in \mathbb{C}^{n\times n}$, $\mathbf{u} \in \mathbb{C}^n$, $j > 0$
Output: $\{\mathbf{v}_1, \ldots, \mathbf{v}_{j+1}\}$ an orthonormal basis for $\mathcal{K}_{j+1}(\mathbf{A}, \mathbf{u})$,
        $\overline{\mathbf{H}}_j \in \mathbb{C}^{j+1\times j}$
1 $\mathbf{v}_1 = \mathbf{u}/\|\mathbf{u}\|$
2 for $i = 1 \ldots j$ do
3     $\mathbf{w} \leftarrow \mathbf{A}\mathbf{v}_i$
4     for $\ell = 1 \ldots i$ do
5         $h_{\ell,i} = \mathbf{v}_\ell^*\mathbf{w}$
6         $\mathbf{w} \leftarrow \mathbf{w} - h_{\ell,i}\mathbf{v}_\ell$
7     $h_{i+1,i} = \|\mathbf{w}\|$
8     $\mathbf{v}_{i+1} = \mathbf{w}/h_{i+1,i}$
```

This fact can be used to interpret the action of the operator $\mathbf{H}_j$ on $\mathbb{C}^j$. Observe that we can write

$$\mathbf{H}_j\mathbf{y} = \mathbf{V}_j^*\mathbf{A}(\mathbf{V}_j\mathbf{y}). \tag{2.16}$$

This is the composition of three operations: $\mathbf{y}$ is lifted to its representation in $\mathbf{C}^n$; $\mathbf{A}$ acts upon this representation; and the result is premultiplied by $\mathbf{V}_j^*$, which has the effect of projecting the result back into the Krylov subspace and representing it in $\mathbb{C}^j$. Thus $\mathbf{H}_j$ is the restriction and projection of the action of the matrix $\mathbf{A}$ onto the Krylov subspace.

Different Krylov subspace iterative methods for solving linear systems are classified by which Petrov-Galerkin conditions are being imposed on the residual at each iteration. For non-Hermitian systems, one of the simplest methods to describe is Arnoldi's method for linear systems and its most popular algorithmic implementation, called the full orthogonalization method (FOM), see e.g., [75, Section 6.4]. Let $\mathbf{x}_j^F$ be the approximation produced by Arnoldi's method and $\mathbf{r}_j^F = \mathbf{b} - \mathbf{A}\mathbf{x}_j^F$. We impose the condition that at the jth step, the residual must satisfy

$$\mathbf{r}_j^F \perp \mathcal{K}_j(\mathbf{A}, \mathbf{b}). \tag{2.17}$$

What does this mean in practice? Let $\mathbf{x}_j^F = \mathbf{V}_j \mathbf{y}_j^F$. Then we can restate the condition (2.17) in an equation formulation,

$$\begin{aligned} \mathbf{V}_j^*(\mathbf{b} - \mathbf{A}\mathbf{x}_j^F) &= \mathbf{0} \qquad (2.18) \\ \mathbf{V}_j^*(\|\mathbf{b}\| \, \mathbf{v}_1) - \mathbf{V}_j^* \mathbf{A} \mathbf{V}_j \mathbf{y}_j^F &= \mathbf{0} \\ \mathbf{V}_j^*(\|\mathbf{b}\| \, \mathbf{v}_1) &= \mathbf{V}_j^* \mathbf{A} \mathbf{V}_j \mathbf{y}_j^F \\ \|\mathbf{b}\| \, \mathbf{e}_j^{(1)} &= \mathbf{H}_j \mathbf{y}_j^F. \end{aligned}$$

Thus, we can compute $\mathbf{y}_j^F = \|\mathbf{b}\| \, \mathbf{H}_j^{-1} \mathbf{e}_j^{(1)}$ and $\mathbf{x}_j^F = \mathbf{V}_j \mathbf{y}_j^F$ is the jth iterate satisfying the Petrov-Galerkin condition (2.17). The FOM algorithm is the most well-known implementation of Arnoldi's Method for Linear Systems for when the coefficient matrix is non-Hermitian. We present it as Algorithm 2.3.2 for reference.

Observe that this Krylov method allows us to reduce the solution of a large linear systems to the solution of a sequence of smaller linear systems to which we can apply a direct method, in this case LU-factorization. This is a prototype implementation which would not be used in practice. The LU-factorization $\mathbf{H}_j = \mathbf{L}_j \mathbf{U}_j$ can be constructed at each step by storing the LU-factorization $\mathbf{H}_{j-1} = \mathbf{L}_{j-1} \mathbf{U}_{j-1}$ and performing row operations on the jth columns of $\mathbf{H}_j$ since $\mathbf{H}_{j-1}$ is the upper left-hand block of $\mathbf{H}_j$.

As we have previously discussed, we can interpret Krylov subspace methods in terms of polynomial interpolation. Krylov subspaces can be used to make eigenvalue and eigenvector approximations. We begin with a theorem which is a version of Theorem 1.2 from [88, Chapter 4].

Theorem 2.6: Let $\mathbf{F} \in \mathbb{C}^{n \times j}$ be a matrix with orthonormal columns spanning an eigenspace of $\mathbf{A}$ and $\mathbf{L} = \mathbf{F}^*\mathbf{A}\mathbf{F} \in \mathbb{C}^{j \times j}$. If $(\theta, \mathbf{y})$ is an eigenpair of $\mathbf{L}$ then $(\theta, \mathbf{F}\mathbf{y})$ is an eigenpair of $\mathbf{A}$.

Let us now suppose that instead of having a matrix $\mathbf{F}$ whose columns span an exact eigenspace of $\mathbf{A}$, we have $\widehat{\mathbf{F}} \in \mathbb{C}^{n \times j}$ with orthonormal columns spanning an **approximate** eigenspace of $\mathbf{A}$, and let $\widehat{\mathbf{L}} = \widehat{\mathbf{F}}^*\mathbf{A}\widehat{\mathbf{F}}$. We call $\widehat{\mathbf{L}}$ the *block Rayleigh quotient* of $\mathbf{A}$ associated to Range($\widehat{\mathbf{F}}$). The eigenvalues of $\widehat{\mathbf{L}}$ are approximations to the eigenvalues of $\mathbf{A}$. The quality of these approximations is linked to the quality of of Range($\widehat{\mathbf{F}}$) as an approximation to an eigenspace of $\mathbf{A}$. See, e.g., [88, Chapter 4]

Algorithm 2.3.2: The Full Orthogonalization Method

```
Input  : $\mathbf{A} \in \mathbb{C}^{n \times n}$, $\varepsilon > 0$, the convergence tolerance
Output: An approximate solution $\overline{\mathbf{x}}$ satisfying $\|\mathbf{b} - \mathbf{A}\overline{\mathbf{x}}\| / \leq \varepsilon \|\mathbf{b}\|$
1  $\overline{\mathbf{x}} \leftarrow \mathbf{0}$
2  $\mathbf{v}_1 = \mathbf{b} / \|\mathbf{b}\|$
3  $i \leftarrow 1$
4  while $\|\mathbf{b} - \mathbf{A}\overline{\mathbf{x}}\| > \varepsilon \|\mathbf{b}\|$ do
5      $\mathbf{w} \leftarrow \mathbf{A}\mathbf{v}_i$
6      for $\ell = 1 \ldots i$ do
7          $h_{\ell,i} = \mathbf{v}_\ell^*\mathbf{w}$
8          $\mathbf{w} \leftarrow \mathbf{w} - h_{\ell,i}\mathbf{v}_\ell$
9      $h_{i+1,i} = \|\mathbf{w}\|$
10     $\mathbf{v}_{i+1} = \mathbf{w}/h_{i+1,i}$
11     Compute LU-factorization $\mathbf{H}_i = \mathbf{L}_i\mathbf{U}_i$
12     $\mathbf{y} \leftarrow \|\mathbf{b}\| \mathbf{U}_i^{-1}\mathbf{L}_i^{-1}\mathbf{e}_i^{(1)}$
13     $\overline{\mathbf{x}} \leftarrow \mathbf{V}_i\mathbf{y}$
14     $i \leftarrow i + 1$
```

for a more detailed discussion of perturbation theory governing the quality of these approximations.

We can consider $\mathbf{H}_j$ as the block Rayleigh quotient of $\mathbf{A}$ associated to the Krylov subspace $\mathcal{K}_j(\mathbf{A}, \mathbf{b})$. Thus, if $(\theta, \mathbf{w})$ is an eigenpair of $\mathbf{H}_j$ then the pair $(\theta, \mathbf{V}_j\mathbf{w})$ is an approximate eigenpair of $\mathbf{A}$. The quality of these approximations depend upon how well $\mathcal{K}_j(\mathbf{A}, \mathbf{b})$ approximates an invariant subspace of $\mathbf{A}$. These approximate eigenvalues and eigenvectors are called *Ritz values* and and *Ritz vectors*, respectively. Ritz values generally provide good approximations to *exterior* eigenvalues, i.e., eigenvalues furthest from zero [69]. The jth FOM approximation $\mathbf{x}_j^F$ can be described in terms of these Ritz values.

Theorem 2.7: [37] Let $\left\{\theta_i^F\right\}_{i=1}^j$ be the j Ritz values of $\mathcal{K}_j(\mathbf{A}, \mathbf{b})$. Let $p_j^F(x)$ be the interpolation polynomial such that $p_j^F(\theta_i^F) = 1/\theta_i^F$ for $1 \leq i \leq j$. Then the jth FOM approximation satisfies $\mathbf{x}_j^F = p_j^F(\mathbf{A})\mathbf{b}$. Furthermore, let $r_j^F(x) = 1 - xp_j^F(x)$ be the polynomial with roots at the at the Ritz values $\left\{\theta_i^F\right\}_{i=1}^j$. Then the jth FOM residual satisfies $\mathbf{r}_j^F = r_j^F(\mathbf{A})\mathbf{b}$.

The FOM algorithm is an excellent prototype Krylov subspace method. However, for non-Hermitian systems, it is seldom used in practice. The algorithm can suffer from breakdown at certain iterations if $\mathbf{H}_j$ is singular. Using the interpretation of the action of $\mathbf{H}_j$ in (2.16), this condition is equivalent to saying that there exist vectors in $\mathcal{K}_j(\mathbf{A}, \mathbf{b})$ which $\mathbf{A}$ maps to the orthogonal complement of $\mathcal{K}_j(\mathbf{A}, \mathbf{b})$. If this occurs, then we simply do not compute $\mathbf{x}_j^F$ and continue to step $j+1$. Furthermore, there is a method with similar computational expense which minimizes the residual norm over all possible solutions in $\mathcal{K}_j(\mathbf{A}, \mathbf{b})$.

The GMRES method is an algorithm in which the jth approximation $\mathbf{x}_j^G$ is constructed by imposing the Petrov-Galerkin condition

$$\mathbf{r}_j^G \perp \mathbf{A}\mathcal{K}_j(\mathbf{A}, \mathbf{b}). \tag{2.19}$$

which is equivalent to selecting the minimum residual solution.

Theorem 2.8: If we solve (1.1) using a Krylov subspace method, where at step j we select an approximation $\mathbf{x}_j^G$ according to the constraint (2.19), then this is equivalent to selecting

$$\mathbf{x}_j^G = \underset{\mathbf{x}\in\mathcal{K}_j(\mathbf{A},\mathbf{b})}{\operatorname{argmin}} \|\mathbf{b} - \mathbf{A}\mathbf{x}\|$$

Proof. Let $\mathbf{y}_j \in \mathcal{K}_j(\mathbf{A}, \mathbf{b})$ be such that $\hat{\mathbf{x}}_j = \mathbf{x}_j^G + \mathbf{y}_j$ is some other approximation in $\mathcal{K}_j(\mathbf{A}, \mathbf{b})$ with residual $\hat{\mathbf{r}}_j$. Then we can write the residual norm of this alternate approximation,

$$\begin{aligned} \|\hat{\mathbf{r}}_j\| &= \|\mathbf{b} - \mathbf{A}\hat{\mathbf{x}}_j\| \\ &= \left\|\mathbf{b} - \mathbf{A}\mathbf{x}_j^G - \mathbf{A}\mathbf{y}_j\right\| \\ &= \left\|\mathbf{b} - \mathbf{A}\mathbf{x}_j^G\right\| + \|\mathbf{A}\mathbf{y}_j\| \end{aligned}$$

where the last equality is true due to the condition (2.19). Thus, $\|\hat{\mathbf{r}}_j\| \geq \left\|\mathbf{r}_j^G\right\|$ for any $\mathbf{y}_j \in \mathcal{K}_j(\mathbf{A}, \mathbf{b})$; thus, $\mathbf{r}_j^G$ is the minimum residual. $\square$

In practice, enforcing (2.19) is equivalent to solving for $\mathbf{y}_j^G$ in the equation,

$$\begin{aligned}(\mathbf{A}\mathbf{V}_j)^*(\mathbf{b} - \mathbf{A}\mathbf{x}_j^G) &= \mathbf{0} \qquad (2.20)\\ (\mathbf{V}_{j+1}\overline{\mathbf{H}}_j)^*(\|\mathbf{b}\|\,\mathbf{v}_1 - \mathbf{A}\mathbf{V}_j\mathbf{y}_j^G) &= \mathbf{0}\\ \overline{\mathbf{H}}_j^*\mathbf{V}_{j+1}^*(\|\mathbf{b}\|\,\mathbf{v}_1 - \mathbf{V}_{j+1}\overline{\mathbf{H}}_j\mathbf{y}_j^G) &= \mathbf{0}\\ \overline{\mathbf{H}}_j^*(\|\mathbf{b}\|\,\mathbf{e}_j^{(1)} - \overline{\mathbf{H}}_j\mathbf{y}_j^G) &= \mathbf{0}\\ \overline{\mathbf{H}}_j^*\,\|\mathbf{b}\|\,\mathbf{e}_j^{(1)} &= \overline{\mathbf{H}}_j^*\overline{\mathbf{H}}_j\mathbf{y}_j^G.\end{aligned}$$

These are the normal equations associated to the least squares problem

$$\mathbf{y}_j^G = \operatorname*{argmin}_{\mathbf{y}\in\mathbb{C}^j} \left\| \|\mathbf{b}\|\,\mathbf{e}_j^{(1)} - \overline{\mathbf{H}}_j\mathbf{y} \right\|. \qquad (2.21)$$

We can solve (2.21) by first computing the QR-factorization, $\overline{\mathbf{H}}_j = \mathbf{Q}_j\overline{\mathbf{R}}_j$, where $\mathbf{Q}_j \in \mathbb{C}^{(j+1)\times(j+1)}$ and $\overline{\mathbf{R}}_j \in \mathbb{C}^{(j+1)\times j}$, and solving the linear system

$$\mathbf{R}_j\mathbf{y}_j^G = \|\mathbf{b}\|\left(\mathbf{Q}_j^*\right)_{1:j}\mathbf{e}_j^{(1)}$$

where $\mathbf{R}_j = \left(\overline{\mathbf{R}}_j\right)_{1:j}$. The factorization can be computed by applying orthogonal transformations on the left of $\overline{\mathbf{H}}_j$ to transform it to the upper triangular $\overline{\mathbf{R}}_j$. We present a simple implementation of the GMRES algorithm as Algorithm 2.3.3.

The GMRES algorithm allows us to reduce the original problem to solving a sequence of small least squares problems directly, via QR-factorization. In practice, the QR-factorization is not freshly computed at each step. Since $\overline{\mathbf{H}}_j$ is an upper Hessenberg matrix, each column only has one subdiagonal entry which needs to be eliminated in the triangularization process. Each lower subdiagonal entry can be annihilated with a single unitary Givens rotation, and the product of these rotations forms $\mathbf{Q}_j^*$. Furthermore, since $\overline{\mathbf{H}}_{j-1}$ is nested as the upper left-hand block of $\overline{\mathbf{H}}_j$, we can store the sines and cosines used to construct the Givens rotations for steps $1, \ldots, j-1$ and apply them to the new columns of $\overline{\mathbf{H}}_j$. Then we need only compute a new sine and cosine to construct the Givens rotation for the subdiagonal entry of column j.

Again, we can interpret the GMRES iteration in terms of polynomial interpolation. An approximate eigenpair $(\theta, \mathbf{y})$ is a *harmonic Ritz pair* (harmonic Ritz value and harmonic Ritz vector, respectively) if $(1/\theta, \mathbf{y})$ is a Ritz pair of $\mathbf{A}^{-1}$ with respect to $\mathcal{K}(\mathbf{A}, \mathbf{b})$. Alternatively from [59], this condition is equivalent to the relation

$$\mathbf{A}^{-1}\mathbf{y} - \mu\mathbf{y} \perp \mathbf{A}\mathcal{K}(\mathbf{A}, \mathbf{b}) \text{ for all } \mathbf{y} \in \mathbf{A}\mathcal{K}(\mathbf{A}, \mathbf{b}) \text{ for } \mu = 1/\theta. \tag{2.22}$$

Algorithm 2.3.3: The GMRES Method

```
Input  : $\mathbf{A} \in \mathbb{C}^{n\times n}$, $\varepsilon > 0$, the convergence tolerance
Output: An approximate solution $\overline{\mathbf{x}}$ satisfying $\|\mathbf{b} - \mathbf{A}\overline{\mathbf{x}}\| \le \varepsilon \|\mathbf{b}\|$
1  $\overline{\mathbf{x}} \leftarrow \mathbf{0}$
2  $\mathbf{v}_1 = \mathbf{b}/\|\mathbf{b}\|$
3  $i \leftarrow 1$
4  while $\|\mathbf{b} - \mathbf{A}\overline{\mathbf{x}}\| > \varepsilon \|b\|$ do
5      $\mathbf{w} \leftarrow \mathbf{A}\mathbf{v}_i$
6      for $\ell = 1 \ldots i$ do
7          $h_{\ell,i} = \mathbf{v}_\ell^* \mathbf{w}$
8          $\mathbf{w} \leftarrow \mathbf{w} - h_{\ell,i}\mathbf{v}_\ell$
9      $h_{i+1,i} = \|\mathbf{w}\|$
10     $\mathbf{v}_{i+1} = \mathbf{w}/h_{i+1,i}$
11     Compute QR-factorization $\overline{\mathbf{H}}_i = \mathbf{Q}_i\mathbf{R}_i$
12     $\mathbf{y} \leftarrow \|\mathbf{b}\| \mathbf{R}_i^{-1}\mathbf{Q}_i^*\mathbf{e}_i^{(1)}$
13     $\overline{\mathbf{x}} \leftarrow \mathbf{V}_i\mathbf{y}$
14     $i \leftarrow i + 1$
```

Since μ is a Ritz value of $\mathbf{A}^{-1}$, it is a good approximation to an exterior eigenvalue of $\mathbf{A}^{-1}$. Therefore, θ will provide a good approximation to an *interior* eigenvalue of $\mathbf{A}$, i.e., one with magnitude close to zero. The jth GMRES approximation $\mathbf{x}_j^G$ can be described in terms harmonic Ritz values.

Theorem 2.9: [37] Let $\left\{\theta_i^G\right\}_{i=1}^{j}$ be the j harmonic Ritz values of $\mathcal{K}_j(\mathbf{A}, \mathbf{b})$. Let $p_j^G(x)$ be the interpolation polynomial such that $p_j^G(\theta_i^G) = 1/\theta_i^G$ for $1 \leq i \leq j$. Then the jth GMRES approximation satisfies $\mathbf{x}_j^G = p_j^G(\mathbf{A})\mathbf{b}$. Furthermore, let $r_j^G(x) = 1 - xp_j^G(x)$ be the polynomial with roots at the at the harmonic Ritz values $\left\{\theta_i^G\right\}_{i=1}^{j}$. Then the jth GMRES residual satisfies $\mathbf{r}_j^G = r_j^G(\mathbf{A})\mathbf{b}$.

Observe that each additional iteration of Krylov subspace methods such as FOM and GMRES requires storage of an additional vector of size n. There are general methods which do not have such storage requirements. For non-Hermitian systems, there are methods based on the non-Hermitian Lanczos method, see, e.g., [54] and [75, Chapter 7]. In this thesis, we will not work with these methods, but they should be mentioned. The non-Hermitian Lanczos method builds bases for $\mathcal{K}_j(\mathbf{A}, \mathbf{b})$ and $\mathcal{K}_j(\mathbf{A}^*, \widehat{\mathbf{b}})$ simultaneously. Neither basis is orthonormal, but the pair of bases are *biorthogonal*. That is, we construct $\{\mathbf{w}_1, \ldots, \mathbf{w}_j\}$, a basis for $\mathcal{K}_j(\mathbf{A}, \mathbf{b})$, and $\{\widehat{\mathbf{w}}_1, \ldots, \widehat{\mathbf{w}}_j\}$, a basis for $\mathcal{K}_j(\mathbf{A}^*, \widehat{\mathbf{b}})$, such that $\mathbf{w}_i^* \widehat{\mathbf{w}}_\ell = 0$ for $i \neq \ell$. This procedure can be accomplished using short-term recurrences, leading to storage efficient linear solvers such as QMR [31] and BiCGStab [95]. A newer method called IDR(s) [87] has recently been shown to be a generalization of BiCGStab [86].

For Hermitian systems, we are most fortunate. Observe that for a Hermitian matrix $\mathbf{A}$, at step j of the Arnoldi process

$$\mathbf{H}_j = \mathbf{V}_j^* \mathbf{A} \mathbf{V}_j$$

is also Hermitian. Since $\mathbf{H}_j$ is an upper Hessenberg matrix, this implies that $\mathbf{H}_j$ is tridiagonal. In this case, we frequently refer to this matrix as $\mathbf{T}_j$, indicating it is tridiagonal. This means that when we compute $\mathbf{w} = \mathbf{A}\mathbf{v}_j$ to construct the next Krylov subspace basis vector, $\mathbf{w} \perp \mathcal{K}_{j-2}(\mathbf{A}, \mathbf{b})$ automatically. There is no need to perform those orthogonalizations. The procedure corresponding to the Arnoldi method in this case is the Hermitian Lanczos method, presented as Algorithm 2.3.4.

Furthermore, Krylov subspace iterative methods based on this fact can be constructed so that $\mathbf{x}_j$ depends only on $\mathbf{x}_{j-1}$, $\mathbf{v}_j$, and $\mathbf{v}_{j-1}$. This means that Krylov

Algorithm 2.3.4: Hermitian Lanczos Algorithm

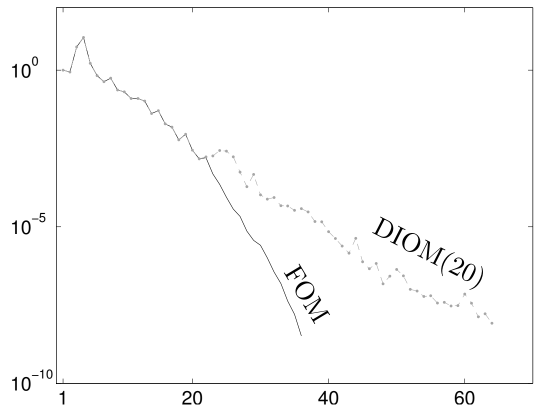
```
Input  : $\mathbf{A} \in \mathbb{C}^{n \times n}$, a Hermitian matrix and $b \in \mathbb{C}^n$ a starting vector,
         $j > 0$
Output: $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{j+1}\}$, an orthonormal basis for $\mathcal{K}_{j+1}(\mathbf{A}, \mathbf{b})$ and
         $\overline{\mathbf{T}}_j = (t_{i,\ell}) \in \mathbb{C}^{(j+1) \times j}$
1 $\mathbf{v}_1 = \mathbf{b} / \|\mathbf{b}\|$ for $i = 1 \ldots j$ do
2     $\mathbf{w} \leftarrow \mathbf{A}\mathbf{v}_i$
3     if $i > 1$ then
4         $t_{i-1,i} = t_{i,i-1}$
5         $\mathbf{w} \leftarrow \mathbf{w} - t_{i,i-1}\mathbf{v}_{i-1}$
6     $t_{i,i} = \mathbf{v}_i^* \mathbf{w}$
7     $\mathbf{w} \leftarrow \mathbf{w} - t_{i,i}\mathbf{v}_i$
8     $t_{i+1,i} = \|\mathbf{w}\|$
9     $\mathbf{v}_{j+1} = \mathbf{w} / t_{i+1,i}$
```

subspace methods for Hermitian linear systems can be written with minimal storage requirements. Some common examples of fixed-storage method for Hermitian systems are the conjugate gradient method for positive definite systems [45], and the MINRES and SYMMLQ methods for indefinite systems [64]. Unfortunately, for general non-Hermitian systems, if we want to maintain an orthonormal basis for the Krylov subspace, we must store the entire basis for the subspace [27].

Krylov subspace methods provide an effective tool for reducing the solution of large linear systems to a size for which a direct solver may be applied. However, the problems of limited storage and speed are still a concern. In a Krylov subspace method in which we maintain an orthonormal basis for the space $\mathcal{K}_j(\mathbf{A}, \mathbf{v}_1)$ an approximation is constructed in the subspace at each iteration. Many real-world problems generate matrices on the order of 10^6 or larger. This means that for each step the Krylov subspace method progresses, we must store an additional vector of size 10^6 or larger. It is unrealistic to think that the method would achieve convergence prior to memory being exhausted or the process bring slowed by the increased communication with main memory. Unfortunately, as Faber and Manteuffel showed in [27], for general, non-normal matrices, we cannot expect storage efficient short-term recurrences when constructing an orthonormal basis for a Krylov subspace. Many recent advances in the field of Krylov subspace methods have sought to mitigate this problem.

Two straightforward methods which were proposed to limit the amount of storage required for a Krylov subspace are truncated and restarted methods. Both allow the user to specify a parameter m indicating the maximum number of vectors that will be stored. A truncated method refers to one in which each new vector is not orthogonalized with respect to all previous basis vectors during the construction of the Krylov subspace, see e.g., [75, Chapter 6]. Rather, each new vector is only orthogonalized with respect to a fixed number of previous vectors, and the others are discarded. Truncated methods generate a basis for the same Krylov subspace, but this basis is nonorthogonal. At each step of the process, a new vector is generated as in the Arnoldi process. However, rather than orthogonalizing the new vector with respect to all previous vectors, we only orthogonalize with respect to the previous m basis vectors, where m is the truncation parameter. Consequently, we see a delay in convergence of the truncated methods, relative to their fully orthogonalized counterparts [83]. At each step, we store the m most recent basis vectors. Those vectors being stored span an ever-changing m- dimensional subspace. Each new basis vector is orthogonally projected away from the subspace. This particular m-dimensional subspace is the most convenient to retain, but we can use information from previous steps to choose the retained subspaces to contain useful information to use for the orthogonal projection. In Figure 2.3, we demonstrate the convergence delay by comparing FOM and truncated FOM.

A restarted method involves running a fixed number of steps of a Krylov method and using the residual at mth step as the seed to generate a new m dimensional Krylov subspace for the next cycle. This certainly limits memory usage, but restarted methods have been shown to exhibit slower convergence [83]; and, in the case of GMRES, they sometimes will not converge [50]. Furthermore, it has been demonstrated that increasing the restart parameter can produce inferior convergence



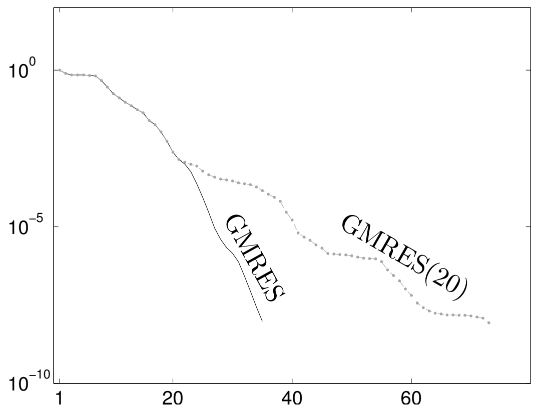


Figure 2.3. A comparison of two optimal Krylov subspace methods with nonoptimal counterparts. In the figure on the left, we compare the full orthogonalization method (FOM) with its truncated version. This implementation of the truncated version of FOM, called DIOM, is described in [75]. DIOM(20) means we orthogonalize the newest Krylov subspace vector against only the previous 20. In the figure on the right, we compare GMRES with restarted GMRES with cycle length 20.The system we are solving is the `sherman5` matrix from Matrix Market [1] preconditioned with `ILU(0)`. The right-hand side is provided with the `sherman5` matrix.

or even stagnation where restarted GMRES for a smaller parameter produces convergence [25]. This phenomenon is not well understood, and there is very little theory behind the convergence problems of restarted methods [75, 79]. It is clear, though, that they relate to the fact that at each restart, we are discarding all of the information used to build the Krylov subspace. In Figure 2.3, we present a comparison of the convergence of GMRES versus restarted GMRES with a restart parameter of 20.

In the context of restarted methods, much work has been done on carrying over information from the subspace generated in one cycle to the next, to mitigate the convergence problems observed in restarted methods. A theoretical analysis of such procedures is presented in [23]. Many of these methods involve generating a few approximate eigenvectors (either Ritz vectors or harmonic Ritz vectors) to enrich the subspace at the next cycle, see, e.g. [57, 59, 74]. Chapman and Saad presented some good theoretical results on augmented Krylov subspace techniques in [74]. We may also generate an orthonormal basis for the subspace spanned by these approximate eigenvectors and include them in the orthogonalization procedure in the next cycle. The classic truncated method can be thought of as a variant on this idea in that we store the m dimensional subspace spanned by the most recent Arnoldi vectors to orthogonalize the next basis vector against. However, simple truncation ignores important information about the Krylov subspace which is being generated at each step. Methods have been proposed that store different subspaces with more important information.

For restarted methods, recent ideas have been proposed to try to capture and retain the most important information about the subspace which has been generated at one cycle and select the optimal solution from an augmented Krylov subspace generated at the next cycle. These methods do not necessarily rely on approximate invariant subspaces or spectral information. This strategy allows us to limit the number of vectors stored while ensuring that we retain the most important information from the subspace. We will restrict this discussion to minimal residual methods, such as GMRES [76] and GCR [24]. Generalize Conjugate Residual (GCR) is a minimum residual method which is mathematically equivalent to GMRES but can produce numerically different approximations. Historically, GMRES has been the preferred method due to its superior stability properties, but this stability comes at a price.

GMRES is restrictive in terms of how it can be modified for orthogonalization against other subspaces. Morgan's GMRES-DR [59] is restricted to the use of approximate invariant subspaces spanned by Ritz or harmonic Ritz vectors.

Such restriction is why GCR serves as the basis from which algorithms to select optimal solutions from arbitrary subspaces have been built. In [90], a method called GCRO is presented, which introduces the machinery through which the user can compute the optimal correction over arbitrary subspaces.

This concept was extended by de Sturler [91], who provided a framework for selecting the optimal subspace to retain from one cycle to the next, and this method is explained more concisely in [68, Section 2.2]. Suppose we run j steps of GMRES, generating $\mathcal{K}_j(\mathbf{A}, \mathbf{b})$ and computing the GMRES residual $\mathbf{r}_j$. This residual is optimal over all approximations in $\mathcal{K}_j(\mathbf{A}, \mathbf{b})$. What if, instead, we run only $m < j$ steps of GMRES, generating the subspace $\mathcal{K}_m(\mathbf{A}, \mathbf{b})$ and the GMRES residual $\mathbf{r}_m$. Then, we restart and run $j - m$ more steps of GMRES generating the subspace $\mathcal{K}_{j-m}(\mathbf{A}, \mathbf{r}_m)$ and restarted GMRES residual $\hat{\mathbf{r}}_{j-m}$. Both $\mathbf{r}_j$ and $\hat{\mathbf{r}}_{j-m}$ require j iterations to construct. We would like to characterize the difference $\hat{\mathbf{r}}_{j-m} - \mathbf{r}_j$; what did we lose by restarting? This question can be rephrased; what is the price of not constructing $\mathcal{K}_{j-m}(\mathbf{A}, \hat{\mathbf{r}}_m)$ orthogonal to $\mathcal{K}_m(\mathbf{A}, \mathbf{b})$? This penalty was quantified by de Sturler in terms of the principal angles between $\mathbf{A}\mathcal{K}_{j-m}(\mathbf{A}, \hat{\mathbf{r}}_m)$ and $\mathbf{A}\mathcal{K}_m(\mathbf{A}, \mathbf{b})$ which can be computed at little additional expense using information in the subspace. Furthermore, what if instead of discarding the entire subspace $\mathcal{K}_m(\mathbf{A}, \mathbf{b})$, we wish to retain a k-dimensional subspace and maintain orthogonality with respect to this subspace? We can use the same framework to select an "optimal" k-dimensional subspace. In this case, optimal means we select the dimension k subspace of $\mathcal{K}_m(\mathbf{A}, \mathbf{b})$ which was most important to maintain orthogonality against in terms of the residual reduction of $\mathbf{r}_m$. The assumption being made here is that if maintaining orthogonality to this k-dimensional subspace was important in the current cycle, it will also be important to maintain orthogonality against it in the next cycle.

This algorithm is called GCROT where OT stands for optimal truncation. A simplified version of the GCROT approach, based on restarted GMRES (called LGMRES) is presented in [7]. Parks et al. in [68] combine the ideas of [91] and [59] and extend them to multiple related linear systems. In this work it is suggested that the harmonic Ritz vectors associated to the smallest harmonic Ritz values can be used as an effective recycled subspace. Parks et al. call these *subspace recycling* methods and, specifically, presented a method which they called GCRODR. [68]. We will return to a discussion of subspace recycling in Chapter 4.

CHAPTER 3

NEARLY-HERMITIAN LINEAR SYSTEMS

3.1 Introduction

A variety of applications warrant the solution of linear systems of equations where the coefficient matrix $\mathbf{A}$ has a skew-Hermitian part $\frac{1}{2}(\mathbf{A} - \mathbf{A}^*)$ with low rank. Such systems arise, for example, from discretized integral equations derived from wave scattering applications and electrostatics [78], as well as path following methods [8]. In this chapter, we consider efficient Krylov subspace methods for the solution of (1.1) where the nonsingular coefficient matrix $\mathbf{A} \in \mathbb{C}^{n\times n}$ has the structure

$$\mathbf{A} = \mathbf{A}^* + \mathbf{F}\mathbf{G}^* = \mathbf{H} + \tfrac{1}{2}\mathbf{F}\mathbf{G}^*, \tag{3.1}$$

for full rank $\mathbf{F}$, $\mathbf{G} \in \mathbb{C}^{n\times s}$ with $s \ll n$; $\mathbf{H} := \frac{1}{2}(\mathbf{A} + \mathbf{A}^*)$ is the Hermitian part of $\mathbf{A}$.

Beckermann and Reichel [8] proposed a “Progressive GMRES” algorithm based on a short recurrence for problems with this structure (3.1), which is mathematically equivalent to the much less efficient full GMRES method [76]. We have found that Progressive GMRES, while theoretically elegant, can suffer from fundamental numerical instabilities that render it unsuitable for some otherwise benign matrices of the form (3.1). We carefully document these instabilities and propose an alternative algorithm that uses a Schur complement approach to solve systems with this structure using short-term recurrence relations. While our method is not equivalent to full GMRES for the original system, it is based on an optimal method (MINRES) for the Hermitian part of $\mathbf{A}$, and can be applied as either a solver or a preconditioner.

We begin by introducing Progressive GMRES (PGMRES) in the next section. In Section 3.2.3 we present numerical experiments that demonstrate the instability of PGMRES for some well-conditioned linear systems, then comment on possible causes for this instability. We propose in Section 3.3 an alternative algorithm for the solution of (1.1) with $\mathbf{A}$ of the form (3.1) based on existing Schur complement methods. In Section 3.3.2, we use this method to precondition coefficient matrices that are small-norm perturbations of a matrix of form (3.1). We present numerical experiments in Section 3.3.3 that compare our new method to existing ones.

3.2 The Progressive GMRES Algorithm

3.2.1 Derivation of the Algorithm

As we have discussed in Chapter 2, for a generic, non-Hermitian matrix $\mathbf{A}$, the GMRES method approximates the solution to (1.1) in an iterative process that at iteration j solves the least squares problem

$$\mathbf{x}_j^G = \text{argmin}_{\mathbf{x}\in\mathcal{K}_j(\mathbf{A},\mathbf{b})} \left\|\mathbf{b} - \mathbf{A}\mathbf{x}\right\|, \tag{3.2}$$

with $\mathbf{x}_0 = \mathbf{0}$, see, e.g., [75, Ch. 6], [76] or [85] which is equivalent to solving the least squares problem

$$\mathbf{y}_j^G = \mathrm{argmin}_{\mathbf{y}\in\mathbb{C}^j} \left\| \|\mathbf{b}\|\mathbf{e}_1 - \overline{\mathbf{H}}_j\mathbf{y}\right\|, \tag{3.3}$$

and then taking $\mathbf{x}_j = \mathbf{V}_j\mathbf{y}_j$.

As discussed in Chapter 2, in the case of an Hermitian matrix $\mathbf{A} = \mathbf{A}^*$, the Hessenberg matrix $\mathbf{H}_j = \mathbf{V}_j^*\mathbf{A}\mathbf{V}_j = \mathbf{H}_j^*$ is also Hermitian, and thus tridiagonal. In this case, we instead use the notation $\overline{\mathbf{T}}_j$ and $\mathbf{T}_j$ for the now tridiagonal rectangular and square matrices. Thus, we now have the Lanczos relation

$$\mathbf{A}\mathbf{V}_j = \mathbf{V}_{j+1}\overline{\mathbf{T}}_j \tag{3.4}$$

which is really just a special case of the Arnoldi relation (2.15). In such a case, the last column of $\overline{\mathbf{T}}_j$ has only three nonzero entries implying that $\mathbf{A}\mathbf{v}_j$ is a linear combination of $\mathbf{v}_{j+1}$, $\mathbf{v}_j$ and $\mathbf{v}_{j-1}$. Explicitly using this fact in the Lanczos relation (3.4) provides a three-term recurrence

$$t_{j+1,j}\mathbf{v}_{j+1} = \mathbf{A}\mathbf{v}_j - t_{j,j}\mathbf{v}_j - t_{j-1,j}\mathbf{v}_{j-1}, \tag{3.5}$$

and the aforementioned Hermitian Lanczos procedure, presented as Algorithm 2.3.4 [75, Ch. 6]. A widely-used method, equivalent to GMRES in exact arithmetic, for solving (1.1) with $\mathbf{A}$ Hermitian indefinite via (2.21) using the Lanczos iteration is the MINRES method of Paige and Saunders [64]. In MINRES, the jth minimum residual iterate $\mathbf{x}_j^M$ can be computed progressively, i.e., $\mathbf{x}_j^M = \mathbf{x}_{j-1}^M + \mathbf{m}_j$, where $\mathbf{m}_j$ is an auxiliary vector computed at each iteration; $\mathbf{m}_j$ is frequently referred to as the *search direction*. It is shown in [64] that to construct $\mathbf{m}_j$, we only require

$\mathbf{m}_{j-1}$, $\mathbf{m}_{j-1}$, and $\mathbf{v}_j$, see also, e.g., [93]. This results in great savings in storage when compared to the Arnoldi-driven GMRES.

Is a similar simplification possible if $\mathbf{A}$ is not Hermitian, but only "nearly Hermitian," in the sense of (3.1)? Beckermann and Reichel [8] demonstrate that for matrices with such structure, the Arnoldi process can be updated through a short recurrence involving a rank-s projection. We provide here a simple derivation of their algorithm. The key idea is similar to the simplification that led to the Lanczos method: if $\mathbf{A}$ is Hermitian, then so is $\mathbf{H}_j$; if the skew-Hermitian part of $\mathbf{A}$ is low-rank, then so is the skew-Hermitian part of $\mathbf{H}_j$.[1] To see this, note that for $\mathbf{A}$ of the form (3.1),

$$\mathbf{H}_j \;=\; \mathbf{V}_j^*(\mathbf{A}^* + \mathbf{F}\mathbf{G}^*)\mathbf{V}_j \;=\; \mathbf{H}_j^* + (\mathbf{V}_j^*\mathbf{F})(\mathbf{V}_j^*\mathbf{G})^*.$$

Inserting this representation of $\mathbf{H}_j$ into the Arnoldi relation (2.15) yields

$$h_{j+1,j}\mathbf{v}_{j+1} \;\;=\;\; \mathbf{A}\mathbf{v}_j - \overline{h}_{j,j}\mathbf{v}_j - h_{j,j-1}\mathbf{v}_{j-1} - \mathbf{V}_j\mathbf{V}_j^*\mathbf{F}\mathbf{G}^*\mathbf{v}_j. \tag{3.6}$$

(The subdiagonal entry $h_{j,j-1}$, derived from the normalization of $\mathbf{v}_j$, is a nonnegative real number.) The orthogonal projection of $\mathbf{F}$ onto the Krylov subspace, $\mathbf{V}_j\mathbf{V}_j^*\mathbf{F}$, can be computed progressively; that is, defining $\widetilde{\mathbf{F}}_j := \mathbf{V}_j\mathbf{V}_j^*\mathbf{F}$, we see that

$$\widetilde{\mathbf{F}}_j = \widetilde{\mathbf{F}}_{j-1} + \mathbf{v}_j\mathbf{v}_j^*\mathbf{F},$$

and therefore (3.6) is a three-term recurrence. From (3.6), $\mathbf{A}\mathbf{v}_j - \mathbf{V}_j\mathbf{V}_j^*\mathbf{F}\mathbf{G}^*\mathbf{v}_j \in \mathcal{K}_{j+1}(\mathbf{A},\mathbf{b})$ is orthogonal to $\mathcal{K}_{j-2}(\mathbf{A},\mathbf{b})$, i.e., the Arnoldi process can implicitly orthogonalize $\mathbf{A}\mathbf{v}_j$ against $\mathcal{K}_{j-2}(\mathbf{A},\mathbf{b})$ by subtracting the (progressively-computed)

[1]A similar observation can be exploited to efficiently reduce such $\mathbf{A}$ to Hessenberg form for eigenvalue computations; see, e.g., [98].

term $\mathbf{V}_j\mathbf{V}_j^*\mathbf{F}\mathbf{G}^*\mathbf{v}_j$; explicit orthogonalization against $\mathbf{v}_j$ and $\mathbf{v}_{j-1}$ is then required to form $\mathbf{v}_{j+1}$. Given the resulting short-term Arnoldi iteration, the iterate $\mathbf{x}_j$ can be computed progressively without needing to access the complete set of Arnoldi vectors $\mathbf{V}_j$, and using only $s+3$ vectors of storage. See [8] for a full description of this process.

This Progressive GMRES (PGMRES) algorithm has two advantages over full GMRES: it only requires storage of three Arnoldi vectors, rather than the entire set, and it avoids the growing computational complexity of orthogonalizing the new Arnoldi vector against the previous Krylov subspace. Unfortunately, we have found that PGMRES suffers a critical drawback: it can introduce numerical instabilities that cause the residual to stagnate well before convergence – even for some well-conditioned examples – as illustrated in Section 3.2.3.

3.2.2 Reinterpretation of Full Arnoldi

It is important to remember that the derivation of Beckermann and Reichel's method is not restricted to nearly-Hermitian linear systems. Their analysis works in theory for any skew-Hermitian rank s, even $s = n$. However, the discussion is restricted to $s \ll n$ since they are deriving a storage-efficient Krylov method. Suppose, though, that we have no restriction on s. Let $\mathbf{K} = \frac{1}{2}\mathbf{F}\mathbf{G}^*$ be the rank s skew-Hermitian part of $\mathbf{A}$. Suppose we have already generated $\mathbf{V}_j$. According to the analysis of Beckermann and Reichel, we have that

$$\mathbf{A}\mathbf{v}_j - 2\mathbf{V}_j\mathbf{V}_j^*\mathbf{K}\mathbf{v}_j \in \mathcal{K}_{j+1}(\mathbf{A},\mathbf{b}) \quad \text{and} \quad \mathbf{A}\mathbf{v}_j - 2\mathbf{V}_j\mathbf{V}_j^*\mathbf{K}\mathbf{v}_j \perp \mathcal{K}_{j-2}(\mathbf{A},\mathbf{b})$$

as evidenced by the equation

$$\mathbf{A}\mathbf{v}_j - 2\mathbf{V}_j\mathbf{V}_j^*\mathbf{K}\mathbf{v}_j = h_{j+1,j}\mathbf{v}_{j+1} + \overline{h_{j,j}}\mathbf{v}_j - h_{j,j-1}\mathbf{v}_{j-1}. \tag{3.7}$$

We now substitute $\mathbf{A} = \mathbf{H} + \mathbf{K}$ into (3.7).

$$\begin{aligned} h_{j+1,j}\mathbf{v}_{j+1} + \overline{h_{j,j}}\mathbf{v}_j + h_{j,j-1}\mathbf{v}_{j-1} &= \mathbf{A}\mathbf{v}_j - 2\mathbf{V}_j\mathbf{V}_j^*\mathbf{K}\mathbf{v}_j \\ &= \mathbf{H}\mathbf{v}_j + \mathbf{K}\mathbf{v}_j - 2\mathbf{V}_j\mathbf{V}_j^*\mathbf{K}\mathbf{v}_j \\ &= \mathbf{H}\mathbf{v}_j + \left(\mathbf{I} - 2\mathbf{V}_j\mathbf{V}_j^*\right)\mathbf{K}\mathbf{v}_j \qquad (3.8) \end{aligned}$$

This leads to an interesting interpretation of the action of $\mathbf{H}$ and $\mathbf{K}$ on the Arnoldi vectors. Let us expand $\mathbf{H}\mathbf{v}_j$ and $\left(\mathbf{I} - 2\mathbf{V}_j\mathbf{V}_j^*\right)\mathbf{K}\mathbf{v}_j$ in the full Arnoldi basis of $\mathbb{C}^n$, i.e.,

$$\mathbf{H}\mathbf{v}_j = \sum_{i=1}^{n} \alpha_i \mathbf{v}_i \quad \text{and} \quad \left(\mathbf{I} - 2\mathbf{V}_j\mathbf{V}_j^*\right)\mathbf{K}\mathbf{v}_j = \sum_{i=1}^{n} \beta_i \mathbf{v}_i. \tag{3.9}$$

Then we can state the following theorem.

Theorem 3.1: Suppose we have generated an orthonormal basis $\{\mathbf{v}_1, \ldots, \mathbf{v}_j\}$ for $\mathcal{K}_j(\mathbf{A}, \mathbf{b})$. Let $\mathbf{H}\mathbf{v}_j$ and $\left(\mathbf{I} - 2\mathbf{V}_j\mathbf{V}_j^*\right)\mathbf{K}\mathbf{v}_j$ have expansions in the full Arnoldi basis of $\mathbb{C}^n$ shown in (3.9). Then we have that

$$\mathbf{K}\mathbf{v}_j = \sum_{i=1}^{j-2} \alpha_i \mathbf{v}_i - \beta_{j-1}\mathbf{v}_{j-1} - \beta_j \mathbf{v}_j + \beta_{j+1}\mathbf{v}_{j+1} - \sum_{i=j+2}^{n} \alpha_i \mathbf{v}_i. \tag{3.10}$$

where

$$\beta_{j-1} = h_{j,j-1} - \alpha_{j-1}, \quad \beta_j = \overline{h_{j,j}} - \alpha_j, \text{ and } \quad \beta_{j+1} = h_{j+1,j} - \alpha_{j+1} \tag{3.11}$$

Proof. First, observe that (3.7) and (3.8) imply that

$$\alpha_{j-1} + \beta_{j-1} = h_{j,j-1}, \quad \alpha_j + \beta_j = \overline{h_{j,j}}, \text{ and } \quad \alpha_{j+1} + \beta_{j+1} = h_{j+1,j},$$

which implies (3.11). They also imply that $\alpha_i = -\beta_i$ for $i < j-1$ and $i > j+1$ since the sum in (3.8) is in span $\{\mathbf{v}_{j-1}, \mathbf{v}_j, \mathbf{v}_{j+1}\}$. Therefore, we have

$$\left(\mathbf{I} - 2\mathbf{V}_j\mathbf{V}_j^*\right)\mathbf{K}\mathbf{v}_j = -\sum_{i=1}^{j-2}\alpha_i\mathbf{v}_i + \sum_{i=j-1}^{j+1}\beta_i\mathbf{v}_i - \sum_{i=j+2}^{n}\alpha_i\mathbf{v}_i.$$

Now, since $\left(\mathbf{I} - 2\mathbf{V}_j\mathbf{V}_j^*\right)\mathbf{K}\mathbf{v}_j$ is a reflection of $\mathbf{K}\mathbf{v}_j$ across $\mathcal{K}_j(\mathbf{A}, \mathbf{b})$, we can simply invert the reflection to conclude (3.10). ☐

3.2.3 Instability of Progressive GMRES: Analysis and Illustration

We begin by applying PGMRES to a problem from acoustic scattering, then move to a contrived class of well-conditioned normal matrices: in both cases our numerical experiments demonstrate instabilities that cause the residual norm to stagnate well before reaching convergence. For such examples, we compare the residuals produced by PGMRES to those generated by a standard implementation of GMRES based on the modified Gram–Schmidt Arnoldi process [75]. Since the iterate produced by PGMRES is computed progressively, for the sake of comparison we explicitly compute the residual norm $\|\mathbf{b} - \mathbf{A}\mathbf{x}_k\|$ at each step. To gain insight into the onset of instability, we also track the departure from orthonormality of the Arnoldi basis, and the rate at which these basis vectors drift toward linear dependence. We gauge the departure from orthonormality using a measure advocated by Paige et al. [65, 66]. Assume the columns of $\mathbf{V}_j$ have norm 1, and let $\mathbf{U}_j$ denote the strictly upper triangular part of $\mathbf{V}_j^*\mathbf{V}_j - \mathbf{I} = \mathbf{U}_j + \mathbf{U}_j^*$. Now define $\mathbf{S}_j := (\mathbf{I} + \mathbf{U}_j)^{-1}\mathbf{U}_j$. The departure from orthonormality can be measured by $\|\mathbf{S}_j\| \in [0,1]$, which is zero when the columns of $\mathbf{V}_j$ are orthonormal and one when those columns are linearly dependent [65, Thm. 2.1]. Moreover, [65, Cor. 5.2], [66, Lem. 5.1],

$$\kappa(\mathbf{V}_j) \leq \frac{1 + \|\mathbf{S}_j\|}{1 - \|\mathbf{S}_j\|}.$$

Each numerical experiment in this section shows $\|\mathbf{S}_j\|$ (though the columns of the computed $\mathbf{V}_j$ are only normalized to machine precision).

3.2.4 The Lippmann–Schwinger Equation

The Lippmann–Schwinger integral equation models acoustic scattering in one dimension at wave number κ:

$$(I + K)u(x) = u^i(x), \tag{3.12}$$

where K is the integral operator $K : L^2(0, 2\pi) \to L^2(0, 2\pi)$ given by

$$(Ku)(x) = \frac{i\kappa}{2} \int_0^{2\pi} e^{i\kappa|x-y|}\, m(y)\, u(y)\, \mathrm{d}y \tag{3.13}$$

and $u^i(x)$ is an incident wave satisfying the one dimensional Helmholtz equation $\mathrm{d}^2u^i/\mathrm{d}x^2 + \kappa^2 u^i = 0$ [15]. In the case of constant m, the skew-adjoint part of the operator (3.13) has a two-dimensional range [78]. Furthermore, a Nyström discretization of (3.13) based on a composite Riemann quadrature rule produces a matrix whose skew-Hermitian part has rank two. (A higher order rule would not necessarily produce a discretization with a skew-Hermitian part of rank two; however a simple preconditioning on the right by the diagonal matrix of quadrature weights gives a coefficient matrix whose skew-Hermitian part is rank two. Similarly, one could obtain the desired structure by using an inner product that incorporates the quadrature weights.)

Figure 3.1 shows the convergence of the GMRES and PGMRES residuals for the Lippmann–Schwinger problem (with a random $\mathbf{b}$) for four choices of the wave number κ, all with constant $m = -1$ and discretization dimension $n = 1000$. As the wave number increases, the PGMRES residual curve departs from the standard GMRES curve, stagnating at increasingly large residuals. One might argue that the instability is not significant at $\kappa = 3$; when $\kappa = 8$ the stagnation entirely compromises the utility of the algorithm.

To investigate the stagnation, we retain all Arnoldi vectors in PGMRES (in spite of only using three at a time to compute the new iterate), and compute the departure of the computed Arnoldi basis from orthogonality, $\|(\mathbf{I} + \mathbf{U}_j)^{-1}\mathbf{U}_j\|$, at each iteration. The right plots in Figure 3.1 compare this loss of orthogonality for PGMRES to that observed for standard GMRES (with a full-length Arnoldi recurrence based on the modified Gram–Schmidt process). Both PGMRES and GMRES produce bases that lose orthogonality, but GMRES does so at a slower pace that does not significantly destabilize the convergence, consistent with the analysis of Greenbaum, Rozložník, and Strakoš [40]. Observe that the PGMRES residual departs from that produced by GMRES when the departure from orthogonality approaches one.

3.2.5 A class of simple examples

The instability of PGMRES is not confined to the application we have just illustrated. Our experiments suggest that striking examples exhibit the following features:

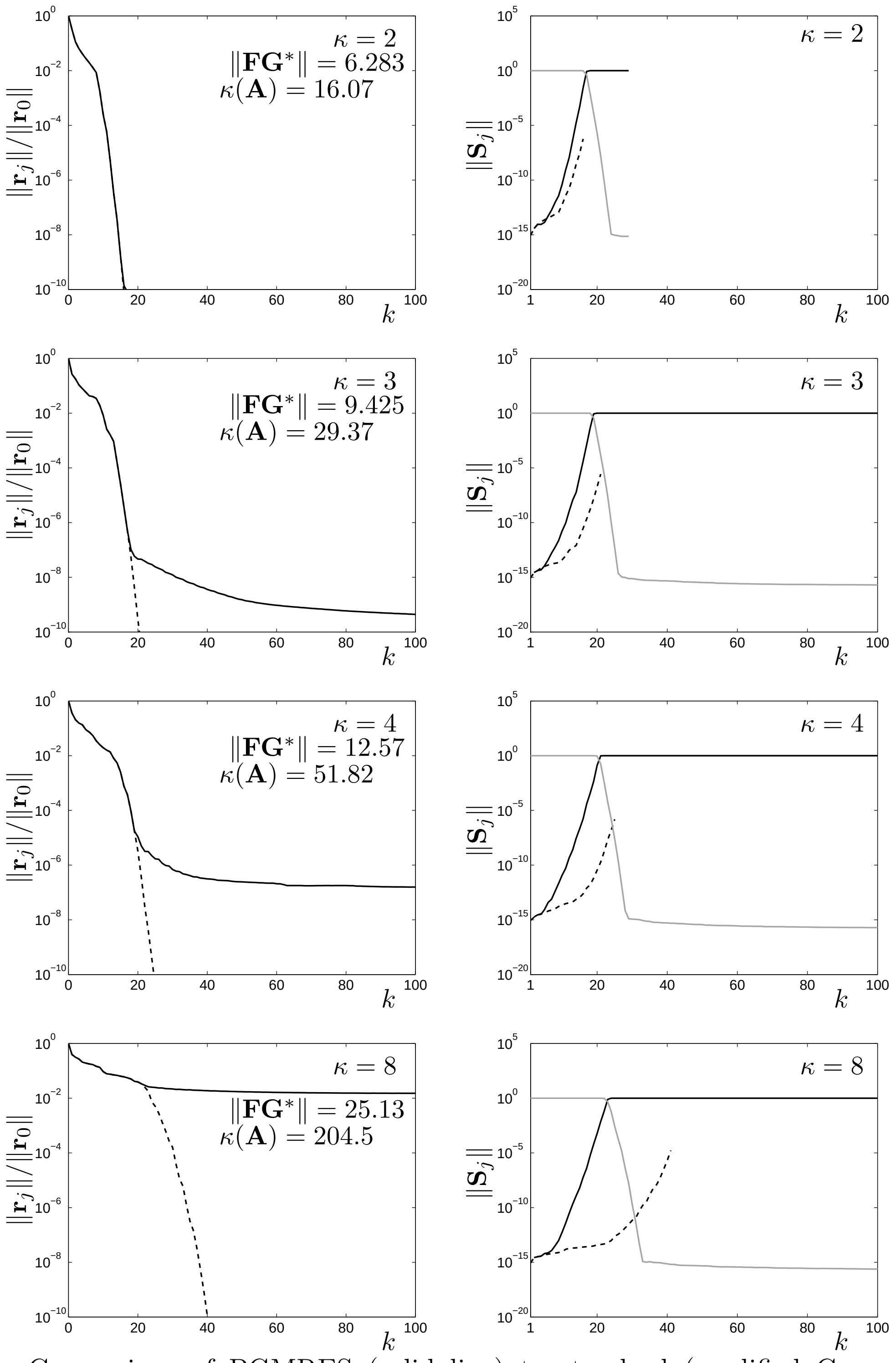


Figure 3.1. Comparison of PGMRES (solid line) to standard (modified Gram–Schmidt) GMRES (**dashed line**) applied to the one dimensional scattering problem for various values of the wave number κ. The left plot shows the relative residual norm for each method; the right plot shows the departure from orthonormality, along with the minimum singular value of the Arnoldi basis matrix $\mathbf{V}_j$ computed by the PGMRES recurrence (**gray line**). In exact arithmetic, the algorithms are identical; in finite precision, the performance of PGMRES degrades as κ increases.

(i) $\|\mathbf{F}\mathbf{G}^*\| \gg 0$, to stimulate the instability;

(ii) GMRES should initially converge slowly, during which period the Arnoldi basis generated by PGMRES degrades;

(iii) GMRES should then enter a phase of rapid convergence, which PGMRES cannot mimic due to its deficient basis.

We shall describe a class of examples that satisfies these three properties, while being normal, nearly diagonal, and even well-conditioned (for appropriate parameter values). Consider block-diagonal matrices of the form

$$\mathbf{A} = \begin{bmatrix} \boldsymbol{\Lambda}_- & & \\ & \boldsymbol{\Lambda}_+ & \\ & & \mathbf{Z} \end{bmatrix}, \tag{3.14}$$

where, for positive constants $\alpha < \beta$ and γ,

$$\boldsymbol{\Lambda}_- = \mathrm{diag}(\lambda_1, \ldots, \lambda_p), \quad \boldsymbol{\Lambda}_+ = \mathrm{diag}(\lambda_{p+1}, \ldots, \lambda_{n-2}), \quad \mathbf{Z} = \begin{bmatrix} 0 & \gamma \\ -\gamma & 0 \end{bmatrix},$$

with eigenvalues

- $\lambda_1, \ldots, \lambda_p$ uniformly spaced in the negative real interval $[-\beta, -\alpha]$;
- $\lambda_{p+1}, \ldots, \lambda_{n-2}$ uniformly spaced in the positive real interval $[\alpha, \beta]$;
- $\lambda_{n-1}, \lambda_n = \pm\gamma\mathrm{i}$, from the skew-Hermitian matrix $\mathbf{Z}$.

By construction $\mathbf{A}$ is a normal matrix with condition number

$$\kappa(\mathbf{A}) := \|\mathbf{A}\|\,\|\mathbf{A}^{-1}\| = \frac{\max\{\beta, \gamma\}}{\min\{\alpha, \gamma\}}.$$

Figure 3.2 illustrates the spectrum of a representative $\mathbf{A}$. The qualitative description of GMRES convergence provided by Driscoll, Toh, and Trefethen [21] informs this

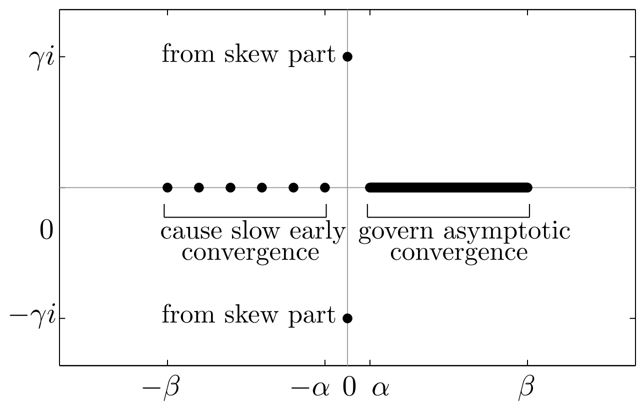


Figure 3.2. Eigenvalues (●) of the matrix (3.14) in the complex plane, for $n = 200$ and $p = 6$.

construction. The purely imaginary eigenvalues $\pm\gamma i$ control $\|\mathbf{F}\mathbf{G}^*\|$: as γ gets large, this pair has little effect on GMRES (the pair delays convergence by roughly two iterations), yet, as indicated in Figure 3.3, the magnitude of these entries induce the onset of instability. The p eigenvalues $\lambda_1, \ldots, \lambda_p$ on the negative real axis associated with the block $\mathbf{\Lambda}_-$ add indefiniteness to the problem, and further delay convergence: for early iterations, GMRES will behave in a fashion similar to MINRES applied to a matrix whose spectrum falls in $[-\beta, -\alpha] \cup [\alpha, \beta]$ (see, e.g., [38, §3.1]), whereby each iteration (asymptotically) reduces the residual norm by the factor

$$\rho_1 = \sqrt{\frac{\beta - \alpha}{\beta + \alpha}}\ .$$

After sufficiently many iterations to annihilate the outlying eigenvalues (roughly $2p + 2$ steps), GMRES then converges at the much more rapid rate

$$\rho_2 = \frac{\sqrt{\beta} - \sqrt{\alpha}}{\sqrt{\beta} + \sqrt{\alpha}}$$

expected for a matrix whose spectrum falls in the interval $[\alpha, \beta]$. (The improvement is substantial: for $\alpha = 1/8$ and $\beta = 1$, the early slow rate is $\rho_1 \approx 0.8819$, which is followed by the rate $\rho_2 \approx 0.4776$: thus it takes nearly six slow-phase iterations to reduce the residual as much as a single fast-phase iteration.) Since $\mathbf{A}$ is normal, this discussion leads to a rigorous bound on GMRES convergence: for $k \geq 2p + 2$,

$$\frac{\|\mathbf{r}_j\|}{\|\mathbf{r}_0\|} \leq (1 + \beta^2/\gamma^2)2^{p+1}\rho_2^{k-p-2}, \tag{3.15}$$

arrived at by bounding the GMRES residual polynomial with an inferior polynomial that has roots at the negative eigenvalues $\lambda_1, \ldots, \lambda_p$ and the imaginary eigenvalues $\pm\gamma i$, and behaves like a Chebyshev polynomial on $[\alpha, \beta]$; see, e.g., [38, §3.1], [93, pp. 70–71].

Figure 3.3 compares the performance of PGMRES and modified Gram–Schmidt GMRES for two instances of the matrix (3.14). The first instance is constructed to have a mild condition number; the second is more ill-conditioned due to the large value of $\gamma = \|\mathbf{FG}^*\| = 10^6$. This extra magnitude brings forward the onset of instability, which is already significant at the fifth iteration. In both instances the PGMRES basis loses orthogonality, then linear dependence, just as for the Lippmann–Schwinger example shown in Figure 3.1.

3.2.6 Analysis of PGMRES orthogonalization

We next demonstrate how the local orthogonalization that gives PGMRES its distinct performance advantage over standard GMRES can cause the numerical instabilities exhibited in the previous computations. Begin with an *exact* decomposition resulting from $j - 1$ steps of the Arnoldi process: $\mathbf{AV}_{j-2} = \mathbf{V}_{j-1}\overline{\mathbf{H}}_{j-1}$. Now suppose that the

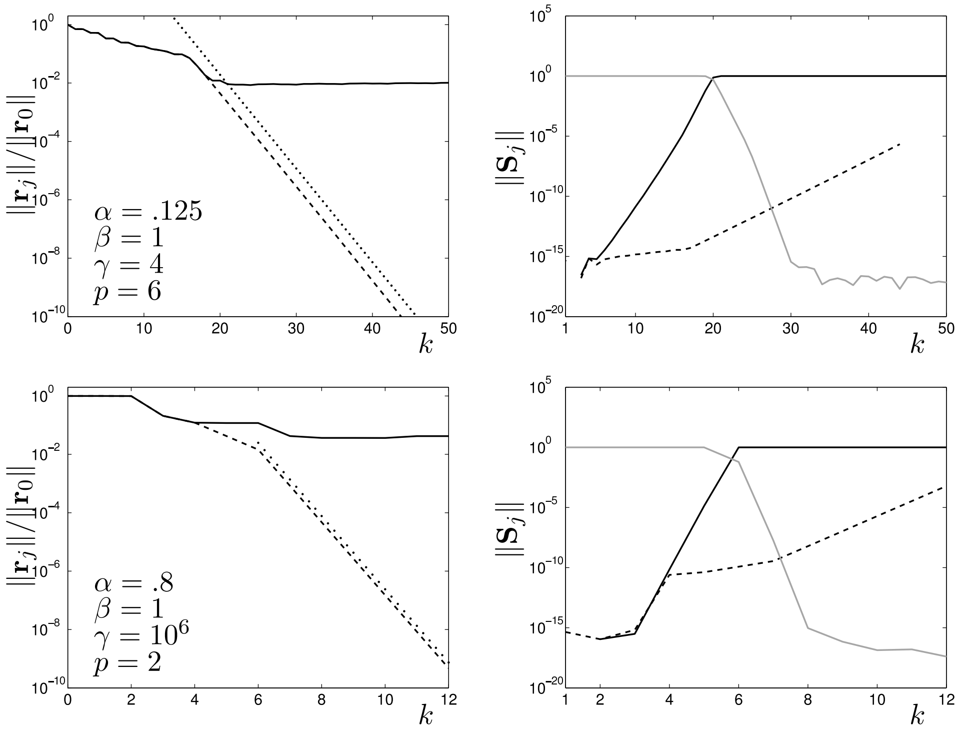


Figure 3.3. PGMRES (**solid line**) and standard (modified Gram–Schmidt) GMRES (**dashed line**) as in Figure 3.1, applied to the simple example (3.14) with a vector $\mathbf{b} = [1, \ldots, 1]^{\mathrm{T}}$ and $n = 200$. The dotted line in each left plot shows the convergence bound (3.15). The gray line in each right plot shows the smallest singular value of the Arnoldi basis matrix $\mathbf{V}_j$ computed by PGMRES. In the well-conditioned top example ($\kappa(\mathbf{A}) = 32$), the instability develops gradually; in the ill-conditioned bottom example ($\kappa(\mathbf{A}) = 1.25 \times 10^6$), the instability is apparent at the fifth iteration.

PGMRES Arnoldi process is used to compute subsequent Arnoldi vectors. We shall demonstrate how an error incurred at step j can be magnified by the next step of the PGMRES recurrence.

Let $\mathbf{V}_{j-1} = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{j-1}]$ denote the matrix whose columns are the first $j-1$ exact Arnoldi vectors, an orthonormal basis for $\mathcal{K}_{j-1}(\mathbf{A}, \mathbf{b})$. Suppose the jth computed Arnoldi vector $\widetilde{\mathbf{v}}_j$ is a unit vector with some error in direction, i.e., $\widetilde{\mathbf{v}}_j = c\mathbf{v}_j + s\mathbf{d}_j$, where $\mathbf{v}_j$ is the jth exact Arnoldi vector, $\|\mathbf{d}_j\| = 1$ and $\mathbf{d}_j \perp \mathbf{v}_j$, and

$|c|^2+|s|^2=1$ with $|s| \ll 1$. Now suppose $\widetilde{\mathbf{v}}_j$ was computed by the PGMRES Arnoldi method (that is, via *local* orthogonalization of $(\mathbf{A}-\mathbf{V}_{j-1}\mathbf{V}_{j-1}^*\mathbf{F}\mathbf{G}^*)\mathbf{v}_{j-1}$ against $\mathbf{v}_{j-1}$ and $\mathbf{v}_{j-2}$). For a first-order analysis, it is reasonable to presume the explicit orthogonalization is exact, i.e., $\widetilde{\mathbf{v}}_j \perp \{\mathbf{v}_{j-1},\mathbf{v}_{j-2}\}$, and thus $\mathbf{d}_j \perp \{\mathbf{v}_{j-1},\mathbf{v}_{j-2}\}$.

We shall use the notation $\mathbf{P}_\ell$ and $\widetilde{\mathbf{P}}_\ell$ for the orthogonal projectors onto $\operatorname{span}\{\mathbf{v}_\ell\}^\perp$ and $\operatorname{span}\{\widetilde{\mathbf{v}}_\ell\}^\perp$. To study the error in $\widetilde{\mathbf{v}}_{j+1}$, let $\mathbf{u} := (\mathbf{A}-\widetilde{\mathbf{V}}_j\widetilde{\mathbf{V}}_j^*\mathbf{F}\mathbf{G}^*)\widetilde{\mathbf{v}}_j$, and consider the unnormalized $(j+1)$st Arnoldi vector $\widetilde{\mathbf{w}}_{j+1} := \widetilde{\mathbf{P}}_j\mathbf{P}_{j-1}\mathbf{u}$ computed by PGMRES. Then

$$\begin{aligned}\mathbf{u} &= \mathbf{A}(c\mathbf{v}_j+s\mathbf{d}_j)-\widetilde{\mathbf{V}}_j\widetilde{\mathbf{V}}_j^*\mathbf{F}\mathbf{G}^*(c\mathbf{v}_j+s\mathbf{d}_j) \\ &= c(\mathbf{A}-\mathbf{V}_j\mathbf{V}_j^*\mathbf{F}\mathbf{G}^*)\mathbf{v}_j+s(\mathbf{A}-\mathbf{V}_j\mathbf{V}_j^*\mathbf{F}\mathbf{G}^*)\mathbf{d}_j \\ &\quad -c(c\bar{s}\mathbf{v}_j\mathbf{d}_j^*+s\bar{c}\mathbf{d}_j\mathbf{v}_j^*)\mathbf{F}\mathbf{G}^*\mathbf{v}_j+\mathcal{O}(s^2),\end{aligned} \tag{3.16}$$

and one can show that

$$\begin{aligned}\widetilde{\mathbf{w}}_{j+1} &= \widetilde{\mathbf{P}}_j\mathbf{P}_{j-1}\mathbf{u} \\ &= \mathbf{P}_j\mathbf{P}_{j-1}\mathbf{u}-\left(|s|^2(\mathbf{d}_j\mathbf{d}_j^*-\mathbf{v}_j\mathbf{v}_j^*)+c\bar{s}\mathbf{v}_j\mathbf{d}_j^*+s\bar{c}\mathbf{d}_j\mathbf{v}_j^*\right)\mathbf{P}_{j-1}\mathbf{u} \\ &= c\mathbf{w}_{j+1}+s\mathbf{P}_j\mathbf{P}_{j-1}(\mathbf{A}-\mathbf{V}_j\mathbf{V}_j^*\mathbf{F}\mathbf{G}^*)\mathbf{d}_j-\bar{s}c^2(\mathbf{d}_j^*\mathbf{w}_{j+1})\mathbf{v}_j \\ &\quad -s|c|^2(\mathbf{v}_j^*\mathbf{A}\mathbf{v}_j)\mathbf{d}_j+\mathcal{O}(s^2) \\ &= c\mathbf{w}_{j+1}+\mathbf{f}_{j+1}+\mathcal{O}(s^2),\end{aligned} \tag{3.17}$$

where $\mathbf{w}_{j+1} := \mathbf{P}_j\mathbf{P}_{j-1}(\mathbf{A}-\mathbf{V}_j\mathbf{V}_j^*\mathbf{F}\mathbf{G}^*)\mathbf{v}_j$ is the unnormalized $(j+1)$st exact Arnoldi vector, and

$$\mathbf{f}_{j+1} := s\mathbf{P}_j\mathbf{P}_{j-1}(\mathbf{A}-\mathbf{V}_j\mathbf{V}_j^*\mathbf{F}\mathbf{G}^*)\mathbf{d}_j-\bar{s}c^2(\mathbf{d}_j^*\mathbf{w}_{j+1})\mathbf{v}_j-s|c|^2(\mathbf{v}_j^*\mathbf{A}\mathbf{v}_j)\mathbf{d}_j, \tag{3.18}$$

so $\|\mathbf{f}_{j+1}\| = \mathcal{O}(s)$. Consider the orthogonal decomposition $c\mathbf{w}_{j+1} = \mathbf{w}_{j+1}^{\perp} + \mathbf{w}_{j+1}^{\|}$, where $\mathbf{w}_{j+1}^{\perp} \perp \widetilde{\mathbf{w}}_{j+1}$ and $\mathbf{w}_{j+1}^{\|}$ is parallel to $\widetilde{\mathbf{w}}_{j+1}$. It follows that

$$\|\mathbf{w}_{j+1}^{\perp}\| = \|c\mathbf{w}_{j+1} - \mathbf{w}_{j+1}^{\|}\| \leq \|c\mathbf{w}_{j+1} - \widetilde{\mathbf{w}}_{j+1}\| = \|\mathbf{f}_{j+1}\| + \mathcal{O}(s^2), \tag{3.19}$$

and therefore $\sin\angle(\widetilde{\mathbf{w}}_{j+1}, \mathbf{w}_{j+1}) = \|\mathbf{w}_{j+1}^{\perp}\|/\|c\mathbf{w}_{j+1}\| \leq \|\mathbf{f}_{j+1}\|/\|c\mathbf{w}_{j+1}\| + \mathcal{O}(s^2)$.

To simplify the analysis, we make one further assumption: $\widetilde{\mathbf{v}}_j \in \mathcal{K}_j(\mathbf{A}, \mathbf{v}_1)$; in other words, the lack of explicit orthogonalization of $\mathbf{A}\mathbf{v}_{j-1}$ against $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{j-3}\}$ in the PGMRES Arnoldi process only introduces errors in the space $\mathcal{K}_j(\mathbf{A}, \mathbf{v}_1)$. (We observe that this assumption is nearly attained in practical computations.) This implies that $\mathbf{d}_j = (\widetilde{\mathbf{v}}_j - c\mathbf{v}_j)/s \in \mathcal{K}_j(\mathbf{A}, \mathbf{v}_1)$. Since $\mathbf{d}_j \perp \{\mathbf{v}_{j-2}, \mathbf{v}_{j-1}, \mathbf{v}_j\}$, it follows that $\mathbf{d}_j \in \mathcal{K}_{j-3}(\mathbf{A}, \mathbf{v}_1)$. In addition, note from (3.18) that since $\mathbf{f}_{j+1} \in \mathcal{K}_j(\mathbf{A}, \mathbf{v}_1)$, we have $\mathbf{w}_{j+1} \perp \mathbf{f}_{j+1}$. Hence $\widetilde{\mathbf{w}}_{j+1} = c\mathbf{w}_{j+1} + \mathbf{f}_{j+1} + \mathcal{O}(s^2)$ from (3.17) is an orthogonal decomposition of $\widetilde{\mathbf{w}}_{j+1}$, up to $\mathcal{O}(s^2)$ terms.

Since $\mathbf{d}_j \in \mathcal{K}_{j-3}(\mathbf{A}, \mathbf{v}_1)$ implies $\mathbf{w}_{j+1} \perp \mathbf{d}_j$, we can use (3.18) to compute

$$\begin{aligned}\frac{\tan\angle(\mathbf{v}_{j+1}, \widetilde{\mathbf{v}}_{j+1})}{\tan\angle(\mathbf{v}_j, \widetilde{\mathbf{v}}_j)} &= \frac{\|\mathbf{f}_{j+1}\|/\|c\mathbf{w}_{j+1}\| + \mathcal{O}(s^2)}{|s|/|c|} \\ &= \frac{\|\mathbf{P}_j\mathbf{P}_{j-1}(\mathbf{A} - \mathbf{V}_j\mathbf{V}_j^*\mathbf{F}\mathbf{G}^*)\mathbf{d}_j - |c|^2(\mathbf{v}_j^*\mathbf{A}\mathbf{v}_j)\mathbf{d}_j\|}{\|\mathbf{P}_j\mathbf{P}_{j-1}(\mathbf{A} - \mathbf{V}_j\mathbf{V}_j^*\mathbf{F}\mathbf{G}^*)\mathbf{v}_j\|} + \mathcal{O}(s).\end{aligned} \tag{3.20}$$

When the low-rank skew-Hermitian part of $\mathbf{A}$ is large, $\left\|\frac{1}{2}\mathbf{F}\mathbf{G}^*\right\| \gg \left\|\mathbf{A} - \frac{1}{2}\mathbf{F}\mathbf{G}^*\right\|$, the formula (3.18) raises particular concern, as it suggests considerable growth in the directional error of the Arnoldi vectors generated by PGMRES. In such scenarios, matrix-vector products $\mathbf{A}\mathbf{v}$ are dominated by $\mathbf{F}\mathbf{G}^*\mathbf{v}$ for generic $\mathbf{v}$, so one expects the Krylov space $\mathcal{K}_{s+1}(\mathbf{A}, \mathbf{b})$ to essentially contain the s-dimensional subspace $\operatorname{Ran}(\mathbf{F})$. (The starting vector has no bias toward $\operatorname{Ran}(\mathbf{F})$; this space emerges through the first

s matrix-vector products with $\mathbf{A}$, i.e., $s+1$ Krylov vectors.) For $j \geq s+1$, we thus approximate $\mathbf{V}_j\mathbf{V}_j^*\mathbf{F}\mathbf{G}^* \approx \mathbf{F}\mathbf{G}^*$, thus using the form (3.1),

$$\mathbf{A} - \mathbf{V}_j\mathbf{V}_j^*\mathbf{F}\mathbf{G}^* \approx \mathbf{A}^*.$$

Similarly, for $j > s+1$ we expect $\mathbf{F}^*\mathbf{v}_j \approx \mathbf{0}$, and thus, in the notation of (3.1),

$$\mathbf{A}^*\mathbf{v}_j = (\mathbf{H} + \tfrac{1}{2}\mathbf{G}\mathbf{F}^*)\mathbf{v}_j \approx \mathbf{H}\mathbf{v}_j,$$

where $\mathbf{H}$ denotes the Hermitian part of $\mathbf{A}$. These observations suggest that the denominator in (3.20) can be approximated by $\|\mathbf{P}_j\mathbf{P}_{j-1}\mathbf{H}\mathbf{v}_j\| \leq \|\mathbf{H}\| \ll \|\mathbf{A}\|$. The same argument gives an approximation to the second term in the numerator of (3.20), via $|\mathbf{v}_j^*\mathbf{A}\mathbf{v}_j| \approx |\mathbf{v}_j^*\mathbf{H}\mathbf{v}_j| \ll \|\mathbf{A}\|$. Now the first term in that numerator behaves like

$$\mathbf{P}_j\mathbf{P}_{j-1}(\mathbf{A} - \mathbf{V}_j\mathbf{V}_j^*\mathbf{F}\mathbf{G}^*)\mathbf{d}_j \approx \mathbf{P}_j\mathbf{P}_{j-1}\mathbf{A}^*\mathbf{V}_{j-3}\mathbf{z} \tag{3.21}$$

for some $\mathbf{z} \in \mathbb{C}^{k-3}$, since $\mathbf{d}_j \in \mathcal{K}_{j-3}(\mathbf{A}, \mathbf{b})$. Presuming $\mathbf{d}_j$ to arise from an arbitrary perturbation, we expect (3.21) to be on the order of $\|\mathbf{A}\|$, and therefore (3.20) will have a large numerator and small denominator: for $j > s+1$, we expect the jth iteration can magnify the angular error in the PGMRES Arnoldi basis vector on the order of $\|\mathbf{A}\| \,/\, \|\mathbf{H}\|$. Indeed, in numerical experiments like those shown in Figures 3.1 and 3.3, increasing $\|\mathbf{F}\mathbf{G}^*\|$ brings about earlier stagnation of PGMRES. Successful PGMRES computations seem to require that $\mathbf{F}\mathbf{G}^*$ be small in both rank *and* norm.

In summary, the step that causes the loss of orthogonality in the PGMRES Arnoldi procedure is the same step that makes PGMRES so computationally attractive, alleviating the need to preserve all Arnoldi basis vectors. In cases where the skew-Hermitian part of $\mathbf{A}$ *is small in both rank and norm*, and GMRES converges steadily, our experience suggests that PGMRES may be viable; otherwise, numerical instabilities often induce stagnation before convergence to a reasonable tolerance. Though we cannot propose a repair for this instability, in the next section we suggest an alternative method for efficiently solving a nearly-Hermitian linear system. This method will have similar storage characteristics, but will avoid the numerical problems endemic to PGMRES.

3.3 The Schur Complement Method

3.3.1 Derivation

Here we seek an efficient alternative for solving linear systems with matrices of the "nearly-Hermitian" form (3.1). In this section, we present such a method, which avoids the unstable performance of PGMRES, and only requires three-term recurrences. This algorithm arises from a simple observation. A nearly- Hermitian matrix such as (3.1) can be seen as a low-rank modification of a Hermitian matrix. As we show shortly, we can thus write a nearly-Hermitian matrix in the form

$$\mathbf{A} = \mathbf{H} + \mathbf{F}\mathbf{C}\mathbf{F}^*, \tag{3.22}$$

where $\frac{1}{2}\mathbf{G} = \mathbf{F}\mathbf{C}^*$. It follows from the skew-Hermitian property of $\mathbf{F}\mathbf{G}^*$, i.e., that $\mathbf{F}\mathbf{G}^* = -\mathbf{G}\mathbf{F}^*$, that $\mathbf{F}$ and $\mathbf{G}$ have the same range; and since they are full rank, their columns span the same subspace. Therefore, there exists a unique, nonsingular, skew-Hermitian $s \times s$ matrix $\mathbf{C}$ such that $\frac{1}{2}\mathbf{G} = \mathbf{F}\mathbf{C}^*$. Indeed, using the reduced QR decomposition, $\mathbf{F} = \mathbf{Q}\mathbf{R}$, we have that $\mathbf{C}^* = 2\mathbf{R}^{-1}\mathbf{Q}^*\mathbf{G}$. Other decompositions may follow more naturally from the underlying mathematical model.

When the the Hermitian part $\mathbf{H}$ is nonsingular, it can be used to precondition the nearly-Hermitian system., i.e., we solve a linear system with the preconditioned system,

$$\mathbf{H}^{-1}\mathbf{A} = \mathbf{I} + \mathbf{H}^{-1}\mathbf{F}\mathbf{C}\mathbf{F}^{*}. \tag{3.23}$$

This new system is a rank-s modification of the identity.

Proposition 3.2: Let $\mathbf{N} = \mathbf{I} + \mathbf{M}$ be a rank-s modification of the identity, with $\text{Rank}(\mathbf{M}) = s$. Then $\mathbf{N}$ has an eigenvalue of 1 with multiplicity $n - s$ and at most $s + 1$ unique eigenvalues.

Proof. Let $\mathbf{M} = \mathbf{P}\mathbf{J}\mathbf{P}^{-1}$ be the Jordan decomposition of $\mathbf{M}$. Then

$$\mathbf{N} = \mathbf{M} + \mathbf{I} = \mathbf{P}(\mathbf{J} + \mathbf{I})\mathbf{P}^{-1}$$

is the Jordan decomposition of $\mathbf{N}$. It is clear the for each zero eigenvalue of $\mathbf{M}$, $\mathbf{N}$ has an eigenvalue of 1, and the nonzero eigenvalues of $\mathbf{M}$ are shifted by 1 to give the eigenvalues of $\mathbf{N}$ different from 1. □

Therefore, GMRES applied to this system will require no more than $s + 1$ iterations to converge. This can easily be seen since at iteration $s + 1$, we will have constructed the dimension $s+1$ Krylov subspace, and the degree $s+1$ residual polynomial constructed at that step will be optimal over all degree $s+1$ polynomials, i.e., it will have roots at the, at most, $s+1$ distinct eigenvalues of $\mathbf{H}^{-1}\mathbf{A}$. This is the same philosophy underlying the CGW method [16, 100], except that $\mathbf{H}$ is assumed to be positive definite (we also say $\mathbf{A}$ is *real-positive*). To obtain the preconditioned system, we can compute $\mathbf{H}^{-1}\mathbf{F}$ and $\mathbf{H}^{-1}\mathbf{b}$ by solving $s + 1$ Hermitian linear systems using, e.g., MINRES [64] if $\mathbf{H}$ is indefinite, or conjugate gradient [45] if $\mathbf{H}$ is positive definite. These systems can be solved in sequence, in parallel, or using a block method, see e.g., [41, 62, 63]. Here we describe an alternative strategy, based on the Schur complement. This method also requires the solution of $s+1$ Hermitian systems followed by the solution of one $s \times s$ linear system. No GMRES iterations are required.

Note that (3.23) is the Schur complement of $-\mathbf{C}^{-1}$ in the matrix

$$\Phi = \begin{bmatrix} \mathbf{H} & \mathbf{F} \\ \mathbf{F}^* & -\mathbf{C}^{-1} \end{bmatrix},$$

so solving $\mathbf{A}\mathbf{x} = \mathbf{b}$ is equivalent to solving

$$\begin{bmatrix} \mathbf{H} & \mathbf{F} \\ \mathbf{F}^* & -\mathbf{C}^{-1} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} = \begin{bmatrix} \mathbf{b} \\ \mathbf{0} \end{bmatrix}. \tag{3.24}$$

Schur complement methods for solving systems of the form (3.24) are well known; see, e.g., [10, 17]. One such method eliminates $\mathbf{x}$ by inserting $\mathbf{x} = \mathbf{H}^{-1}(\mathbf{b} - \mathbf{F}\mathbf{y})$ into $\mathbf{F}^*\mathbf{x} - \mathbf{C}^{-1}\mathbf{y} = \mathbf{0}$, and solving for $\mathbf{y}$ through the s-dimensional system

$$\left(\mathbf{F}^*\mathbf{H}^{-1}\mathbf{F} + \mathbf{C}^{-1}\right)\mathbf{y} = \mathbf{F}^*\mathbf{H}^{-1}\mathbf{b}. \tag{3.25}$$

When the formulation (3.1) is more natural, this last equation takes the form

$$\left(\mathbf{G}^*\mathbf{H}^{-1}\mathbf{F} + 2\mathbf{I}\right)\mathbf{y} = \mathbf{G}^*\mathbf{H}^{-1}\mathbf{b}. \tag{3.26}$$

This approach is equivalent to applying the Sherman–Morrison–Woodbury formula to (3.23); see, e.g., [43, 77, 101].

The solution of (1.1) via the method just described requires the solution of

$$\mathbf{HW} = \mathbf{F} \quad \text{and} \quad \mathbf{Hu} = \mathbf{b}, \tag{3.27}$$

for $\mathbf{W} \in \mathbb{C}^{n\times s}$ and $\mathbf{u} \in \mathbb{C}^n$, as well as the $s \times s$ system (3.25) or (3.26) for $\mathbf{y}$. From these ingredients, one can construct $\mathbf{x} = \mathbf{u} - \mathbf{Wy}$. This approach is described in a more general setting in, e.g., [43, 102].

It is important to observe that $\mathbf{F}$ and $\mathbf{G}$ described in (3.1) are not unique. The low-rank skew Hermitian part of $\mathbf{A}$ has more than one such decomposition. In [102], it is shown that the condition number of the $s \times s$ matrix $(\mathbf{G}^*\mathbf{W} + 2\mathbf{I})$ can depend on the choice of $\mathbf{F}$ and $\mathbf{G}$. Furthermore, it is shown in [102, Theorem 1] that theoretically $\mathbf{F}$ and $\mathbf{G}$ always can be chosen so that the condition number $\kappa(\mathbf{G}^*\mathbf{H}^{-1}\mathbf{F} + 2\mathbf{I}) \leq \kappa(\mathbf{A})\kappa(\mathbf{H})$.

Since $\mathbf{H}$ is Hermitian, we can approximate the solution of the $s + 1$ systems (3.27) by three-term recurrence Lanczos iterations. Existing methods for solving an Hermitian system with several right-hand sides (see e.g., in [41, 63, 72, 81]) offer potential savings in time and computation.

Observe that restating the problem in this Schur complement framework has the effect of expressing the right-hand-side $\mathbf{b}$ as the sum of components in the range of $\mathbf{H}$ and in the range of $\mathbf{F}$. The vector $\mathbf{y}$ represents the coordinates of the component of $\mathbf{b}$ in the small-dimension range of $\mathbf{F}$ which allows us to adjust $\mathbf{b}$, we can express $\mathbf{x}$ as the solution to a corresponding Hermitian linear system. We can obtain $\mathbf{y}$ by directly solving $s + 1$ Hermitian systems and a small $s \times s$ system.

We now turn our attention to the development of a stopping criteria for our method. For simplicity, assume henceforth in this section that $\mathbf{F}$ is scaled so that

$\|\mathbf{F}\| = 1$, as is the case when $\mathbf{F}$ is derived from an economy-sized unitary diagonalization of the skew-Hermitian part of $\mathbf{A}$. Therefore $\|\mathbf{A} - \mathbf{A}^*\|/2 = \|\mathbf{FCF}^*\| \leq \|\mathbf{C}\|$. Let $\widetilde{\mathbf{W}}$ and $\widetilde{\mathbf{u}}$ denote approximate solutions to $\mathbf{HW} = \mathbf{F}$ and $\mathbf{Hu} = \mathbf{b}$ derived, e.g., from MINRES. With these approximations in hand, one would replace (3.25) with the perturbed s-dimensional system

$$(\mathbf{F}^*\widetilde{\mathbf{W}} + \mathbf{C}^{-1})\widetilde{\mathbf{y}} = \mathbf{F}^*\widetilde{\mathbf{u}}, \tag{3.28}$$

which can be solved directly with Gaussian elimination to yield the approximation

$$\widetilde{\mathbf{x}} = \widetilde{\mathbf{u}} - \widetilde{\mathbf{W}}\widetilde{\mathbf{y}}$$

to the desired solution $\mathbf{x}$. (For purposes of this analysis, we implicitly assume that this direct solve is computed exactly.) The residual of this approximation can be expressed in terms of the other approximations:

$$\begin{aligned}
\widetilde{\mathbf{r}} &:= \mathbf{b} - \mathbf{A}\widetilde{\mathbf{x}} \\
&= \mathbf{b} - (\mathbf{H} + \mathbf{FCF}^*)(\widetilde{\mathbf{u}} - \widetilde{\mathbf{W}}\widetilde{\mathbf{y}}) \\
&= \mathbf{b} - \mathbf{H}\widetilde{\mathbf{u}} + \mathbf{H}\widetilde{\mathbf{W}}\widetilde{\mathbf{y}} - \mathbf{FC}(\mathbf{F}^*\widetilde{\mathbf{u}} - \mathbf{F}^*\widetilde{\mathbf{W}}\widetilde{\mathbf{y}}) \\
&= \mathbf{b} - \mathbf{H}\widetilde{\mathbf{u}} + (\mathbf{H}\widetilde{\mathbf{W}} - \mathbf{F})\widetilde{\mathbf{y}}.
\end{aligned}$$

Basic norm inequalities yield a simple, dynamic stopping criterion.

Theorem 3.3: Suppose $\mathbf{A} \in \mathbb{C}^{n\times n}$ is a nonsingular matrix of the form (3.23) with $\|\mathbf{F}\| \leq 1$ and nonsingular Hermitian part $\mathbf{H}$. Let $\widetilde{\mathbf{W}}$ and $\widetilde{\mathbf{u}}$ be approximate solutions to $\mathbf{HW} = \mathbf{F}$ and $\mathbf{Hu} = \mathbf{b}$ with residuals $\mathbf{R}_{\mathbf{W}} := \mathbf{F} - \mathbf{H}\widetilde{\mathbf{W}}$ and $\mathbf{r}_{\mathbf{u}} := \mathbf{b} - \mathbf{H}\widetilde{\mathbf{u}}$, and suppose further that $\mathbf{F}^*\widetilde{\mathbf{W}} + \mathbf{C}^{-1}$ is nonsingular. Then provided

$$\frac{\|\mathbf{R}_\mathbf{W}\|\|\widetilde{\mathbf{y}}\|}{\|\mathbf{b}\|} < \varepsilon/2 \quad \text{and} \quad \frac{\|\mathbf{r}_\mathbf{u}\|}{\|\mathbf{b}\|} < \varepsilon/2,$$

and $\widetilde{\mathbf{y}}$ exactly solves (3.28), then the approximate solution $\widetilde{\mathbf{x}} := \widetilde{\mathbf{u}} - \widetilde{\mathbf{W}}\widetilde{\mathbf{y}}$ satisfies

$$\frac{\|\mathbf{b} - \mathbf{A}\widetilde{\mathbf{x}}\|}{\|\mathbf{b}\|} < \varepsilon. \tag{3.29}$$

Since $\widetilde{\mathbf{y}}$ depends on $\widetilde{\mathbf{W}}$, the stopping criterion for $\widetilde{\mathbf{W}}$ (i.e., $\|\mathbf{R}_\mathbf{W}\|\|\widetilde{\mathbf{y}}\|/\|\mathbf{b}\| < \varepsilon/2$) in Theorem 3.3 cannot be expressed *a priori*. Once a candidate value for $\widetilde{\mathbf{W}}$ has been found, one can solve the small $s \times s$ system (3.28) for $\widetilde{\mathbf{y}} \in \mathbb{C}^s$, where $s \ll n$.[2] With $\widetilde{\mathbf{y}}$ in hand, one can check if $\widetilde{\mathbf{W}}$ satisfies the stopping criterion; if not, conduct further MINRES iterations to refine $\widetilde{\mathbf{W}}$, and test the criterion again with the updated $\widetilde{\mathbf{y}}$.

Adapting notation slightly, let $\widetilde{\mathbf{W}}$ and $\widehat{\mathbf{W}}$ denote two approximate solutions to $\mathbf{HW} = \mathbf{F}$ with corresponding solutions $\widetilde{\mathbf{y}}$ and $\widehat{\mathbf{y}}$ to (3.28). The following result quantifies the rate at which $\widehat{\mathbf{y}} \to \widetilde{\mathbf{y}}$ as $\widehat{\mathbf{W}} \to \widetilde{\mathbf{W}}$, thus emphasizing that the dynamic stopping criterion supplied by Theorem 3.3 is stable with respect to refinements to $\widetilde{\mathbf{W}}$.

Theorem 3.4: Let $\widetilde{\mathbf{y}}, \widehat{\mathbf{y}} \in \mathbb{C}^s$ solve the nonsingular linear systems

$$\begin{aligned}(\mathbf{F}^*\widetilde{\mathbf{W}} + \mathbf{C}^{-1})\widetilde{\mathbf{y}} &= \mathbf{F}^*\widetilde{\mathbf{u}} \\ (\mathbf{F}^*\widehat{\mathbf{W}} + \mathbf{C}^{-1})\widehat{\mathbf{y}} &= \mathbf{F}^*\widetilde{\mathbf{u}},\end{aligned}$$

[2]In the case of the one-dimensional scattering problem in the last section, $s = 2$.

with $\|\mathbf{F}\| \leq 1$. Then for sufficiently small $\|\widetilde{\mathbf{W}} - \widehat{\mathbf{W}}\|$ and $\widetilde{\mathbf{\Omega}} := \mathbf{F}^*\widetilde{\mathbf{W}} + \mathbf{C}^{-1}$,

$$\|\widetilde{\mathbf{y}} - \widehat{\mathbf{y}}\| \leq \frac{\|\widetilde{\mathbf{\Omega}}^{-1}\| \, \|\widetilde{\mathbf{W}} - \widehat{\mathbf{W}}\| \, \|\widetilde{\mathbf{y}}\|}{1 - \|\widetilde{\mathbf{\Omega}}^{-1}\| \, \|\widetilde{\mathbf{W}} - \widehat{\mathbf{W}}\|},$$

and

$$\|\widehat{\mathbf{y}}\| \leq \|\widetilde{\mathbf{y}}\| + \frac{\|\widetilde{\mathbf{\Omega}}^{-1}\| \, \|\widetilde{\mathbf{W}} - \widehat{\mathbf{W}}\| \, \|\widetilde{\mathbf{y}}\|}{1 - \|\widetilde{\mathbf{\Omega}}^{-1}\| \, \|\widetilde{\mathbf{W}} - \widehat{\mathbf{W}}\|}.$$

Proof Since the difference between the coefficient matrix in solving for $\widetilde{\mathbf{y}}$ and $\widehat{\mathbf{y}}$ is bounded in norm by $\|\widehat{\mathbf{W}} - \widetilde{\mathbf{W}}\|$, the result follows directly from basic perturbation theory for linear systems [46, Thm. 7.2], provided $\|\widetilde{\mathbf{\Omega}}^{-1}\| \, \|\widetilde{\mathbf{W}} - \widehat{\mathbf{W}}\| < 1$. □

Recasting $\mathbf{A}\mathbf{x} = \mathbf{b}$ into $\mathbf{H}\mathbf{x} + \mathbf{F}\mathbf{y} = \mathbf{b}$ and $\mathbf{C}\mathbf{F}^*\mathbf{x} = \mathbf{y}$ expresses the right-hand side vector $\mathbf{b}$ as the sum of vectors in $\mathrm{Ran}(\mathbf{H})$ and $\mathrm{Ran}(\mathbf{F})$. An *a priori* bound for $\|\mathbf{y}\|$ or $\|\widetilde{\mathbf{y}}\|$ would thus require knowledge of quantities such as $\|\mathbf{A}^{-1}\|$ or $\|\mathbf{H}^{-1}\|$, both of which are computationally prohibitive, particularly compared to the cost of evaluating the dynamic bound in Theorem 3.3. We provide such a bound mainly for theoretical interest.

Theorem 3.5: Suppose $\|\mathbf{F}\| \leq 1$, and that $\mathbf{y}$ and $\widetilde{\mathbf{y}}$ solve

$$\mathbf{\Omega}\mathbf{y} = \mathbf{F}^*\mathbf{u}, \qquad \widetilde{\mathbf{\Omega}}\widetilde{\mathbf{y}} = \mathbf{F}^*\widetilde{\mathbf{u}},$$

for nonsingular $\mathbf{\Omega} := \mathbf{F}^*\mathbf{W} + \mathbf{C}^{-1}$ and $\widetilde{\mathbf{\Omega}} := \mathbf{F}^*\widetilde{\mathbf{W}} + \mathbf{C}^{-1}$. If $\|\mathbf{H}^{-1}\| \, \|\mathbf{\Omega}^{-1}\| \, \|\mathbf{R}_{\mathbf{W}}\| < 1$, then

$$\|\widetilde{\mathbf{y}}\| \quad \leq \quad \|\mathbf{\Omega}^{-1}\| \, \|\mathbf{H}^{-1}\| \left(\frac{\|\mathbf{b}\| + \|\mathbf{r}_{\mathbf{u}}\|}{1 - \|\mathbf{H}^{-1}\| \, \|\mathbf{\Omega}^{-1}\| \, \|\mathbf{R}_{\mathbf{W}}\|} \right),$$

where $\mathbf{R_W} := \mathbf{F} - \mathbf{H}\widetilde{\mathbf{W}}$ and $\mathbf{r_u} := \mathbf{b} - \mathbf{H}\widetilde{\mathbf{u}}$.

Proof The perturbation bound [46, Thm. 7.2] implies that

$$\begin{aligned}
\|\widetilde{\mathbf{y}}\| &\leq \|\mathbf{y}\| + \|\widetilde{\mathbf{y}} - \mathbf{y}\| \\
&\leq \|\mathbf{y}\| + \frac{\|\mathbf{H}^{-1}\|\,\|\mathbf{\Omega}^{-1}\|}{1 - \|\mathbf{H}^{-1}\|\,\|\mathbf{\Omega}^{-1}\|\,\|\mathbf{R_W}\|}\left(\|\mathbf{r_u}\| + \|\mathbf{R_W}\|\,\|\mathbf{y}\|\right), \qquad (3.30)
\end{aligned}$$

since $\|\mathbf{\Omega} - \widetilde{\mathbf{\Omega}}\| \leq \|\mathbf{H}^{-1}\|\,\|\mathbf{R_W}\|$ and $\|\mathbf{F}^*\mathbf{u} - \mathbf{F}^*\widetilde{\mathbf{u}}\| \leq \|\mathbf{H}^{-1}\|\,\|\mathbf{r_u}\|$. The fact that $\mathbf{y} = \mathbf{\Omega}^{-1}\mathbf{F}^*\mathbf{u} = \mathbf{\Omega}^{-1}\mathbf{F}^*\mathbf{H}^{-1}\mathbf{b}$ implies $\|\mathbf{y}\| \leq \|\mathbf{\Omega}^{-1}\|\,\|\mathbf{H}^{-1}\|\,\|\mathbf{b}\|$, from which follows

$$\begin{aligned}
\|\widetilde{\mathbf{y}}\| &\leq \|\mathbf{\Omega}^{-1}\|\,\|\mathbf{H}^{-1}\|\left(\|\mathbf{b}\| + \frac{\|\mathbf{r_u}\| + \|\mathbf{\Omega}^{-1}\|\,\|\mathbf{R_W}\|\,\|\mathbf{H}^{-1}\|\,\|\mathbf{b}\|}{1 - \|\mathbf{H}^{-1}\|\,\|\mathbf{\Omega}^{-1}\|\,\|\mathbf{R_W}\|}\right) \\
&= \|\mathbf{\Omega}^{-1}\|\,\|\mathbf{H}^{-1}\|\left(\frac{\|\mathbf{b}\| + \|\mathbf{r_u}\|}{1 - \|\mathbf{H}^{-1}\|\,\|\mathbf{\Omega}^{-1}\|\,\|\mathbf{R_W}\|}\right). \qquad \square
\end{aligned}$$

Corollary 3.6: Provided the approximate solutions $\widetilde{\mathbf{u}}$ and $\widetilde{\mathbf{W}}$ are sufficiently accurate that

$$\frac{\|\mathbf{r_u}\|}{\|\mathbf{b}\|} \leq \frac{\varepsilon}{2} \qquad \text{and} \qquad \|\mathbf{R_W}\| \leq \frac{\varepsilon}{\|\mathbf{\Omega}^{-1}\|\,\|\mathbf{H}^{-1}\|(2+2\varepsilon)},$$

the relative residual satisfies $\|\mathbf{r}\|/\|\mathbf{b}\| < \varepsilon$.

We emphasize that the quantity $\|\mathbf{\Omega}^{-1}\|$ in the hypothesis and bound renders this result inapplicable *a priori*, since $\mathbf{\Omega}$ is a function of $\mathbf{W}$. Approximating the value $\|\mathbf{\Omega}^{-1}\|$ from $\widetilde{\mathbf{W}}$ could give a dynamic estimate. Note that given an approximation $\widetilde{\mathbf{W}}$, one can directly compute $\|\widetilde{\mathbf{y}}\|$.

The Schur complement method for solving nearly Hermitian linear systems is summarized in Algorithm 3.3.1, which uses the dynamic stopping criterion. Since this convergence test involves $\widetilde{\mathbf{y}}$, which requires an approximation to $\mathbf{W}$, MINRES is

Algorithm 3.3.1: The Schur Complement Method

```
Input  : $\mathbf{A} \in \mathbb{C}^{n\times n}$, $\mathbf{F}, \mathbf{G} \in \mathbb{C}^{n\times s}$, $\mathbf{b} \in \mathbb{C}^n$, $tol > 0$, $m =$ maximum
         number of MINRES (or CG) iterations
Output: $\tilde{\mathbf{x}}$, an approximate solution to $\mathbf{Ax} = \mathbf{b}$

1  $\mathbf{H} \leftarrow (\mathbf{A} + \mathbf{A}^*)/2$
2  $\widetilde{\mathbf{u}} \leftarrow$ apply MINRES to $\mathbf{Hu} = \mathbf{b}$ with relative residual tolerance $tol/2$
   and initial iterate $\widetilde{\mathbf{u}}_0 = \mathbf{0}$ (or CG if $\mathbf{H}$ is positive definite)
3  for $k = 1$ to $m$ do
4      for $j = 1$ to $s$ do
5          $\mathbf{f} \leftarrow j$th column of $\mathbf{F}$
6          $\widetilde{\mathbf{w}}_j \leftarrow \operatorname{argmin}_{\widehat{\mathbf{w}} \in \mathcal{K}_k(\mathbf{H},\mathbf{f})} \|\mathbf{f} - \mathbf{H}\widehat{\mathbf{w}}\|$ via MINRES (or CG if $\mathbf{H}$ is
           positive definite, using $\|\cdot\| = \|\cdot\|_{\mathbf{H}}$)
7      if $\|\mathbf{F} - \mathbf{H}\widetilde{\mathbf{W}}\| < tol/2$ then
8          $\widetilde{\mathbf{y}} \leftarrow (\mathbf{F}^*\widetilde{\mathbf{W}} + \mathbf{C}^{-1})^{-1}\mathbf{F}^*\mathbf{u}$
9          if $\|\mathbf{F} - \mathbf{H}\widetilde{\mathbf{W}}\| \, \|\widetilde{\mathbf{y}}\|/\|\mathbf{b}\| < tol/2$ then
10             End For Loop
11 $\widetilde{\mathbf{x}} \leftarrow \widetilde{\mathbf{u}} - \widetilde{\mathbf{W}}\widetilde{\mathbf{y}}$
```

applied to the problems $\mathbf{Hw}_j = \mathbf{f}_j$ for $j = 1, \ldots, s$ concurrently. Lines 4 through 6 of the algorithm can be replaced by a step-by-step block MINRES [63]. (Block methods for multiple right-hand sides can potentially give more rapid convergence than the aggregate cost of solving the systems one at a time [41, 63].) Also observe that it is not necessary to build the matrix $\mathbf{H}$; one need only compute the matrix-vector product $\mathbf{Hx}$ for a vector $\mathbf{x}$.

There is no guarantee that $\mathbf{H}$ is well-conditioned, or even invertible (even when $\mathbf{A}$ is well conditioned: indeed, $\mathbf{H}$ is singular for the examples constructed in Section 3.2.5). Furthermore, $\mathbf{F}^*\mathbf{W} + \mathbf{C}^{-1}$ can be singular, in which case accurate approximations $\widetilde{\mathbf{W}}$ give matrices $\mathbf{F}^*\widetilde{\mathbf{W}} + \mathbf{C}^{-1}$ that are highly ill-conditioned. For such cases, Algorithm 3.3.1 would likely be unsuitable. While Algorithm 3.3.1 does offer a method for approximating the solution to (1.1) that requires only three-term recurrences, it is not more efficient than the PGMRES method, provided the latter method does not suffer from the instabilities identified in Section 3.2.3. When such instabilities occur, Algorithm 3.3.1 provides an appealing alternative to full GMRES (with its long recurrences) or other methods based on short recurrences that do not satisfy any natural optimality properties.

3.3.2 Schur Complement Method as a Preconditioner

Given the utility of Algorithm 3.3.1 for solving linear systems for which the coefficient matrix has the form (3.1), one naturally wonders whether this Schur complement strategy can be used as a preconditioner for matrices that are close to the form (3.1). Apply $\mathbf{M}$ as a right preconditioner by modifying (1.1) to have the form

$$\mathbf{A}\mathbf{M}^{-1}\mathbf{z} = \mathbf{b}, \quad \mathbf{x} = \mathbf{M}^{-1}\mathbf{z}, \tag{3.31}$$

where $\mathbf{M}$ is a matrix that is cheap to compute, and makes $\mathbf{A}\mathbf{M}^{-1}$ more favorable for GMRES convergence than $\mathbf{A}$ on its own. A right-preconditioned Krylov subspace method selects the jth approximation from the affine subspace

$$\mathbf{x}_j \in \mathbf{x}_0 + \operatorname{span}\{\mathbf{r}_0, \mathbf{A}\mathbf{M}^{-1}\mathbf{r}_0, \ldots, \left(\mathbf{A}\mathbf{M}^{-1}\right)^{j-1}\mathbf{r}_0\},$$

see, e.g., [73],[75, Chapter 9],[85].

Suppose $\mathbf{A} \in \mathbb{C}^{n\times n}$ has the decomposition $\mathbf{A} = \mathbf{H} + \mathbf{K}$, where $\mathbf{H}$ is the Hermitian part, and the skew-Hermitian part $\mathbf{K}$ an be well-approximated by $\mathbf{F}\mathbf{G}^*$ for some $\mathbf{F}, \mathbf{G} \in \mathbb{C}^{n\times s}$ with $s \ll n$, e.g., $\mathbf{K} = \mathbf{F}\mathbf{G}^* + \mathbf{E}$ with $\|\mathbf{E}\| \ll 1$. Systems such as this arise, for example, in discretizations of PDEs with certain Neumann boundary conditions, or more generically when $\mathbf{K}$ has a small number of dominant singular values. In such instances, we expect $\mathbf{M} = \mathbf{H} + \mathbf{F}\mathbf{G}^*$ to be an effective preconditioner for $\mathbf{A}$, while allowing for rapid application through Algorithm 3.3.1.

Recall that Algorithm 3.3.1 requires $s + 1$ applications of MINRES, which could be prohibitive to apply at each iteration of GMRES applied to the preconditioned system. However, s of these applications are needed to solve $\mathbf{HW} = \mathbf{F}$; the solution of this system can be computed once and reused at each GMRES iteration. Using $\mathbf{v}_j$ here to denote the jth Arnoldi vector in the (outer) preconditioned GMRES iteration, each preconditioner application $\mathbf{M}^{-1}\mathbf{v}_j$ will only require the solution of one Hermitian system, $\mathbf{H}\widehat{\mathbf{x}} = \mathbf{v}_j$, and the direct solution of one $s \times s$ system. Thus this preconditioner is relatively cheap to compute at each step, after the up-front cost of solving $\mathbf{HW} = \mathbf{F}$.

3.3.3 Schur Complement Method Numerical Results

To demonstrate the effectiveness of the proposed Schur complement approach, we apply Algorithm 3.3.1 to several problems. To put this new algorithm in context, we also solve our linear system using MATLAB's full GMRES algorithm [55] and the recently proposed IDR(2) method [86, 87], as implemented by van Gijzen [97] which we have previously mentioned in Section 2.3. The IDR(2) method is a modern short recurrence method that uses right and left Krylov subspaces. In the numerical examples in this section, the Schur Complement Method is implemented using the MINRES algorithm described in [28, Sec. 6.5] (which is considerably more expedient than MATLAB's implementation in our experiments). In most of our experiments we apply the MINRES runs in series, but the Schur complement algorithm allows for easy parallelization of these calls, since each MINRES run can be done independently. In fact, we show one parallel numerical experiment in Section 3.3.6. When PGMRES is successful, it is often the most efficient algorithm.

3.3.4 Simple example, revisited

In Section 3.2.5, we introduced a simple, well-conditioned matrix (3.14) with a skew-Hermitian part of low rank. The analysis presented in Section 3.2.6 suggests that as the norm of the skew-Hermitian part grows, the numerical instability of PGMRES causes the residual to stagnate earlier. To apply Algorithm 3.3.1, however, we require that the Hermitian part of $\mathbf{A}$ be nonsingular; thus for the tests in this section, we modify (3.14) by adding a 2×2 identity to the $\mathbf{Z}$ block (thus shifting the zero eigenvalues of the Hermitian part to one). As is clear from Figure 3.4, this does not change the convergence behavior discussed in Section 3.2.5. It follows from Theorem 3.3 that

$$\gamma_k := \|\mathbf{r}_{\mathbf{b},k}\| + \sqrt{2}\|\mathbf{y}\|(\|\mathbf{r}_{\mathbf{f}_1,k}\| + \|\mathbf{r}_{\mathbf{f}_2,k}\|) \tag{3.32}$$

is an upper bound on the norm of the residual produced at each iteration of the Schur Complement Method. The vectors $\mathbf{r}_{\mathbf{b},k}$, $\mathbf{r}_{\mathbf{f}_1,k}$, and $\mathbf{r}_{\mathbf{f}_2,k}$ are the residuals produced by MINRES with coefficient matrix $\mathbf{H} = (\mathbf{A} + \mathbf{A}^*)/2$ and right-hand sides $\mathbf{b}$, $\mathbf{f}_1$, and $\mathbf{f}_2$. To simplify the illustration of our numerical results in Figure 3.4, we plot γ_k versus k for the Schur complement approach. Observe that this method is superior in run time to full GMRES, PGMRES, and IDR(2), and competitive or superior in iteration count as well. (Note that γ_k can be smaller than the full GMRES residual

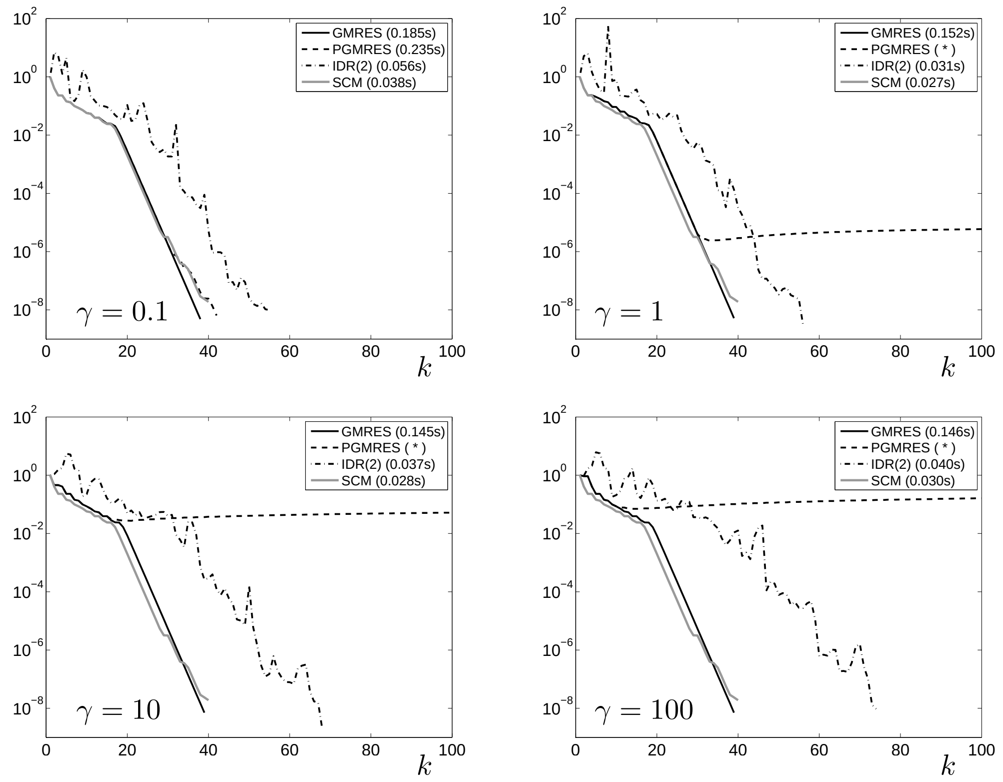


Figure 3.4. Convergence of the Schur Complement Method for $\mathbf{Ax} = \mathbf{b}$, with coefficient matrix of dimension $n = 10^4$ given by the simple example (3.14) with a 2×2 identity added to the $\mathbf{Z}$ block, and parameters $\alpha = 0.125$, $\beta = 1$, and $p = 6$; the vector $\mathbf{b}$ is equal to $1/\sqrt{n}$ in all components. The gray line shows the upper bound γ_k in (3.32) on the residual norm. For context, the residual norms for full GMRES, PGMRES, and IDR(2) are also shown.

norm, since the Schur Complement Method does not draw its approximations from the same Krylov subspace from which GMRES draws its optimal iterates.) We chose IDR(2) (as opposed to, e.g., IDR(4)) to compare the Schur Complement Method to a modern short-term recurrence with a small storage requirement.

Whether the Schur Complement Method is better than preconditioning with the Hermitian part of the matrix depends on the structure and spectral properties

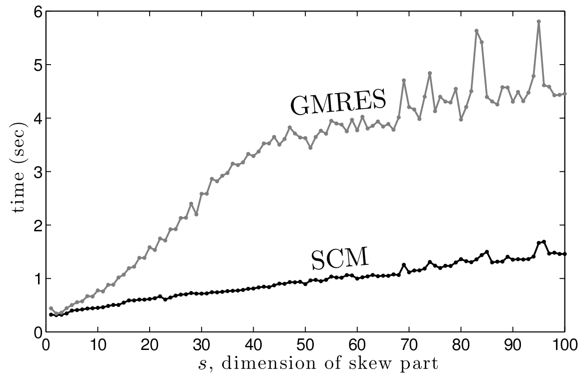


Figure 3.5. Comparison of the Schur Complement Method to GMRES (with Hermitian preconditioning) for the simple example (3.14) of dimension $n = 10^5$ with skew-Hermitian part of rank s. The eigenvalues of the Hermitian-preconditioned matrix fill the interval $[1 - 2i, 1 + 2i]$, so the condition number of this matrix is essentially fixed for all these experiments.

of the preconditioned matrix. To illustrate the potential superiority of the Schur Complement Method, we alter the simple example to have an $s \times s$ skew-Hermitian block, with sub- and super-diagonal entries equal to 1 and -1. The eigenvalues of the preconditioned matrix fill the complex interval $[1-2i, 1+2i]$ as s increases, thus keeping the conditioning of the preconditioned problem essentially fixed. The dimension of $\mathbf{A}$ is fixed at $n = 100{,}000$; the right-hand size vector is random. Figure 3.5 compares the CPU time of GMRES with Hermitian preconditioning to the Schur Complement Method, showing that the superiority of the latter algorithm improves as s grows.

3.3.5 Lippmann–Schwinger equation

Next, we apply Algorithm 3.3.1 to the Lippmann–Schwinger integral equation described in Section 3.2.4. Figure 3.6 shows the convergence of each MINRES application required by the Schur Complement Method, when applied to the integral equation with wave number $\kappa = 10$ and constant refractive index $m = -1$.

When the refractive index $m(x)$ varies in space, the skew-Hermitian part of the operator no longer has low rank. (Since the model describes wave scattering by a non-homogeneous obstacle, we assume throughout that $m(x) < 0$ for all $x \in (0,\, 2\pi)$.) In this case, let M denote the multiplication operator defined by $(Mu)(x) = m(x)u(x)$ on $L^2(0, 2\pi)$, and define the integral operator $K_0 : L^2(0, 2\pi) \to L^2(0, 2\pi)$ based on K in (3.13) but with m removed:

$$(K_0 u)(x) = \frac{i\kappa}{2} \int_0^{2\pi} e^{i\kappa|x-y|} u(y)\, \mathrm{d}y.$$

The integral equation (3.12) is then equivalent to

$$(M^{-1} + K_0)(Mu)(x) = u^i(x), \tag{3.33}$$

where $M^{-1} + K_0$ has the special structure required by Algorithm 3.3.1, since it is the sum of the self-adjoint operator M^{-1} with an operator K_0 that has a rank-2 skew-adjoint part. Indeed, we can view (3.33) as a "right preconditioned" version of (3.12), where the preconditioner is selected not to accelerate convergence, but rather to pose the problem in a way that allows for solution via a short recurrence.(For another form of such strategic preconditioning, see [29].)

We approximate this problem with the Nyström discretization used in Section 3.2.4, whereby the operator M is represented as a diagonal matrix $\mathbf{M}$ with diagonal entries equal to m evaluated at each of the quadrature points. Figure 3.6 (left) shows the convergence of the MINRES iterations when Algorithm 3.3.1 is applied to the discretization of (3.33) for $m(x) = -2 - \sin(x)$ with wave number $\kappa = 10$ and $n = 10^3$ quadrature points.

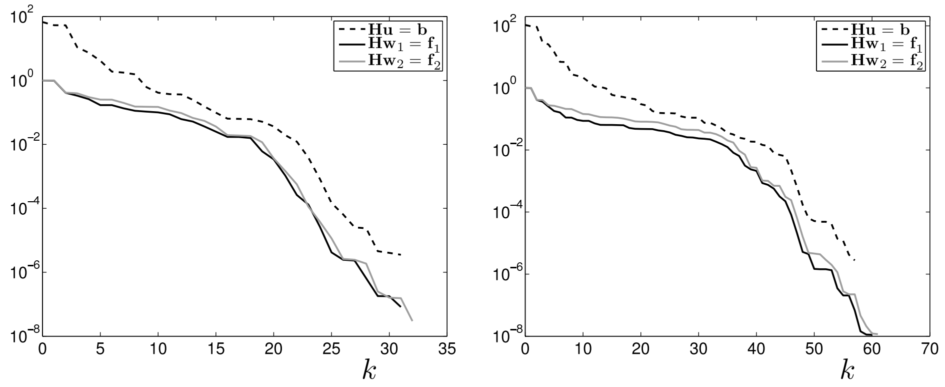


Figure 3.6. Residual norms for the MINRES iterations for the discretized Lippmann–Schwinger integral equation for wave number $\kappa = 10$ on a grid of $n = 10^3$ quadrature points with refractive indices $m(x) \equiv -1$ (left) and $m(x) = -2 - \sin(x)$ (right).

To put this approach in context, we also apply the Schur Complement Approach to an equation with constant refractive index $m(x) = -1$ for various wave numbers κ, and a discretization of $n = 10^3$ points. We also apply MATLAB's full GMRES method [55], IDR(2) [97], and PGMRES [8]. The times required to reach a tolerance of 10^{-10} are displayed in Table 3.1.

3.3.6 A parallel experiment

While our goal is to present the Schur Complement Method as a viable alternative to PGMRES and other GMRES-based methods, we also want to show that even for large problems, this approach can be competitive. In particular, here we test the performance of the Schur Complement Method in comparison both to restarted

Table 3.1. MATLAB's implementation of GMRES, PGMRES, van Gijzen's implementation of IDR(2), and the Schur Complement Method are applied to the discretized Lippmann–Schwinger integral operator for various wave numbers, using a discretization of $n = 10^3$ points and a right-hand side vector randomly generated by MATLAB's `randn` function. The timings are in seconds; a $*$ indicates that the method did not reduce the relative residual norm to 10^{-10} within 1000 iterations. Experiments were run on `Macbooki5.`

κ	GMRES	PGMRES	IDR(2)	SCM
1	0.0848	0.0296	0.0404	0.0680
2	0.0293	0.0199	0.0094	0.0134
3	0.0297	$*$	0.0102	0.0138
4	0.0349	$*$	0.0146	0.0203
5	0.0520	$*$	0.0171	0.0167
10	0.0872	$*$	0.0339	0.0314
20	0.1827	$*$	0.0744	0.0634
30	0.2925	$*$	0.1219	0.1096
40	0.4520	$*$	0.2019	0.1391
50	0.6308	$*$	0.3304	0.1976
60	0.8538	$*$	$*$	0.2585
70	1.1140	$*$	$*$	0.3441
80	1.3619	$*$	$*$	0.4231
90	1.7271	$*$	$*$	0.5279
100	2.0571	$*$	$*$	0.6596

GMRES and IDR(2), the latter requiring 11 vectors of storage. Recall Example 1 in the paper of Beckermann and Reichel [8], a Bratu path-following test problem:

$$\begin{aligned} -\Delta u(\mathbf{x}) - \lambda \exp(u(\mathbf{x})) &= u(\mathbf{x}), \quad \mathbf{x} \in \Omega, \\ u &= 0, \qquad \mathbf{x} \in \partial\Omega, \end{aligned}$$

where Ω is the unit square. We chose the same finite difference discretization and parameters as in the tests in [8], but used a grid of 500×500 points. This gives a matrix of order $n = 25 \times 10^4$ that is Hermitian plus a rank-2 skew-Hermitian modification. For this test, we implemented a parallelized version of the Schur Complement Method in which the three Hermitian solves are done in parallel using the looping control structure `parfor` from MATLAB's Parallel Computing Toolbox. In Table 3.2, we see that the parallel Schur Complement Method outperforms the other methods in run time, though it requires more matrix-vector products than any other method. This is a result of being able to perform the inner Hermitian iterations in parallel, with an average of 246 matrix-vector products per processor being performed simultaneously. Tests of the parallel Schur Complement Method were run on `Euler`.

3.3.7 Discussion and Conclusions

PGMRES cleverly exploits the structure of nearly Hermitian matrices to provide a short-term recurrence that is mathematically equivalent to GMRES. Unfortunately, our experiments illustrate that the method suffers from numerical instabilities that can cause the residual to stagnate, even for mildly conditioned coefficient matrices. The analysis presented in Section 3.2.6, as well as the computational experiments in

Table 3.2. Run time and matrix-vector product count for the parallelized Schur Complement Method, IDR(2) and GMRES(11), and GMRES(100) on the two-dimensional Bratu problem from [8], but discretized on a grid of 500×500 points. The convergence tolerance of the relative residual, $\|\mathbf{b} - \mathbf{A}\widetilde{\mathbf{x}}\|$, was 10^{-7}. All methods were preconditioned with an incomplete Cholesky factorization of the Hermitian part with drop tolerance 10^{-2}. For the Schur Complement method, there were three systems to solve, and they were done in parallel on three processors.

method	# matvecs	run time
Parallel SCM	661	38.9113
Serial SCM	661	86.9660
IDR(2)	340	57.4099
GMRES(11)	$*$	$*$
GMRES(100)	337	1061.7052

$*$ No convergence in a reasonable amount of time.

Section 3.2.5, demonstrate that instabilities occur when the the skew-Hermitian part is not small in norm. The instability corresponds to a severe loss of orthogonality of the Arnoldi vectors produced by PGMRES, so severe that the "basis" loses linear independence. This is consistent with the analysis presented in [39] for the standard GMRES algorithm.

As an alternative, we show that solving linear systems with a nearly-Hermitian coefficient matrix (3.1) is similar in structure to Schur complement problems. A Schur complement approach, with stopping criteria developed in Section 3.3.1, computes the solution to the original system by solving $s+1$ Hermitian linear systems and an $s \times s$ system (provided the Hermitian part is invertible). Since one can solve Hermitian linear systems using MINRES or conjugate gradients, this approach requires only three-term recursions (though poor conditioning of the Hermitian part can hamper convergence). The method is easily parallelizable and simple to implement.

3.4 An Implementation of Block MINRES

3.4.1 Background

In the course of developing the Schur Complement Method, we investigated the performance of the method when the $s+1$ Hermitian systems are solved using a block method (specifically block MINRES). We describe here an algorithm for solving a Hermitian linear system with multiple right hand sides,

$$\mathbf{AX} = \mathbf{B} \tag{3.34}$$

where the coefficient matrix $\mathbf{A} \in \mathbb{C}^{n\times n}$ is Hermitian indefinite, and $\mathbf{B} \in \mathbb{C}^{n\times p}$. We wish to extend the MINRES algorithm of Paige and Saunders [64] to the case when we have multiple right-hand sides.

There has been much work on the solution of linear systems with multiple right-hand sides for both the non-Hermitian, e.g., [41, 72], and Hermitian, cases, e.g., [62, 63, 81]. In many cases, extending the framework of a Krylov method to the block right-hand side setting involves generalizing the machinery of, e.g., the Arnoldi process to deal with block vectors, see, for example, [75, Page 208]. What was normalization in the case of a single vector becomes computing the QR-factorization of a block vector to get an orthonormal basis for its column space. For clarity, we will refer to methods that perform operations on block vectors as *true block Krylov methods*. At step j, true block methods generate a block Krylov subspace

$$\mathbb{K}_j(\mathbf{A}, \mathbf{V}_1) = \mathcal{K}_j(\mathbf{A}, \mathbf{v}_1^{(1)}) + \mathcal{K}_j(\mathbf{A}, \mathbf{v}_1^{(2)}) + \cdots + \mathcal{K}_j(\mathbf{A}, \mathbf{v}_1^{(p)})$$

spanned by the columns of

$$\mathbf{W}_j = \begin{bmatrix} \mathbf{V}_1, & \mathbf{V}_2, & \ldots & , \mathbf{V}_j \end{bmatrix},$$

where $\mathbf{V}_\ell = \begin{bmatrix} \mathbf{v}_\ell^{(1)}, & \mathbf{v}_\ell^{(2)}, & \ldots, & \mathbf{v}_\ell^{(p)} \end{bmatrix}$ for $\ell = 1, \ldots, j$. Let $\mathbf{H}_j = (\mathbf{H}_{i,\ell})$ be generated by the block Arnoldi process and $\mathbf{H}_{i,\ell} \in \mathbb{C}^{p \times p}$. As a consequence of the block normalization, $\mathbf{V}_\ell \in \mathbb{C}^{n \times p}$ has orthonormal columns and the columns in each block are orthonormal to the columns of the other blocks. Let $\mathbf{E}_j$ be the matrix that contains contains the last p columns of the dimension jp identity matrix. As in the single vector case, we have a block Arnoldi relation

$$\mathbf{A}\mathbf{W}_j = \mathbf{W}_j\mathbf{H}_j + \mathbf{V}_{j+1}\mathbf{H}_{j+1,j}\mathbf{E}_j^*.$$

To derive our algorithm, we will use the block Arnoldi algorithm proposed by Ruhe [71] and described in [75, Page 209]. For this method, we adopt the notation that $\mathbf{U}_p = \begin{bmatrix} \mathbf{v}_1, \ldots, \mathbf{v}_p \end{bmatrix}$ denotes the normalized starting block of vectors. Note that $\mathbf{U}_p$ was called $\mathbf{V}_1$ when describing the true block method. Essentially, Ruhe's method performs the same orthogonalization as the true block method, only one vector at a time. Therefore, at each iteration of Ruhe's block Arnoldi method, one matrix-single-vector product is performed as opposed to a matrix-block-vector product in a true block method. For review, we present Ruhe's block Arnoldi process as Algorithm 3.4.1, as described in [75, Page 209].

Algorithm 3.4.1: Ruhe's Block Arnoldi Method

```
    Input  : $\mathbf{A} \in \mathbb{C}^{n\times n}$, $\mathbf{U}_p \in \mathbb{C}^{n\times p}$, $\mathbf{U}_p^*\mathbf{U}_p = \mathbf{I}_p$, $j > 0$
    Output: $\mathbf{U}_{p+j} \in \mathbb{C}^{n\times(m+p)}$, $\mathbf{U}_{p+j}^*\mathbf{U}_{p+j} = \mathbf{I}_{p+j}$ and $\overline{\mathbf{H}}_j \in \mathbb{C}^{(j+p)\times j}$, $\overline{\mathbf{H}}_j$ has
            $p$ lower subdiagonal entries
1   for $j = p : j + p - 1$ do
2       Set $k := j - p + 1$
3       Compute $\mathbf{w} := \mathbf{A}\mathbf{v}_k$ for $i = 1 : j$ do
4           $h_{i,k} := \mathbf{v}_i^*\mathbf{w}$
5           $\mathbf{w} \leftarrow \mathbf{w} - h_{i,k}\mathbf{v}_i$
6       Compute $h_{j+1,k} := \|\mathbf{w}\|_2$ and $\mathbf{v}_{j+1} := \mathbf{w}/h_{j+1,k}$
```

We can observe that iteration j where $j = \ell p$ is a multiple of the block size, Algorithm 3.4.1 has produced the same orthonormal basis as the true block Arnoldi method at step ℓ. We also have Ruhe's block Arnoldi relation

$$\mathbf{A}\mathbf{U}_j = \mathbf{U}_{j+p}\overline{\mathbf{H}}_j. \tag{3.35}$$

Observe that $\overline{\mathbf{H}}_j$ has p lower subdiagonal bands and has the structure

$$\overline{\mathbf{H}}_j = \begin{bmatrix} \mathbf{H}_j \\ \mathbf{H}_{p\times j} \end{bmatrix}$$

where $\mathbf{H}_j$ is a square $j \times j$ matrix which is banded lower subdiagonal with p bands. Observe that $\mathbf{H}_{p\times j}$ only has nonzero entries in the last p columns with structure

$$\mathbf{H}_{p\times j} = \begin{bmatrix} \mathbf{0}_{p\times(m-p)} & \mathbf{C}_j \end{bmatrix}$$

where $\mathbf{C}_j \in \mathbb{C}^{p\times p}$ is upper triangular. At step j, we will use the following notation to describe the space we have constructed. We can always write $j = kp + m$ where $0 < j < p$. The subspace that has been generated at step j is

$$\mathbb{K}_{k,j}(\mathbf{A}, \mathbf{U}_p) = \mathcal{K}_{k+1}(\mathbf{A}, \mathbf{v}_1) + \cdots + \mathcal{K}_{k+1}(\mathbf{A}, \mathbf{v}_j) + \mathcal{K}_k(\mathbf{A}, \mathbf{v}_{j+1}) + \cdots + \mathcal{K}_k(\mathbf{A}, \mathbf{v}_p) \tag{3.36}$$

Similar to the Hermitian Lanczos relation in the case of a normal Krylov method, observe that if $\mathbf{A}$ is Hermitian, the relation

$$\mathbf{H}_j = \mathbf{U}_j^* \mathbf{A} \mathbf{U}_j$$

implies that $\mathbf{H}_j$ is also Hermitian. Due to the banded lower subdiagonal structure of $\mathbf{H}_j$, we see that $\mathbf{H}_j$ is $2p+1$ banded matrix with p superdiagonal entries and p subdiagonal entries. This structure implies that we only need the previous $2p$ basis vectors of $\mathbb{K}_{k,j}(\mathbf{A}, \mathbf{U}_p)$ in order to compute $\mathbf{v}_{j+1}$. We have the $2p+1$ term recurrence relation

$$\mathbf{h}_{j+p,j} \mathbf{v}_{j+1} = \mathbf{A}\mathbf{v}_j - \sum_{\ell=\min(1,m-p)}^{m+p-1} h_{j,\ell} \mathbf{v}_\ell \tag{3.37}$$

Due to symmetry, we do not need to compute $h_{i,j}$ where $i < j$ since it was computed previously as $\overline{h_{j,i}}$. This will require the storage of the lower subdiagonal entries of the last p columns of $\overline{\mathbf{H}}_j$. This yields Ruhe's block Lanczos method.

3.4.2 A Block Minimum Residual Method

We now derive a minimal residual algorithm based on Ruhe's block Lanczos method. We take a few moments to describe what is meant by a block minimum residual algorithm. At the jth step the following method will produce an approximation $\mathbf{X}_j$ such that for each $0 < i \leq p$, the residual $\left\|\mathbf{B}^{(i)} - \mathbf{A}\mathbf{X}_j^{(i)}\right\|$ is minimized over the space $\mathbb{K}_{k,j}(\mathbf{A}, \mathbf{B})$. Furthermore, for $\mathbf{Y} \in \mathbb{C}^{n \times p}$ let $\|\mathbf{Y}\|_{col} = \max_{0<i\leq p} \left\|\mathbf{Y}^{(i)}\right\|$ be the maximum of the individual norms of all columns of $\mathbf{Y}$. It is straightforward to show that $\|\cdot\|_{col}$ is a norm using the properties of $\|\cdot\|$. We can also say that $\left\|\mathbf{B}^{(i)} - \mathbf{A}\mathbf{X}_j^{(i)}\right\|_{col}$ will be minimized over $\mathbb{K}_{k,j}(\mathbf{A}, \mathbf{B})$. We use this norm only for simplicity of discussion in developing the algorithm.

At step j, we want to minimize each column of the block residual $\mathbf{F}_j$. Following the explanation of MINRES presented in [38], we can derive a block MINRES algorithm based on Ruhe's block Lanczos process. Let $\mathbf{E}_1^{(j)} \in \mathbb{R}^{(j+p)\times p}$ be the matrix containing the first p columns of the $(j+p)$-dimensional identity matrix. Observe that

$$\mathbf{E}_1^{(j)} = \begin{bmatrix} \mathbf{E}_1^{(j-1)} \\ \mathbf{0}_{1\times p} \end{bmatrix}. \tag{3.38}$$

Given an initial residual we can normalize it using the QR factorization $\mathbf{F}_0 = \mathbf{V}_p\mathbf{S}$. At step j of Ruhe's block Lanczos process, we have the QR factorization $\overline{\mathbf{H}}_j = \mathbf{Q}_j\overline{\mathbf{R}}_j$ is such that $\mathbf{Q}_j \in \mathbb{C}^{(j+p)\times(m+p)}$ is unitary, and $\overline{\mathbf{R}}_j \in \mathbb{C}^{(j+p)\times j}$. The matrix $\overline{\mathbf{R}}_j$ has a simple block structure,

$$\overline{\mathbf{R}}_j = \begin{bmatrix} \mathbf{R}_j \\ \mathbf{0}_{p\times j} \end{bmatrix},$$

where $\mathbf{R}_j$ is a square, upper triangular, $j \times m$ matrix. The minimization of $\|\mathbf{F}_j\|$ can be rewritten as

$$\begin{aligned} \|\mathbf{F}_j\| &= \min_{\mathbf{X}\in\mathbf{X}_0+\mathbb{K}_{k,j}(\mathbf{A},\mathbf{B})} \|\mathbf{B}-\mathbf{A}\mathbf{X}\|_{col} \\ &= \min_{\mathbf{Y}\in\mathbb{R}^{m\times p}} \|\mathbf{F}_0 - \mathbf{A}\mathbf{U}_j\mathbf{Y}\|_{col} \\ &= \min_{\mathbf{Y}\in\mathbb{R}^{m\times p}} \|\mathbf{U}_p\mathbf{S} - \mathbf{A}\mathbf{U}_j\mathbf{Y}\|_{col} \\ &= \min_{\mathbf{Y}\in\mathbb{R}^{m\times p}} \left\|\mathbf{U}_{j+p}\mathbf{E}_1^{(j)}\mathbf{S} - \mathbf{U}_{j+p}\overline{\mathbf{H}}_j\mathbf{Y}\right\|_{col} \\ &= \min_{\mathbf{Y}\in\mathbb{R}^{m\times p}} \left\|\mathbf{E}_1^{(j)}\mathbf{S} - \overline{\mathbf{H}}_j\mathbf{Y}\right\|_{col} \\ &= \min_{\mathbf{Y}\in\mathbb{R}^{m\times p}} \left\|\mathbf{Q}_j^*\mathbf{E}_1^{(j)}\mathbf{S} - \overline{\mathbf{R}}_j\mathbf{Y}\right\|_{col}. \end{aligned} \tag{3.39}$$

The solution of this least squares problem can be computed by solving the normal equations,

$$\begin{aligned}
\overline{\mathbf{R}}_j^*\mathbf{Q}_j^*\mathbf{E}_1^{(j)}\mathbf{S} &= \mathbf{R}_j^*\mathbf{R}_j\mathbf{Y}_j \qquad (3.40)\\
\mathbf{R}_j^{-*}\overline{\mathbf{R}}_j^*\mathbf{Q}_j^*\mathbf{E}_1^{(j)}\mathbf{S} &= \mathbf{R}_j\mathbf{Y}_j\\
\begin{bmatrix}\mathbf{I}_j & \mathbf{0}_{k\times p}\end{bmatrix}\mathbf{Q}_j^*\mathbf{E}_1^{(j)}\mathbf{S} &= \mathbf{R}_j\mathbf{Y}_j\\
(\mathbf{Q}_j^*\mathbf{E}_1^{(j)}\mathbf{S})_{1:j} &= \mathbf{R}_j\mathbf{Y}_j.
\end{aligned}$$

Thus we have that $\mathbf{Y}_j = \mathbf{R}_j^{-1}(\mathbf{Q}_j^*\mathbf{E}_1^{(j)}\mathbf{S})_{1:j}$, and the minimum residual approximation at step j is

$$\begin{aligned}
\mathbf{X}_j &= \mathbf{X}_0 + \mathbf{U}_j\mathbf{Y}_j \qquad (3.41)\\
&= \mathbf{X}_0 + \mathbf{U}_j\mathbf{R}_j^{-1}(\mathbf{Q}_j^*\mathbf{E}_1^{(j)}\mathbf{S})_{1:j}.
\end{aligned}$$

Let $\overline{\mathbf{T}}_j = \mathbf{Q}_j^*\mathbf{E}_1^{(j)}\mathbf{S}$. We define our search directions as the columns of $\mathbf{M}_j = \mathbf{U}_j\mathbf{R}_j^{-1}$ and observe that the columns of $\mathbf{M}_j$ successively span the same subspaces as the columns of $\mathbf{U}_j$. We denote block vector of search direction coordinates $\mathbf{T}_j = (\overline{\mathbf{T}}_j)_{1:j}$.

3.4.3 Block MINRES for Hermitian Linear Systems

To obtain a storage-efficient block MINRES algorithm based on Ruhe's block Lanczos method, we must discuss the structure of $\mathbf{R}_j$. This matrix is the upper $j \times j$ block of $\overline{\mathbf{R}}_j$ which is obtained from the QR-factorization of $\overline{\mathbf{H}}_j$. We can obtain this factorization column by column using Givens rotations to annihilate the subdiagonal entries of each column. Recall that for column pair

$$\mathbf{h} = \begin{bmatrix} h_{i,j} \\ h_{i+1,j} \end{bmatrix},$$

to annihilate $h_{i+1,j}$, we can construct the Givens sine/cosine pair

$$s_{i+1,j} = \frac{h_{i+1,j}}{\sqrt{h_{i,j}^2 + h_{i+1,j}^2}} \text{ and } c_{i+1,j} = \frac{h_{i,j}}{\sqrt{h_{i,j}^2 + h_{i+1,j}^2}}.$$

Annihilating $h_{i+1,j}$ can now be accomplished by premultiplication with a unitary matrix,

$$\begin{bmatrix} c_{i+1,j} & s_{i+1,j} \\ s_{i+1,j} & -c_{i+1,j} \end{bmatrix} \mathbf{h} = \begin{bmatrix} * \\ 0 \end{bmatrix}.$$

In column j, the Givens rotations annihilating the lower subdiagonal entries will affect rows j to $j+p$. Since $\overline{\mathbf{H}}_j$ has p superdiagonal entries, the Givens rotations associated to column j will add no more than p additional subdiagonal entries with any other column. Thus $\mathbf{R}_j$ is upper triangular and $(2p+1)$-banded. Thus, the jth column of $\mathbf{R}_j$ has the following structure,

$$\mathbf{R}_j^{(j)} = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ r_{j-2p,j} \\ r_{j-2p+1,j} \\ \vdots \\ r_{j,j} \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

We have that $\mathbf{M}_j = \mathbf{V}_j\mathbf{R}_j^{-1}$ so that $\mathbf{M}_j\mathbf{R}_j = \mathbf{V}_j$, and this gives us the relationship between the block Lanczos vectors and the search directions,

$$\begin{aligned} r_{1,1}\mathbf{m}_1 &= \mathbf{v}_1 \\ r_{1,2}\mathbf{m}_1 + r_{2,2}\mathbf{m}_2 &= \mathbf{v}_2 \\ &\vdots \\ r_{1,2p+1}\mathbf{m}_1 + r_{2,2p+1}\mathbf{m}_2 + \cdots + r_{2p+1,2p+1}\mathbf{m}_{2p+1} &= \mathbf{v}_{2p+1} \\ r_{2,2p+2}\mathbf{m}_2 + r_{3,2p+2}\mathbf{m}_3 + \cdots + r_{2p+2,2p+2}\mathbf{m}_{2p+2} &= \mathbf{v}_{2p+2} \\ &\vdots \\ r_{j-2p,j}\mathbf{m}_{j-2p} + r_{j-2p+1,j}\mathbf{m}_{j-2p+1} + \cdots + r_{j,j}\mathbf{m}_j &= \mathbf{v}_j. \end{aligned} \tag{3.42}$$

Thus, to compute $\mathbf{m}_j$, we need $\mathbf{v}_j$ and the $2p$ previous search directions.

Since $\mathbf{R}_j$ results from the QR-factorization of $\overline{\mathbf{H}}_j$, we can use Givens rotations to annihilate the lower subdiagonal entries of $\overline{\mathbf{H}}_j$. This procedure is a generalization of the one for MINRES; however, since there are p lower subdiagonal entries, each columns requires p Givens rotations. At iteration j, let $\mathcal{G}_j^{(j)}$ denote the the unitary matrix used to annihilate the lower subdiagonal entries of column j of $\overline{\mathbf{H}}_j$. We write $\mathcal{G}_j^{(j)} = \mathbf{G}_{j+1,j}^{(j)}\mathbf{G}_{j+2,j}^{(j)} \cdots \mathbf{G}_{j+p,j}^{(j)}$ where $\mathbf{G}_{i,j}^{(j)}$ is the matrix applying the Givens rotation to annihilate entry $h_{i,j}$ from $\overline{\mathbf{H}}_j$. The application of the Givens rotations in $\mathcal{G}_j^{(j)}$ will affect columns j to $j+2p$; so, at step j, the rotations in $\mathcal{G}_{j-2p}^{(j)}$ to $\mathcal{G}_{j-1}^{(j)}$ will affect the jth column of $\overline{\mathbf{H}}_j$ after it is constructed, prior to computing the Givens rotations for step j. We can also use Householder reflections which require about half the work of using Givens rotations. It has been shown that we can effectively store products of

Householder reflections as a single matrix, applying them at once, without sacrificing accuracy [42]. Based on the implementation details discussed in Section 3.4.4 below, using Householder reflections here would be straightforward.

Observe that at every step, we construct a new set of Givens rotations. At step j, $\mathcal{G}_j^{(j)}$ is immediately applied to $\mathcal{G}_{j-1}^{(j)} \cdots \mathcal{G}_1^{(j)} \mathbf{E}_1^{(j)} \mathbf{S}$. Initially $\mathbf{E}_1^{(1)} \mathbf{S}$ is upper triangular. Multiplying by the first set of Givens rotations adds a subdiagonal band. From (3.38), we see that $\mathbf{E}_1^{(j-1)} \mathbf{S}$ is a submatrix of $\mathbf{E}_1^{(j)} \mathbf{S}$. Furthermore, $\mathcal{G}_{j-1}^{(j-1)} \cdots \mathcal{G}_1^{(j-1)} \mathbf{E}_1^{(j-1)} \mathbf{S}$ is contained as the upper block in $\mathcal{G}_{j-1}^{(j)} \cdots \mathcal{G}_1^{(j)} \mathbf{E}_1^{(j)} \mathbf{S}$. Therefore, the first subdiagonal in $\mathcal{G}_1^{(1)} \mathbf{E}_1^{(1)} \mathbf{S}$ exists in $\mathcal{G}_1^{(2)} \mathbf{E}_1^{(2)} \mathbf{S}$. Each further application of Givens rotations adds a new subdiagonal so that $\mathcal{G}_j^{(j)} \cdots \mathcal{G}_1^{(j)} \mathbf{E}_1^{(j)} \mathbf{S}$ has j lower subdiagonal bands. The rotations contained in $\mathcal{G}_j^{(j)}$ only effect rows j to $j+p$ of $\mathcal{G}_{j-1}^{(j)} \cdots \mathcal{G}_1^{(j)} \mathbf{E}_1^{(j)} \mathbf{S}$. Thus we have the relation $\mathbf{T}_j = \begin{bmatrix} \mathbf{T}_{j-1} \\ \mathbf{t}_j^T \end{bmatrix}$ where $\mathbf{t}_j \in \mathbb{C}^p$, and we can write the approximation $\mathbf{X}_j$ as an update of $\mathbf{X}_{j-1}$,

$$\mathbf{X}_j = \mathbf{X}_{j-1} + \mathbf{m}_j \mathbf{t}_j^T. \tag{3.43}$$

In [75], Saad observes that, as in the single right-hand-side case, a computed residual (also sometimes called the *recursive residual*) is available,

$$\left\| \mathbf{F}_j^{(i)} \right\| = \left\| (\overline{\mathbf{T}}_j^{(i)})_{j+1:j+i} \right\| \tag{3.44}$$

With this, we present Algorithm 3.4.2, the block MINRES algorithm.

3.4.4 Implementation Details

We conclude our description with some notes about practical implementation details. To summarize our storage needs; we must store $2p$ block Lanczos vectors, $2p$ search directions, $2p$ sets of Givens rotations (or Householder reflections), the lower subdiagonal entries of p previous columns, and the jth column of $\overline{\mathbf{H}}_j$ which will be transformed into the jth column of $\overline{\mathbf{R}}_j$. The fact that $\mathbf{A}$ is Hermitian allows for a large savings in the storage requirements of the block MINRES algorithm, but this also destroys much of the natural indexing that would be available if complete storage were necessary. We want the algorithm to deal with as few special cases as possible. With this in mind, we present data structures which allow for simple, efficient local indexing from which some simple patterns arise, allowing for a clean, simple algorithm with only one special case for the first iteration. We use the case block size $p = 5$ when presenting examples demonstrating the utility of the strategy.

We would like to store our basis vectors and search directions in queues. In Matlab, this structure does not exist. Thus, we will store information in an array, column-wise. A column in the array will represent data generated at a particular iteration. We treat this data structure as queue-like; any initial data is stored in the rightmost columns, and when a new column of information is created, it is inserted into the last column with all other data being shifted to the left by one column. The first column of data is discarded as new data is added to the last column. This means that in any data structure, the last column is associated to the current iteration (the most current information) allowing for local indexing. There is no need to explicitly store any sort of column indexing. We just keep track of the current global iteration.

We must store $2p$ block Lanczos vectors, and the most straightforward structure to store them in is a $n \times 2p$ array called $\mathbf{V}$. In order to establish a regular pattern, this array functions in a queue-like capacity. At the beginning, $\mathbf{V}$ is initialized with all zeros, and the columns of $\mathbf{U}_p$ are stored in the last p columns of $\mathbf{V}$.

Aside from providing a natural way to create and discard Lanczos vectors without explicitly keeping track of indices, with this setup, we always know that the vector in the $(p+1)$st column is the one on which $\mathbf{A}$ acts to generate the next Lanczos vector. The $2p$ search directions are stored as $\mathbf{M}$ in a similar fashion and $\mathbf{M}$ is initialized to all zeros.

For the matrices $\overline{\mathbf{H}}_j$ and $\mathbf{R}_j$, we only need to store the complete set of nonzero entries of the jth column. The jth column of $\overline{\mathbf{H}}_j$ has at most $2p+1$ nonzero entries. We store them in an array called $\mathbf{r}$ which is large enough to accommodate padding at the top by zeros which will be filled when the Givens rotations from previous steps are applied. While we do not need to store all the entries of the previous columns of $\overline{\mathbf{H}}_j$ or $\mathbf{R}_j$, we do need to store the lower subdiagonal entries of the p most recent columns of $\overline{\mathbf{H}}_j$, since the block Lanczos method uses the symmetry of $\mathbf{H}_j$ to fill the superdiagonal entries of the newest column. We store these entries in a $p \times p$ array called $\mathbf{H}$ where each column of $\mathbf{H}$ contains the lower subdiagonal entries of a particular column of $\overline{\mathbf{H}}_j$. At the start $\mathbf{H}$ is initialized as all zeros. Storing the entries in this manner results in the nonzero superdiagonal entries of the current column of $\overline{\mathbf{H}}_j$ being available as the nonzero antidiagonal entries of $\mathbf{H}$. This allows us to obtain the super diagonal entries of the current column without computing the associated inner products. For example, suppose $j = 7$, so we are constructing the 7th column of $\overline{\mathbf{H}}_j$. Then we would have

$$\mathbf{r} = \begin{bmatrix} 0 \\ h_{2,7} \\ h_{3,7} \\ h_{4,7} \\ h_{5,7} \\ h_{6,7} \\ h_{7,7} \\ h_{8,7} \\ h_{9,7} \\ h_{10,7} \\ h_{11,7} \\ h_{12,7} \\ h_{13,7} \\ h_{14,7} \end{bmatrix} \quad \text{and} \quad \mathbf{H} = \begin{bmatrix} h_{3,2} & h_{4,2} & h_{5,2} & h_{6,2} & h_{7,2} \\ h_{4,3} & h_{5,3} & h_{6,3} & h_{7,3} & h_{8,3} \\ h_{5,4} & h_{6,4} & h_{7,4} & h_{8,4} & h_{9,4} \\ h_{6,5} & h_{7,5} & h_{8,5} & h_{9,5} & h_{10,5} \\ h_{7,6} & h_{8,6} & h_{9,6} & h_{10,6} & h_{11,6} \end{bmatrix} \tag{3.45}$$

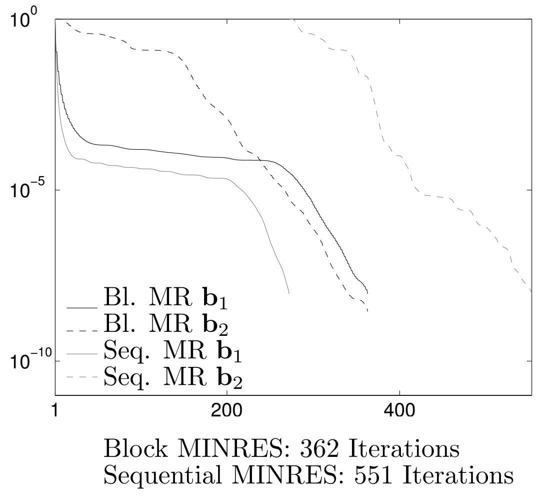
Observe in (3.45) that the red entries in $\mathbf{r}$ can be obtained from the antidiagonal entries of $\mathbf{H}$, also in red. Lastly, the Givens sine and cosines are stored in $p \times 2p$ arrays $\mathbf{s}$ and $\mathbf{c}$. Each column of $\mathbf{s}$ and $\mathbf{c}$ represents the sines and cosines of Givens rotations used to annihilate entries of a particular column $\overline{\mathbf{H}}_j$.

3.4.5 Effectiveness of Block MINRES and Some Numerical Test Cases

Worst-Case Residual Quality. We can easily describe the quality of the residual produced at iteration $j = k + mp$. For $\mathbf{b}^{(i)}$, the ith column of the right-hand side $\mathbf{B}$, Algorithm 3.4.2 minimizes the ith column of the residual $\mathbf{f}_j^{(i)}$ over the subspace $\mathbb{K}_{k,m}(A, \mathbf{F}_0)$. Thus, we can expect $\left\|\mathbf{f}_j^{(i)}\right\|$ to be at least as good as the norm of the residual produced by running J steps of MINRES with $\mathbf{b}^{(i)}$ as the single right-hand side, where $J = \begin{cases} k \text{ if } i < m \\ k+1 \text{ if } i \geq m \end{cases}$. This easily can be understood by recalling the definition of $\mathbb{K}_{k,m}(\mathbf{A}, \mathbf{F}_0)$ in (3.36). We see that unless the multiple right-hand sides are related in some special way, the additional information contained in $\mathbb{K}_{k,m}(A, \mathbf{F}_0)$

may not be helpful in the minimization process. For specially related right-hand sides, though, we may have convergence in many fewer steps.

Numerical Test Cases. We demonstrate how the relationship between the right-hand sides can effect the performance of this implementation of block MINRES with three experiments. In these experiments, we will compare the performance of block MINRES with sequential applications of Matlab's MINRES function. We will compare performance using iteration counts rather than CPU timings. Though our implementation of this block MINRES algorithm is efficient, there is one aspect of it which is not efficient when implemented as a Matlab code. In Section 3.4.4, we described storing the Lanczos vectors and search directions in queue structures $\mathbf{V}$ and $\mathbf{M}$. There are no queue structures in Matlab, so we simply store the vectors as columns in appropriately sized matrices, shifting columns to the left by one index at each step to mimic the queue structure when a new vector is constructed. This is an expensive operation, involving a great deal of data movement. When we profile our code using the Matlab code performance profiler, we see that this data movement accounts for approximately 20% of the runtime of the algorithm. In any production coding environment, a queue data structure only would move pointers in the queue, a much cheaper proposition. Therefore, we only count iterations for these experiments. Let $\mathbf{L} \in \mathbb{C}^{n_1 \times n_1}$, with $n_1 = 40000$, be the discretization of the Laplacian operator on a 200×200 using central differences. This matrix is negative-definite. Let $\mathbf{A} = -\mathbf{L} + 200\mathbf{I}$. Due to the eigenvalue distribution of $\mathbf{L}$, we have that $\mathbf{A}$ is indefinite. We will compare the performance of our block MINRES implementation with that of sequential runs of Matlab's MINRES for $\mathbf{A}$ with three pairs of right-hand sides.



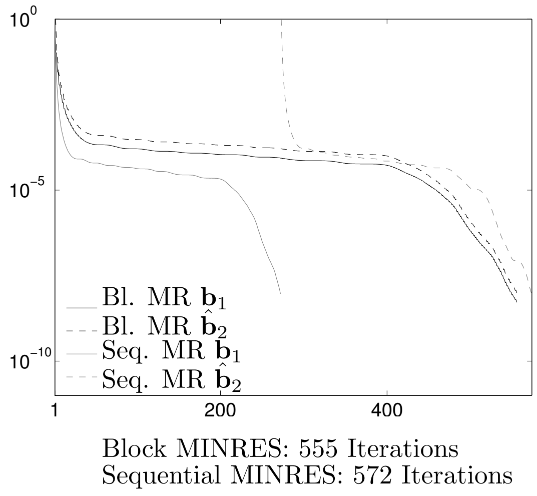


Figure 3.7. Performance of block MINRES for different right-hand sides. In the figure on the left, the two right-hand sides are $\mathbf{b}_1 = \mathbf{e}_{n_1}^{(1)}$ and $\mathbf{b}_2 = \mathbf{1}$. In the figure on the right, $\mathbf{b}_1$ does not change, but $\hat{\mathbf{b}}_2 = \mathbf{e}_{n_1}^{(2)}$.

For the first pair, let $\mathbf{b}_1 = \mathbf{e}_{n_1}^{(1)}$ and $\mathbf{b}_2 = \mathbf{1}$, the vector of all ones. For second pair of right-hand sides, we let $\hat{\mathbf{b}}_1 = \mathbf{b}_1$ but change the second right-hand side by letting $\hat{\mathbf{b}}_2 = \mathbf{e}_{n_1}^{(2)}$. In Figure 3.7, we show a comparison of convergence curves for these pairs of right-hand sides. We observe that exchanging $\mathbf{b}_2$ for $\hat{\mathbf{b}}_2$ degrades the performance of our Block MINRES implementation. Recall from (2.12) that the convergence of a Krylov subspace method for a Hermitian system is completely determined by its eigenvalues. For an indefinite system, the eigenvalues closest to the origin cause a delay in convergence. Therefore, we hypothesize that a pair of right-hand-sides have strong components from different parts of the eigenspace associated to eigenvalues of small magnitude might compliment each other well. Let $\mathbf{Q} \in \mathbb{C}^{n_1 \times 50}$ have orthonormal columns spanning the eigenspace associated to the 50 eigenvalues of $\mathbf{A}$ of smallest magnitude. We can study the magnitude of the components of $\mathbf{b}_1$,

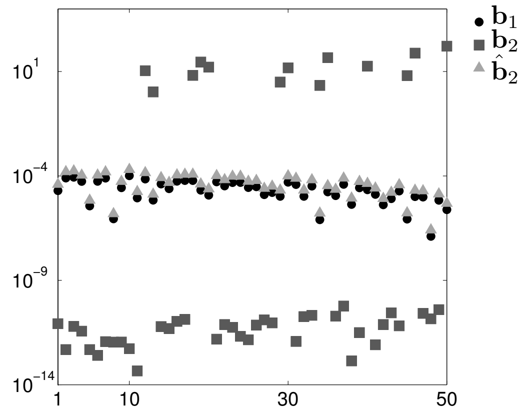
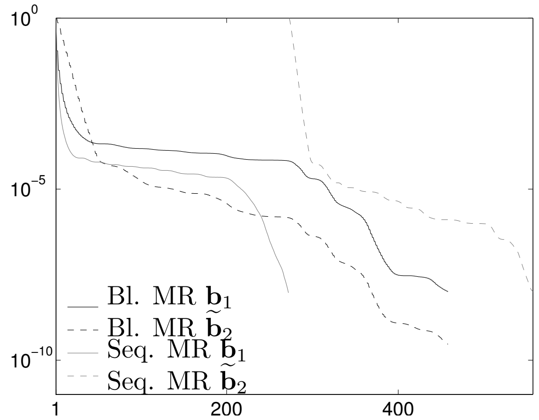


Figure 3.8. Magnitude of the components of different right-hand-sides in the eigenspace associated to the fifty eigenvectors associated to the smallest magnitude eigenvalues.

$\mathbf{b}_2$, and $\hat{\mathbf{b}}_2$ in $\mathcal{R}(\mathbf{Q})$ to better understand the convergence curves we see in Figure 3.7. In Figure 3.8, we plot the components of $\mathbf{Q}^*\mathbf{b}_1$, $\mathbf{Q}^*\mathbf{b}_2$, and $\mathbf{Q}^*\hat{\mathbf{b}}_2$. We see that the eigencomponents of $\mathbf{b}_1$, and $\hat{\mathbf{b}}_2$ are similar in magnitude, meaning that a block Krylov subspace for these two right-hand sides might not contain subspace information useful for accelerating convergence, when compared to its single-vector Krylov subspace counterparts. Many of the components of $\mathbf{b}_2$ in $\mathcal{R}(\mathbf{Q})$ are orders of magnitude smaller than those of $\mathbf{b}_1$. Thirteen are orders of magnitude larger. This may be part of the reason that we are able to achieve some acceleration of convergence using block MINRES for this pair of right-hand-sides. To test this theory, we can construct a new right-hand-side $\widetilde{\mathbf{b}}_2 = \mathbf{b}_2 - \mathbf{Q}\mathbf{Q}^*\mathbf{b}_2 + \mathbf{Q}\mathbf{Q}^*\hat{\mathbf{b}}_2$. This has the effect of removing the components of $\mathbf{b}_2$ in $\mathcal{R}(\mathbf{Q})$ and replacing them with the components of $\hat{\mathbf{b}}_2$. In Figure 3.9, we see that when applying block MINRES to the pair of right-hand-sides $\mathbf{b}_1$ and $\widetilde{\mathbf{b}}_2$, the method is less effective when compared to the performance $\mathbf{b}_1$ and



Block MINRES: 457 Iterations
Sequential MINRES: 555 Iterations

Figure 3.9. Performance of block MINRES after altering some components of the right-hand side $\mathbf{b}_2$.

$\mathbf{b}_2$ in Figure 3.7. This is by no means a rigorous analysis of convergence of a block method. These experiments only are meant to illustrate the variability of performance of a block method for different right-hand sides and provide some insight into this phenomenon.

It should be noted that using Ruhe's Lanczos implementation allows us to compare with sequential applications of a single-vector method iteration for iteration. However, this may not accurately reflect the performance of block methods compared with sequential applications of single-vector methods. One of the strengths of a true block method is that, at each iteration, we compute one matrix-block-vector product. This is certainly more expensive than multiplying a matrix times a single vector. However, for block size p, the matrix-block-vector products is not p times as expensive. Therefore, computational savings may be achieved in a true block method which cannot be achieved in implementations based on Ruhe's method. We could compute one matrix-block-vector product every p iterations of the algorithm to to achieve those savings. However, one might wonder if there is any reason not to use a true block method if we want to take advantage the matrix- block-vector product.

Algorithm 3.4.2: Block MINRES (Ruhe Implementation)

```
    Input  : $\mathbf{A} \in \mathbb{C}^{n\times n}$ Hermitian, $\mathbf{B} \in \mathbb{C}^{n\times p}$, $\mathbf{X}_0 = \mathbf{0}$ , $\epsilon > 0$, $M \in \mathbb{N}$
    Output: $\mathbf{X} \in \mathbb{C}^{n\times p}$ such that
            $\|\mathbf{B}(:,j) - \mathbf{AX}(:,j)\| / \|\mathbf{B}(:,j) - \mathbf{AX}_0(:,j)\| < \epsilon \; \forall j \le p$
 1  Compute the QR-Factorization $\mathbf{BV}_p\mathbf{S}$
 2  $\widehat{\mathbf{S}} \leftarrow \mathbf{SE}_1^{(1)}$
 3  $\mathbf{X} \leftarrow \mathbf{X}_0$
 4  $\mathbf{R} \leftarrow \mathbf{B} - \mathbf{AX}$
 5  $j \leftarrow 1$
 6  while $\max_{0<i\le p}\left\{\left\|(\overline{\mathbf{T}}_j^{(i)})_{j+1:j+i}\right\|\right\} < \varepsilon \left\|\mathbf{b}^{(i)}\right\|$ and $j \le M$ do
 7      $\mathbf{w} \leftarrow \mathbf{Av}_j$
 8      if $j > 1$ then
 9          for $i = j-p : j-1$ do
10              $h_{i,j} = h_{j,i}$
11              $\mathbf{w} \leftarrow h_{i,j}\mathbf{w}$
12      for $i = j : j+p-1$ do
13          $h_{i,j} = \mathbf{v}_i^*\mathbf{w}$
14          $\mathbf{w} \leftarrow h_{i,j}\mathbf{v}_i$
15      $h_{j+p,j} = \|\mathbf{w}\|$
16      $\mathbf{v}_{j+1} = \mathbf{w}/h_{j+p,j}$
17      if $j > 1$ then
18          $\overline{\mathbf{r}}_j^{(j)} \leftarrow \mathcal{G}_{j-1}^{(j)} \cdots \mathcal{G}_{j-2p}^{(j)}\overline{\mathbf{h}}_j^{(j)}$
19      $\mathcal{G}_j^{(j)} \leftarrow \mathbf{I}_j$
20      for $i = j+p-1 : j$ do
21          Generate Givens rotation for $\mathbf{G}_{i+1,j}^{(j)}$ to annihilate $h_{i+1,j}$
22          $\overline{\mathbf{r}}_j^{(j)} \leftarrow \mathbf{G}_{i+1,j}^{(j)}\overline{\mathbf{r}}_j^{(j)}$
23          $\widehat{\mathbf{S}} \leftarrow \mathbf{G}_{i+1,j}^{(j)}\widehat{\mathbf{S}}$
24          $\mathcal{G}_j^{(j)} \leftarrow \mathbf{G}_{i+1,j}^{(j)}\mathcal{G}_j^{(j)}$
25      if $m = 1$ then
26          $\mathbf{m}_1 = \mathbf{v}_1/\overline{\mathbf{R}}_1(1,1)$
27      else
28          $\mathbf{w} \leftarrow \mathbf{v}_j$
29          for $i = j-2p : j-1$ do
30              $\mathbf{w} \leftarrow \mathbf{w} - \overline{\mathbf{R}}_j(i,j)\mathbf{m}_i$
31          $\mathbf{m}_j = \mathbf{w}/\overline{\mathbf{R}}_j(j,j)$
32      $\mathbf{t}^T \leftarrow \widehat{\mathbf{S}}(j,:)$
33      $\mathbf{X} \leftarrow \mathbf{X} + \mathbf{m}_j\mathbf{t}^T$
34      $\widehat{\mathbf{S}} \leftarrow \begin{bmatrix} \widehat{\mathbf{S}} \\ \mathbf{0}_{1\times p} \end{bmatrix}$
35      $j \leftarrow j+1$
```

CHAPTER 4

SUBSPACE RECYCLING FOR SEQUENCES OF LINEAR SYSTEMS

4.1 Introduction

We wish to efficiently solve a sequence of linear systems

$$\mathbf{A}_i \mathbf{x}_i = \mathbf{b}_i, \tag{4.1}$$

where the coefficient matrices $\mathbf{A}_i \in \mathbb{C}^{n\times n}$ are assumed to be non-Hermitian and the right-hand sides $\mathbf{b}_i \in \mathbb{C}^{n\times 1}$. This is an important problem which arises in many computational applications, e.g., the solution of a sequence of discretized partial differential equations in which we vary parameters yielding different coefficient matrices, as well as the time-dependent PDEs. Such problems also arise in the solution of systems involving the Jacobian in Newton's method for solving a system of coupled nonlinear equations. In 2006, Parks et al. [68] proposed subspace recycling methods to solve a sequence of problems such as (4.1). They called this method GCRODR; but, in essence, it is GMRES residual minimization over an augmented

Krylov subspace. We will, though, continue to refer to this method as GCRODR for conciseness. In this chapter, we will develop two extensions to the existing subspace recycling technology.

4.1.1 Block GMRES with Subspace Recycling

First, we will extend GCRODR to use block Krylov subspaces. An application of particular interest for this algorithm arises in the field of fluid density functional theory (DFT). Fluid DFTs are frequently used to analyze properties of inhomogeneous fluids, such as predicting the electronic structure of a material. Recently, Heroux, Salinger, and Frink [44] introduced a new Schur complement formulation and developed a new mathematical framework for the solution of DFTs. Within this framework there is a Newton iteration, thus requiring that we solve a linear system involving a Jacobian at each step. We will seek to accelerate the solution of this sequence of systems by introducing random right-hand sides and solving them using block GCRODR.

4.1.2 Krylov Subspace Recycling for Families of Shifted Systems

Let $\mathcal{F}$ denote a family of coefficient matrices differing by multiples of the identity. In other words,

$$\mathcal{F} = \left\{\mathbf{A} + \sigma^{(\ell)}\mathbf{I}\right\}_{\ell=1}^{L}, \tag{4.2}$$

where L is the number of systems in the family, and we are solving the family of linear systems

$$\left(\mathbf{A} + \sigma^{(\ell)}\mathbf{I}\right)\mathbf{x}^{(\ell)} = \mathbf{b} \quad \text{for} \quad \ell = 1, \ldots, L.$$

We call the numbers $\left\{\sigma^{(\ell)}\right\}_{\ell=1}^{L} \subset \mathbb{C}$ *shifts.* There are many applications which warrant the solution of a family of shifted linear systems with coefficient matrices belonging to $\mathcal{F}$, such as quantum chromodynamics; see, e.g., [34]. Krylov subspace methods have been proposed to simultaneously solve this family of systems [32, 33, 80]. In our second project, we will extend subspace recycling technology to treat sequences of families of linear systems, where for the ith family $\mathcal{F}_i$ of linear systems, the coefficient matrices belong to

$$\mathcal{F}_i = \left\{\mathbf{A}_i + \sigma_i^{(\ell)}\mathbf{I}\right\}_{\ell=1}^{L_i}, \tag{4.3}$$

where L_i denotes the number of linear systems to be solved at step i. In other words, at step i, for shifts $\left\{\sigma_i^{(\ell)}\right\}_{\ell=1}^{L_i}$ we are solving systems of the form

$$\left(\mathbf{A}_i + \sigma_i^{(\ell)}\mathbf{I}\right)\mathbf{x}_i^{(\ell)} = \mathbf{b}_i \quad \text{for} \quad \ell = 1 \ldots L_i.$$

4.2 Background

We present a brief background and derivation of GCRODR. This method is, at its core, a GMRES method, where the minimization of the residual takes place over an augmented Krylov subspace. We will justify this statement in the derivation of the method. There is a rich history of Krylov subspace augmentation from which GCRODR follows; see, e.g., [6, 11, 13, 26, 51, 57, 59, 61, 90, 91]. One notational consideration: when the minimization is over a Krylov subspace of dimension m augmented with a recycled subspace of dimension k, we will denote this method GCRODR(m,k). Frequently, it is more convenient to perform experiments using

GCRODR($m-k$,k), thus minimizing over a subspace of dimension m, making it easier to compare the method to GMRES(m).

As we have discussed, Krylov subspace methods provide an effective tool for reducing the solution of large linear systems to a size for which a direct solver may be applied. However, a method such as GMRES requires that we retain all previous basis vectors and this basis grows at each iteration. The problems of limited storage and speed are still a concern as the size of n may limit how many vectors we can store at a time. One straightforward method, proposed to limit the amount of storage required for a Krylov subspace, is a restarted method. However, as discussed in Section 2.3, methods such as restarted GMRES can suffer from slow convergence or stagnation, as demonstrated in Figure 4.1. Previous methods have been proposed

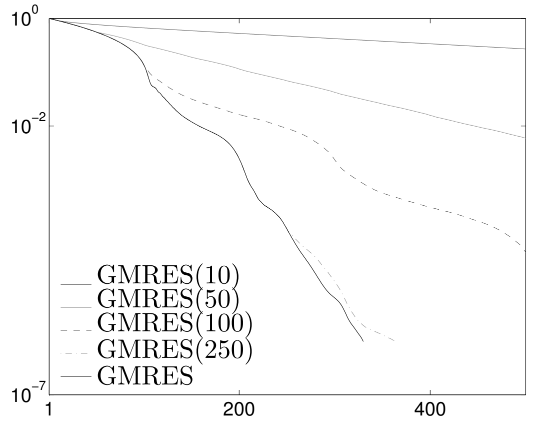


Figure 4.1. Performance of restarted GMRES with different restart parameters for a two-dimensional convection diffusion problem $10\Delta u + \mathbf{a}^T(\nabla u) + u = f$ with a reaction term on the domain $\Omega = [0,1]\times[0,1]$, where Ω is discretized with a 500×500 grid, $\mathbf{a}\in\mathbb{C}^2$ is a vector of ones, and $f = f(x,y) = 1$ is a constant function. We assume the Dirichlet condition that the function is zero on the boundary. Restart parameters are 10, 50, 100, and 250. Full GMRES is also included for comparison. We preconditioned with `ILU(0)`.

to save some part of the Krylov subspace at the end of a restart cycle, to improve convergence in the next cycle [57, 59, 74, 91].

4.2.1 GMRES with Subspace Recycling

GCRODR was presented by Parks et al. [68] in 2006. It is a method for solving the sequence of systems (4.1). Its development addressed some key deficiencies in previous algorithms. Methods such as GMRES-DR [59] compute harmonic Ritz vectors at the end of each cycle, and the Krylov subspace generated at the next cycle is augmented with these vectors. The solutions generated at each cycle are optimal (in the minimum residual sense) over the augmented subspace. However, this method is restricted to augmenting with harmonic Ritz vectors. As a result, there is no mechanism in GMRES-DR to augment the Krylov subspace using information generated while solving a previous linear system. Other augmented Krylov methods [13] allow for the use of arbitrary subspaces, but they do not produce optimal corrections over this augmented subspace. Parks et al. addressed these issues by combining the technology introduced by de Sturler in [90] and the augmentation techniques used by Morgan's GMRES-DR [59] to derive an algorithm which produces optimal corrections over an augmented Krylov subspace and allows for augmentation with arbitrary subspace information. This algorithm is called GCRODR.

We review this algorithm. In order to simplify notation, we drop the subscript i and discuss subspace recycling from the point of view of a single linear system. Thus, we are simply solving (1.1). Let us assume that we possess a matrix $\widehat{\mathbf{U}} \in \mathbb{C}^{n \times k}$, whose columns span a subspace obtained from the solution of a previous linear system. We begin by obtaining an orthonormal basis for the subspace spanned by the image of $\widehat{\mathbf{U}} \in \mathbb{C}^{n\times k}$ under the action of $\mathbf{A}$ using the QR-factorization $\mathbf{CS} = \mathbf{A}\widehat{\mathbf{U}}$ and assigning $\mathbf{U} = \widehat{\mathbf{U}}\mathbf{S}^{-1}$. Now we have

$$\mathbf{U} \in \mathbb{C}^{n\times k} \quad \text{and} \quad \mathbf{C} = \mathbf{AU} \quad \text{such that} \quad \mathbf{C}^*\mathbf{C} = \mathbf{I}. \tag{4.4}$$

We will now run an $m-k$ step cycle of GMRES, but for the projected operator $(\mathbf{I}-\mathbf{CC}^*)\mathbf{A}$. This will allow us to construct a solution update $\mathbf{x}_m = \mathbf{x}_0 + \mathbf{t}_m$ such that $\mathbf{t}_m \in \mathcal{S}$ where $\mathcal{S} = \mathcal{R}(\mathbf{U}) + \mathcal{K}_{m-k}(\mathbf{A}, \mathbf{r}_0)$. For this to work, we require that we begin with an approximation $\mathbf{x}_0$ such that our initial residual $\mathbf{r}_0$ be orthogonal to $\mathcal{R}(\mathbf{C})$. Thus, for an initial approximation $\widetilde{\mathbf{x}}_0$ and initial residual $\widetilde{\mathbf{r}}_0 = \mathbf{b} - \mathbf{A}\widetilde{\mathbf{x}}_0$, we must begin by orthogonally projecting the residual $\widetilde{\mathbf{r}}_0$ away from $\mathcal{R}(\mathbf{C})$ and then updating the approximation,

$$\mathbf{r}_0 = \widetilde{\mathbf{r}}_0 - \mathbf{CC}^*\widetilde{\mathbf{r}}_0 \;\text{ and }\; \mathbf{x}_0 = \widetilde{\mathbf{x}}_0 + \mathbf{UC}^*\widetilde{\mathbf{r}}_0. \tag{4.5}$$

Let us now generate the Krylov subspace $\mathcal{K}_{m-k}((\mathbf{I}-\mathbf{CC}^*)\mathbf{A}, \mathbf{r}_0)$ of dimension $m-k$ associated to the operator. Let $\mathbf{V}_{m-k} \in \mathbb{C}^{n\times(m-k)}$ have columns which form an orthonormal basis for $\mathcal{K}_{m-k}((\mathbf{I}-\mathbf{CC}^*)\mathbf{A}, \mathbf{r}_0)$ where we have the usual Arnoldi relation

$$\begin{aligned}
(\mathbf{I}-\mathbf{CC}^*)\mathbf{AV}_{m-k} &= \mathbf{V}_{m-k+1}\overline{\mathbf{H}}_{m-k} \\
\mathbf{AV}_{m-k} - \mathbf{CC}^*\mathbf{AV}_{m-k} &= \mathbf{V}_{m-k+1}\overline{\mathbf{H}}_{m-k} \\
\mathbf{AV}_{m-k} &= \mathbf{CC}^*\mathbf{AV}_{m-k} + \mathbf{V}_{m-k+1}\overline{\mathbf{H}}_{m-k} \qquad (4.6)
\end{aligned}$$

with $\overline{\mathbf{H}}_{m-k} \in \mathbb{C}^{(m-k+1)\times(m-k)}$ being the upper Hessenberg matrix representing the restriction and projection of $(\mathbf{I}-\mathbf{CC}^*)\mathbf{A}$ onto $\mathcal{K}_{m-k}((\mathbf{I}-\mathbf{CC}^*)\mathbf{A}, \mathbf{r}_0)$. Since the columns of $\mathbf{V}_{m-k}$ are orthogonal to the columns of $\mathbf{C}$, we can write the Arnoldi-like relation,

$$\mathbf{A}\begin{bmatrix}\mathbf{U} & \mathbf{V}_{m-k}\end{bmatrix} = \begin{bmatrix}\mathbf{C} & \mathbf{V}_{m-k+1}\end{bmatrix}\begin{bmatrix}\mathbf{I} & \mathbf{B} \\ \mathbf{0} & \overline{\mathbf{H}}_{m-k}\end{bmatrix}, \tag{4.7}$$

where $\mathbf{B} = \mathbf{C}^*\mathbf{A}\mathbf{V}_{m-k}$. If we write

$$\widehat{\mathbf{V}}_m = \begin{bmatrix}\widetilde{\mathbf{U}} & \mathbf{V}_{m-k}\end{bmatrix} \text{ and } \widehat{\mathbf{W}}_{m+1} = \begin{bmatrix}\mathbf{C} & \mathbf{V}_{m-k+1}\end{bmatrix} \text{ and } \overline{\mathbf{G}}_m = \begin{bmatrix}\mathbf{I}_k & \mathbf{B} \\ \mathbf{0} & \overline{\mathbf{H}}_{m-k}\end{bmatrix}, \tag{4.8}$$

then we can rewrite (4.7) as

$$\mathbf{A}\widehat{\mathbf{V}}_m = \widehat{\mathbf{W}}_{m+1}\overline{\mathbf{G}}_m. \tag{4.9}$$

At the end of the cycle, we can solve

$$\mathbf{x} = \underset{\mathbf{x}\in\mathbf{x}_0+\mathcal{R}(\widehat{\mathbf{V}}_m)}{\operatorname{argmin}} \|\mathbf{b} - \mathbf{A}\mathbf{x}\|$$

by solving the small least-squares problem

$$\mathbf{y}_m = \underset{\mathbf{y}\in\mathbb{C}^m}{\operatorname{argmin}} \left\| \|\mathbf{r}_0\| \, \mathbf{e}_{m+1}^{(k+1)} - \overline{\mathbf{G}}_m\mathbf{y} \right\| \tag{4.10}$$

and setting

$$\mathbf{x} = \mathbf{x}_0 + \widehat{\mathbf{V}}_m\mathbf{y} \text{ and } \mathbf{r} = \mathbf{r}_0 - \widehat{\mathbf{W}}_{m+1}\overline{\mathbf{G}}_m\mathbf{y}. \tag{4.11}$$

However, we do not actually need to solve this least squares problem. We can solve a smaller problem. We can decouple (4.10), and instead solve a small GMRES minimization problem. Let

$$\mathbf{y}_m = \begin{bmatrix}\mathbf{y}_{k\times 1} \\ \mathbf{y}_{(m-k)\times 1}\end{bmatrix} \tag{4.12}$$

where $\mathbf{y}_{k\times 1} \in \mathbb{C}^k$ and $\mathbf{y}_{(m-k)\times 1} \in \mathbb{C}^{m-k}$. We can then solve

$$\mathbf{y}_{(m-k)\times 1} = \underset{\mathbf{y}\in\mathbb{C}^{(m-k)}}{\operatorname{argmin}} \left\| \|\mathbf{r}_0\| \, \mathbf{e}_{m-k}^{(1)} - \overline{\mathbf{H}}_{m-k}\mathbf{y} \right\|,$$

and compute $\mathbf{y}_{k\times 1} = \mathbf{B}\mathbf{y}_{(m-k)\times 1}$. This is a consequence of [90, Equation 28] used in the proof of [90, Theorem 2.2]. Furthermore, by this same theorem, the GMRES recessive residual norm computing during this minimization is also the residual norm for the approximation computed over the augmented space, thus yielding a cheap residual norm computation. We can rewrite the update (4.11)

$$\mathbf{x} = \mathbf{x}_0 + \mathbf{U}\mathbf{y}_{k\times 1} + \mathbf{V}_{m-k}\mathbf{y}_{(m-k)\times 1} \text{ and } \mathbf{r} = \mathbf{r}_0 - \mathbf{V}_{m-k+1}\overline{\mathbf{H}}_{m-k}\mathbf{y}_{(m-k)\times 1}. \tag{4.13}$$

At the end of the cycle, we construct a new recycled subspace; and if we have not already converged, we begin the next cycle. If we have converged, we save this constructed recycled subspace for use in solving the next linear system. It should be noted that if we begin with no initial recycled subspace, select harmonic Ritz vectors when constructing the recycled space at the end of each cycle, and solve only one system, this method is algebraically equivalent to the GMRES-DR method of Morgan [59].

In Figure 4.2, we demonstrate the acceleration of convergence of recycled GMRES with a two-dimensional recycled subspace, as compared to GMRES and GMRES(20).

Convergence results for augmented Krylov subspace methods have been previously presented, see, e.g., [23, 74], but not much work has been done in the context of subspace recycling with GCRODR. Some not yet published work was recently presented by de Sturler which specifically addressed the convergence behavior of optimal methods in which we orthogonally project the Krylov subspace away from the recycled subspace and compute an optimal correction over the augmented subspace

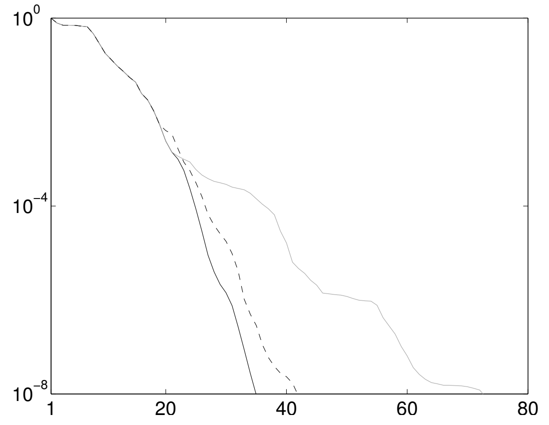


Figure 4.2. The convergence curves of GMRES (**solid black curve**), GMRES(20) (**solid grey curve**), and GCRODR(18,2) (**dashed black curve**). Observe that the latter two methods have the same storage requirements. The coefficient matrix and right-hand side are from the `sherman5` system from Matrix Market [1]. We preconditioned with `ILU(0)`. We see that for this problem, it suffices to retain a two-dimensional subspace between restart cycles to achieve convergence almost as good as full GMRES.

[92]. This work asserts that the improvement of convergence bounds from recycling a particular subspace can be quantified according to the quality of the recycled subspace as an invariant subspace. A particular finding, backed up by empirical observation, is that an approximate invariant subspace of modest quality will still yield improvements in bounds on the residual norm.

4.3 Block GMRES with Subspace Recycling

We wish to develop a block version of the GCRODR algorithm. Such an algorithm could be used to solve a sequence of block systems,

$$\mathbf{A}_i \mathbf{X}_i = \mathbf{B}_i \tag{4.14}$$

where $\mathbf{B}_i \in \mathbb{C}^{n \times p}$. Furthermore, in the case of (4.14) we can introduce random right-hand sides simply to accelerate convergence of the Krylov method and to enrich the subspace from which we select our recycled vectors. This is a natural extension of GCRODR, and the derivation should be straightforward. We will now be augmenting a block Krylov subspace rather than one generated from a single vector. We will also need to make a few notation changes to accommodate the notation associated to block Krylov methods.

We have seen already a block Lanczos-based method based on Ruhe's implementation in Section 3.4. Here, we will work with a classical block Krylov subspace method. We begin by reviewing the block Arnoldi process, which has been previously described in, e.g., [75, Section 6.12]. As before, for ease of notation, we describe our method for one system $\mathbf{AX} = \mathbf{B}$, and we will use information from a subspace generated by a previous system.

Let $\mathbf{X}_0$ be the initial approximation to the block solution and $\mathbf{R}_0 = \mathbf{B} - \mathbf{AX}_0$ be the initial block residual. Recall that the block Krylov subspace $\mathbb{K}_m(\mathbf{A}, \mathbf{R}_0)$ is a

Algorithm 4.3.1: A step of the block Arnoldi algorithm

```
   Input  : $\mathbf{A} \in \mathbb{C}^{n \times n}$, $\mathbf{V}_1, \ldots, \mathbf{V}_m \in \mathbb{C}^{n \times p}$
   Output: The block Arnoldi vector $\mathbf{V}_{m+1}$
 1 $\widehat{\mathbf{V}}_{m+1} = \mathbf{AV}_m$
 2 for $i = 1$ to $m$ do
 3     $\mathbf{H}_{i,m} = \mathbf{V}_i^* \widehat{\mathbf{V}}_{m+1} \in \mathbb{C}^{p \times p}$
 4     $\widehat{\mathbf{V}}_{m+1} \leftarrow \widehat{\mathbf{V}}_{m+1} - \mathbf{V}_i \mathbf{H}_{i,m}$
 5 Define $\mathbf{V}_{m+1}$ and $\mathbf{H}_{m+1,m}$ using reduced QR-factorization
   $\widehat{\mathbf{V}}_{m+1} = \mathbf{V}_{m+1} \mathbf{H}_{m+1,m}$
```

generalization of the definition of a Krylov subspace, i.e.,

$$\mathbb{K}_m(\mathbf{A},\mathbf{R}_0) = \operatorname{span}\left\{\mathbf{R}_0, \mathbf{A}\mathbf{R}_0, \mathbf{A}^2\mathbf{R}_0, \ldots \mathbf{A}^{m-1}\mathbf{R}_0\right\}.$$

As we previously discussed, this definition is equivalent to

$$\mathbb{K}_m(\mathbf{A},\mathbf{R}_0) = \mathcal{K}_m(\mathbf{A},\mathbf{r}_0^{(1)}) + \mathcal{K}_m(\mathbf{A},\mathbf{r}_0^{(2)}) + \cdots + \mathcal{K}_m(\mathbf{A},\mathbf{r}_0^{(p)}),$$

where $\mathbf{z} \in \mathbb{K}_m(\mathbf{A},\mathbf{R}_0)$ can be decomposed as $\mathbf{z} = \mathbf{z}_1 + \mathbf{z}_2 + \cdots \mathbf{z}_p$ with $\mathbf{z}_i \in \mathcal{K}_m(\mathbf{A},\mathbf{r}_0^{(p)})$, where the decomposition is **not necessarily unique**. This is because the component Krylov subspaces of $\mathbb{K}_m(\mathbf{A},\mathbf{R}_0)$ are not necessarily disjoint. Let us assume for the following discussion that m is small enough that $\dim \mathbb{K}_m(\mathbf{A},\mathbf{R}_0) = mp$. Following from the description in [75, Section 6.12], we represent $\mathbb{K}_m(\mathbf{A},\mathbf{R}_0)$ in terms of the block Arnoldi basis $\{\mathbf{V}_1, \mathbf{V}_2, \ldots, \mathbf{V}_m\}$ where $\mathbf{V}_i \in \mathbb{C}^{n\times p}$ has orthonormal columns and each column of $\mathbf{V}_i$ is orthogonal to all columns of $\mathbf{V}_j$ for all $j \neq i$. We obtain $\mathbf{V}_1$ via the reduced QR-factorization $\mathbf{R}_0 = \mathbf{V}_1\mathbf{S}_0$ where $\mathbf{S}_0 \in \mathbb{C}^{p\times p}$ is upper triangular. We can generate $\mathbf{V}_{m+1}$ with the block Arnoldi step, shown in Algorithm 4.3.1. Let $\mathbf{W}_m = \begin{bmatrix}\mathbf{V}_1 & \mathbf{V}_2 \cdots \mathbf{V}_m\end{bmatrix} \in \mathbb{C}^{n\times mp}$. Let $\overline{\mathbf{H}}_m = (\mathbf{H}_{ij}) \in \mathbb{C}^{(m+1)p\times mp}$. This yields the block Arnoldi relation

$$\mathbf{A}\mathbf{W}_m = \mathbf{W}_{m+1}\overline{\mathbf{H}}_m. \tag{4.15}$$

A straightforward generalization of GMRES for block Krylov subspaces (called block GMRES) was described, e.g., in [75, Chapter 6]. In Figure 4.3, we demonstrate the convergence of block GMRES for a sample system. The notation we will use for a block version of restarted GMRES with cycle length m is block-GMRES(m). In this case, for each length m cycle, we will generate a block Krylov subspace of size mp.

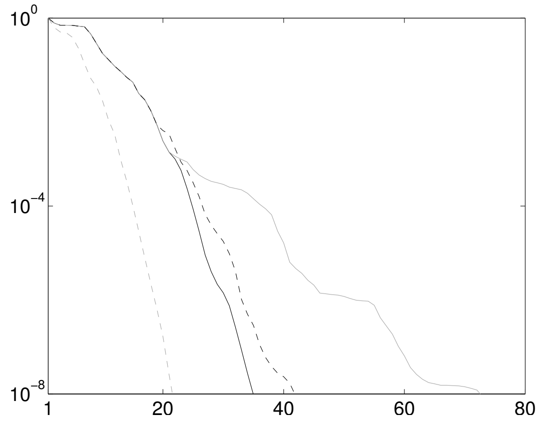


Figure 4.3. Demonstration of the convergence of block GMRES (**dashed grey curve**) as compared to convergence of the methods used in Figure 4.2. For block GMRES, a second right-hand side was generated using Matlab's `rand()`. Convergence is measured using only the residual norm for the first right-hand side. One matrix-block-vector product is executed at each iteration. If the cost of one block-p matrix-vector product is equal to the cost of $\frac{2}{3}p$ single matrix-vector products, as in Figure 4.6, then block GMRES outperforms GMRES in this experiment.

We now describe a cycle of block GCRODR. As in the case of our description of GCRODR in Section 4.2, let us suppose we have a k-dimensional subspace, spanned by the columns of $\mathbf{U}$ where $\mathbf{AU} = \mathbf{C}$ such that $\mathbf{C}^*\mathbf{C} = \mathbf{I}_k$. Let $\widetilde{\mathbf{X}}_0$ be the initial approximation, and let $\widetilde{\mathbf{R}}_0 = \mathbf{B} - \mathbf{A}\widetilde{\mathbf{X}}_0$ be the initial residual. If the columns of $\widetilde{\mathbf{R}}_0$ are not orthogonal to $\mathcal{R}(\mathbf{C})$, then we must orthogonally project the columns of the block residual and update the solution, i.e.,

$$\mathbf{R}_0 = \widetilde{\mathbf{R}}_0 - \mathbf{C}\mathbf{C}^*\widetilde{\mathbf{R}}_0 \quad \text{and} \quad \mathbf{X}_0 = \widetilde{\mathbf{X}}_0 + \mathbf{U}\mathbf{C}^*\widetilde{\mathbf{R}}_0.$$

Let $M = mp + k$. We define $\mathbf{V}_1$ using the reduced QR-factorization $\mathbf{R}_0 = \mathbf{V}_1\mathbf{S}_0$. We generate the M-dimensional augmented block Krylov subspace

$$\mathcal{R}(\mathbf{U}) + \mathbb{K}_m((\mathbf{I} - \mathbf{C}\mathbf{C}^*)\mathbf{A}, \mathbf{V}_1)$$

from which to select the solution update. In spite of this notation, we will still call this method Block GCRODR(m,k). Let

$$\mathbf{W}_m = \begin{bmatrix}\mathbf{V}_1 & \mathbf{V}_2 \cdots \mathbf{V}_m\end{bmatrix}$$

be generated by the block Arnoldi method, with orthonormal columns spanning $\mathbb{K}_m((\mathbf{I} - \mathbf{C}\mathbf{C}^*)\mathbf{A}, \mathbf{V}_1)$ with the associated block Arnoldi relation

$$(\mathbf{I} - \mathbf{C}\mathbf{C}^*)\mathbf{A}\mathbf{W}_m = \mathbf{W}_{m+1}\overline{\mathbf{H}}_m.$$

By construction, the columns of $\mathbf{W}_{m+1}$ are orthogonal to the columns of $\mathbf{C}$, yielding a block version of the Arnoldi-like relation (4.7),

$$\mathbf{A}\begin{bmatrix}\mathbf{U} & \mathbf{W}_m\end{bmatrix} = \begin{bmatrix}\mathbf{C} & \mathbf{W}_{m+1}\end{bmatrix}\begin{bmatrix}\mathbf{I}_k & \mathbf{B} \\ \mathbf{0} & \overline{\mathbf{H}}_m\end{bmatrix}, \tag{4.16}$$

where $\mathbf{B} = \mathbf{C}^*\mathbf{A}\mathbf{W}_m$ represents the entries generated by orthogonalizing the columns of the new block Krylov basis vector against $\mathcal{R}(\mathbf{C})$. Let

$$\widehat{\mathbf{W}}_M = \begin{bmatrix}\mathbf{U} & \mathbf{W}_m\end{bmatrix} \quad \text{and} \quad \widetilde{\mathbf{W}}_{M+p} = \begin{bmatrix}\mathbf{C} & \mathbf{W}_{m+1}\end{bmatrix} \quad \text{and} \quad \overline{\mathbf{G}}_M = \begin{bmatrix}\mathbf{I}_k & \mathbf{B} \\ \mathbf{0} & \overline{\mathbf{H}}_m\end{bmatrix}. \tag{4.17}$$

This yields the block Arnoldi-like relation, analogous to (4.9),

$$\mathbf{A}\widehat{\mathbf{W}}_M = \widetilde{\mathbf{W}}_{M+p}\overline{\mathbf{G}}_M. \tag{4.18}$$

At the end of the cycle, we want to solve

$$\mathbf{X} = \underset{\mathbf{X} \in \mathbf{X}_0 + \mathcal{R}(\widehat{\mathbf{W}}_M)}{\operatorname{argmin}} \left\| \mathbf{B} - \mathbf{A}\mathbf{X} \right\|_{\mathrm{F}},$$

which is equivalent to solving

$$\mathbf{x}^{(i)} = \underset{\mathbf{x} \in \mathbf{x}_0^{(i)} + \mathcal{R}(\widehat{\mathbf{W}}_M)}{\operatorname{argmin}} \left\| \mathbf{b}^{(i)} - \mathbf{A}\mathbf{x} \right\| \tag{4.19}$$

for each $i = 1, \ldots, p$. If we let $\mathbf{E}_p \in \mathbb{C}^{(M+p)\times p}$ be the matrix consisting of the $(k+1)$st to $(k+p)$th columns of the identity matrix of order $M+p$. Then solving (4.19) is equivalent to solving

$$\mathbf{Y}_m = \underset{\mathbf{Y} \in \mathbb{C}^{M\times p}}{\operatorname{argmin}} \left\| \mathbf{E}_p \mathbf{S}_0 - \overline{\mathbf{G}}_M \mathbf{Y} \right\| \tag{4.20}$$

and updating the block solution and residual,

$$\mathbf{X} = \mathbf{X}_0 + \widehat{\mathbf{W}}_M \mathbf{Y} \quad \text{and} \quad \mathbf{R} = \mathbf{R}_0 - \widetilde{\mathbf{W}}_{M+p} \overline{\mathbf{G}}_M \mathbf{Y}. \tag{4.21}$$

As in the single-vector case, we can follow [90], and decouple this least squares problem into the solution of a block GMRES minimization over $\mathbb{K}_m((\mathbf{I}-\mathbf{C}\mathbf{C}^*)\mathbf{A}, \mathbf{V}_1)$ and then solve for a correction from $\mathcal{R}(\mathbf{U})$ as we did in Section 4.2.1. Let

$$\mathbf{Y}_m = \begin{bmatrix} \mathbf{Y}_{k\times p} \\ \mathbf{Y}_{mp\times p} \end{bmatrix},$$

with $\mathbf{Y}_{k\times p} \in \mathbb{C}^{k\times p}$ and $\mathbf{Y}_{mp\times p} \in \mathbb{C}^{mp\times p}$. Then we can solve the block GMRES minimization,

$$\mathbf{Y}_{mp\times p} = \underset{\mathbf{Y} \in \mathbb{C}^{mp\times p}}{\operatorname{argmin}} \left\| \overline{\mathbf{H}}_m \mathbf{Y} - \mathbf{E}_1 \mathbf{S}_0 \right\|$$

where $\mathbf{E}_1 \in \mathbb{C}^{(M+p)\times p}$ is the matrix whose columns are the first p columns of the $M+p$ dimension identity matrix. We then compute $\mathbf{Y}_{k\times p} = \mathbf{B}\mathbf{Y}_{mp\times p}$ by back substitution. We compute a new approximate invariant subspace; and if we have not converged, we begin the next cycle. Algorithm 4.3.2 gives a complete pseudocode description of the algorithm, without specifying the particular method used to select the new recycled subspace.

4.3.1 Convergence Comparison with GCRODR for a Single Right-hand Side

Intuitively, using block GCRODR to construct a richer Krylov subspace from which to select solutions and a recycled subspace should result in faster convergence, or at least convergence should be no slower than GCRODR in terms of iteration count. We would like to formalize this idea.

We have derived and implemented a block GCRODR algorithm, and which, at its core, has a block GMRES kernel. For a single cycle, block GCRODR is block GMRES for the operator $(\mathbf{I} - \mathbf{C}\mathbf{C}^*)\mathbf{A}$. Therefore, we can look to some of the convergence properties of block GMRES to gain further insight. Simoncini and Gallopoulos [82] discussed the convergence properties of block GMRES, including a result by Vital [99].

Theorem 4.1: Given the block Krylov subspace $\mathbb{K}_m(\mathbf{A}, \mathbf{R}_0)$ and $\mathcal{K}_m(\mathbf{A}, \mathbf{r}_0^{(j)})$ for $j = 1, \ldots, p$, we have the following relationship between the block residual produced by block GMRES and the residuals produced by GMRES for each right-hand side,

$$\min_{\mathbf{Z} \in \mathbb{K}_m(\mathbf{A}, \mathbf{R}_0)} \|\mathbf{R}_0 - \mathbf{A}\mathbf{Z}\|_{\mathrm{col}} \leq \max_{j=1,\ldots,s} \min_{\mathbf{z}_j \in \mathcal{K}_m(\mathbf{A}, \mathbf{r}_0^{(j)})} \left\| \mathbf{r}_0^{(j)} - \mathbf{A}\mathbf{z}_j \right\|. \tag{4.22}$$

Algorithm 4.3.2: The Block GCRODR Algorithm

```
   Input  : $\mathbf{A} \in \mathbb{C}^{n\times n}$, $\mathbf{B} \in \mathbb{C}^{n\times p}$, $\mathbf{X}_0 \in \mathbb{C}^{n\times p}$, $\varepsilon > 0$ the convergence
            tolerance, $m$ the number of block Arnoldi vectors generated, $k$
            the desired dimension of the recycled subspace.
   Output: $\mathbf{X} \in \mathbb{C}^{n\times p}$ such that $\|\mathbf{B} - \mathbf{AX}\|_{col} \leq \varepsilon$
1  Set $\mathbf{R}_0 = \mathbf{B} - \mathbf{AX}_0$
2  Set $i = 1$
3  Set $M = mp + k$
4  if $\mathbf{U}$ is defined from solving a previous linear system then
5      Define $\mathbf{C}$ using reduced QR-Factorization $\mathbf{AU} = \mathbf{CS}$ and compute
       $\mathbf{U} \leftarrow \mathbf{US}^{-1}$
6      $\mathbf{X}_1 = \mathbf{X}_0 + \mathbf{UC}^*\mathbf{R}_0$; $\mathbf{R}_1 = \mathbf{R}_0 - \mathbf{CC}^*\mathbf{R}_0$
7  else
8      Define $\mathbf{V}_1$ using reduced QR-Factorization $\mathbf{R}_0 = \mathbf{V}_1\mathbf{S}_0$
9      Perform $m$ steps of block GMRES, solving
       $\mathbf{Y}_1 = \underset{\mathbf{Y}\in\mathbb{C}^{m\times p}}{\operatorname{argmin}} \left\|\mathbf{ES}_0 - \mathbf{W}_{m+1}\overline{\mathbf{H}}_m\mathbf{Y}\right\|_F$, generating $\mathbf{W}_{m+1}$ and $\overline{\mathbf{H}}_m$
       where $\mathbf{E} = \begin{bmatrix}\mathbf{e}_{mp}^{(1)} & \cdots & \mathbf{e}_{mp}^{(p)}\end{bmatrix}$
10     $\mathbf{X}_1 = \mathbf{X}_0 + \mathbf{W}_m\mathbf{Y}_1$; $\mathbf{R}_1 = \mathbf{R}_0 - \mathbf{W}_{m+1}\overline{\mathbf{H}}_m\mathbf{Y}_1$
11     Select a dimension $k$ subspace of $\mathbb{K}_m(\mathbf{A}, \mathbf{V}_1)$ and store coefficients as
       $\mathbf{P} \in \mathbb{C}^{mp\times k}$
12     Set $\mathbf{U} = \mathbf{W}_m\mathbf{P}$
13     Compute reduced QR-factorization $\overline{\mathbf{H}}_m\mathbf{P} = \mathbf{QS}$
14     Set $\mathbf{C} = \mathbf{W}_{m+1}\mathbf{Q}$; $\mathbf{U} \leftarrow \mathbf{US}^{-1}$
15 while $\|\mathbf{B} - \mathbf{AX}_i\|_F > \varepsilon$ do
16     $i \leftarrow i + 1$
17     Define $\mathbf{V}_1$ using reduced QR-Factorization $\mathbf{R}_1 = \mathbf{V}_1\mathbf{S}_1$
18     Compute a basis for $\mathbb{K}_m((\mathbf{I} - \mathbf{CC}^*)\mathbf{A}, \mathbf{V}_1)$ using the block Arnoldi
       method, generating $\mathbf{W}_{m+1}$, $\overline{\mathbf{H}}_m$, and $\mathbf{B}$
19     Define $\mathbf{D}$ to be the diagonal matrix such that $\widetilde{\mathbf{U}} = \mathbf{UD}$ has columns
       of unit norm
20     Set $\widehat{\mathbf{W}}_M = \begin{bmatrix}\widetilde{\mathbf{U}} & \mathbf{W}_m\end{bmatrix}$; $\widetilde{\mathbf{W}}_{M+P} = \begin{bmatrix}\mathbf{C} & \mathbf{W}_{m+1}\end{bmatrix}$
21     Set $\overline{\mathbf{G}}_M = \begin{bmatrix}\mathbf{D} & \mathbf{B} \\ \mathbf{0} & \overline{\mathbf{H}}_m\end{bmatrix}$
22     Solve for $\mathbf{Y}_i = \underset{\mathbf{Y}\in\mathbb{C}^{M\times p}}{\operatorname{argmin}} \left\|\mathbf{E}_p\mathbf{S}_1 - \overline{\mathbf{G}}_M\mathbf{Y}\right\|_F$
23     Set $\mathbf{X}_i = \mathbf{X}_{i-1} + \widehat{\mathbf{W}}_M\mathbf{Y}_i$; $\mathbf{R}_i = \mathbf{R}_{i-1} - \widetilde{\mathbf{W}}_{M+p}\overline{\mathbf{G}}_M\mathbf{Y}_i$
24     Select a dimension $k$ subspace of $\mathcal{R}(\widehat{\mathbf{W}}_M)$ and store coefficients as
       $\mathbf{P} \in \mathbb{C}^{Np\times k}$
25     Set $\mathbf{U} = \widehat{\mathbf{W}}_M\mathbf{P}$
26     Compute reduced QR-factorization $\overline{\mathbf{G}}_m\mathbf{P} = \mathbf{QS}$
27     Set $\mathbf{C} = \widetilde{\mathbf{W}}_{M+p}\mathbf{Q}$ and $\mathbf{U} \leftarrow \mathbf{US}^{-1}$
28 Store $\mathbf{U}$ in memory to serve as initial recycle subspace for next function
   call.
```

Furthermore, we can see that there is an even more straightforward relationship between residuals produced by the two methods.

Proposition 4.2: Given the same hypotheses as in Theorem 4.1 we have that

$$\min_{\mathbf{z}_j \in \mathbb{K}_m(\mathbf{A},\mathbf{R}_0)} \left\| \mathbf{r}_0^{(j)} - \mathbf{A}\mathbf{z}_j \right\| \leq \min_{\mathbf{z}_j \in \mathcal{K}_m(\mathbf{A},\mathbf{r}_0^{(j)})} \left\| \mathbf{r}_0^{(j)} - \mathbf{A}\mathbf{z}_j \right\|. \tag{4.23}$$

Proof. This can be shown by observing that in the block method, the residual is projected orthogonal to a larger subspace. □

At step m, for $\mathbf{r}_m^{(i)}$, the ith column of the block residual, we can bound the norm $\left\|\mathbf{r}_m^{(i)}\right\| \leq \left\|\hat{\mathbf{r}}_m^{(i)}\right\|$ where $\hat{\mathbf{r}}_m^{(i)}$ is the residual produced by running m iterations of GMRES on the ith right-hand side. Each column of the block residual at step m is bounded from above by the norm of the mth residual produced by running m steps of GMRES for each right- hand side individually. We can now easily derive a corollary for comparing the convergence of single-vector and block GCRODR.

Corollary 4.3: Given a recycled space $\mathbf{U}$ such that $\mathbf{C} = \mathbf{AU}$ and $\mathbf{C}^*\mathbf{C} = \mathbf{I}_k$, suppose we run m-step cycles of GCRODR and the block GMRES variant. Define $\widehat{\mathbf{V}}_m^{(j)}$ as in (4.8) where the superscript (j) indicates we have generated the Krylov subspace $\mathcal{K}((\mathbf{I} - \mathbf{CC}^*)\mathbf{A}, \mathbf{r}_0^{(j)})$, and define $\widehat{\mathbf{W}}_M$ as in (4.17) with $M = mp + k$. Then we have the following relationship between the block residual produced by block GCRODR and the individual residuals produced by GCRODR for each right-hand side,

$$\min_{\mathbf{Z}\in\mathcal{R}(\widehat{\mathbf{W}}_M)} \|\mathbf{R}_0 - \mathbf{A}\mathbf{Z}\|_{\mathrm{col}} \leq \max_{j=1,\ldots,s} \min_{\mathbf{z}_j\in\mathcal{R}(\widehat{\mathbf{V}}_m^{(j)})} \left\|\mathbf{r}_0^{(j)} - \mathbf{A}\mathbf{z}_j\right\|. \tag{4.24}$$

Furthermore, we have that

$$\min_{\mathbf{z}_j\in\mathcal{R}(\widehat{\mathbf{W}}_M)} \left\|\mathbf{r}_0^{(j)} - \mathbf{A}\mathbf{z}_j\right\| \leq \min_{\mathbf{z}_j\in\mathcal{R}(\widehat{\mathbf{V}}_m^{(j)})} \left\|\mathbf{r}_0^{(j)} - \mathbf{A}\mathbf{z}_j\right\|. \tag{4.25}$$

Proof. Since we are running cycles of GMRES and block GMRES for the operator $(\mathbf{I} - \mathbf{C}\mathbf{C}^*)\mathbf{A}$ restricted to a subspace orthogonal to the null space, (4.24) follows directly from Theorem 4.1 and Theorem 22 from [90]. For the same reasons, (4.25) follows from Proposition 4.2. $\square$

In Figure 4.4, we compare the convergence of block GCRODR with the convergence curves presented in Figure 4.3. Note that this comparison cannot exactly capture the behavior described in Corollary 4.3, since the recycled subspaces are drawn from different Krylov subspaces.

4.3.2 Recycled Subspace Selection Techniques

In the current version of our codes, and also in the current version of the publicly available GCRODR codes [67], harmonic Ritz vectors; see, e.g., [56], [30], and [60]; are computed to generate a subspace to recycle. This is a sensible strategy. Eigenspaces which cause the most trouble for GMRES are those associated to eigenvalues near zero. Following from previous work [12, 96], Simoncini and Szyld, [84], showed that the convergence of GMRES accelerates, entering a superlinear phase, once the Krylov method has adequately captured a subspace spanned by eigenvectors associated to troublesome eigenvalues. Recall from Section 2.3 that the residual polynomial (2.10)

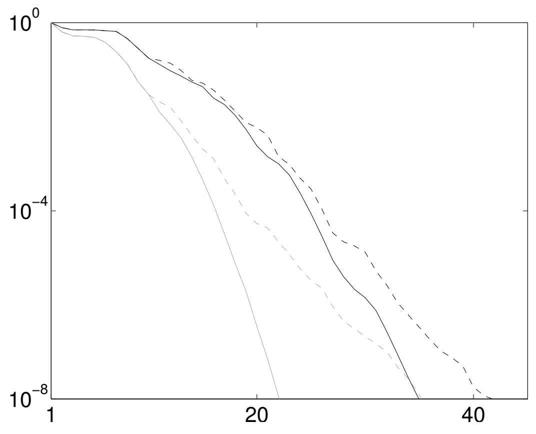


Figure 4.4. Demonstration of the convergence of block GCRODR(10,2) (**gray dot-dashed curve**) as compared to GCRODR(10,2) (**black dashed curve**), block GMRES (**solid grey curve**), and GMRES (**solid black curve**). For block GMRES and block GCRODR, a second right-hand side was generated using Matlab's `rand()`. Convergence is measured using only the residual norm for the first right-hand side. One matrix-block-vector product is executed at each iteration. Here we see that block GCRODR(10,2) converges with the same number of iterations as GMRES but uses much less storage.

at the mth iteration has roots approximating the eigenvalues of $\mathbf{A}$. However, this polynomial, $r_m(t)$ also has y-intercept $r_m(0) = 1$. Thus, if $\mathbf{A}$ has some eigenvalues near the origin, a Krylov subspace method will have difficulty constructing a good **low-degree** polynomial with roots approximating those eigenvalues which also has y-intercept of 1. Once this eigenspace is well-represented by the Krylov subspace, the convergence behavior mimics that of an operator from which the eigenspace has been removed, leading to the phase of superlinear convergence. This explains some of the impetus for the subspace augmentation and recycling technique; see e.g., [59] and [68]. By recycling a selected subspace and iterating orthogonally to it, we hope to achieve the superlinear convergence phase of GMRES more quickly.

Other choices are possible, and we review some of them here, but we restrict our experiments to harmonic Ritz vectors. Morgan suggests that in an eigenvector deflation algorithm based on FOM, called FOM-DR [59], deflation using Ritz vectors is more effective. Parks et al. [68] suggest that perhaps a mixture of Ritz and harmonic Ritz vectors may be appropriate in some cases. In his paper on optimal truncation methods [91], de Sturler demonstrates that one can calculate which subspace of dimension k of the current Krylov subspace of dimension m was most important against which to maintain orthogonality, for the purpose of reducing the residual. This subspace is then recycled under the assumption that it is most important to continue to maintain orthogonality with respect to this subspace. Ahuja et al.; [4], observed that the preconditioned systems with which they dealt had eigenvalue clusters well separated from the origin, rendering the use of harmonic Ritz vectors less effective. Instead, they chose to recycle Krylov vectors which had dominant components in the right-hand side, and this gave improved convergence results.

4.3.3 Harmonic Ritz Vector Computation

Though there are many ways to select a recycled space, we chose to use harmonic Ritz vector selection for our experiments, simply because numerical experience indicates this is a reliable default strategy. If we begin solving a system and possess no recycled space, we run a cycle of block GMRES and compute harmonic Ritz vectors in the block Krylov subspace. At the end of the cycle, we have generated an orthonormal basis for the subspace $\mathbb{K}_m(\mathbf{A}, \mathbf{R}_0)$ with the block Arnoldi relation $\mathbf{A}\mathbf{W}_m = \mathbf{W}_{m+1}\overline{\mathbf{H}}_m$ where $\overline{\mathbf{H}}_m \in \mathbb{C}^{(m+1)p \times mp}$ is **block** upper Hessenberg. The matrix $\overline{\mathbf{H}}_m$ contains in its first mp rows the square matrix $\mathbf{H}_m \in \mathbb{C}^{mp \times mp}$. In the last p rows of $\overline{\mathbf{H}}_m$, there is only one non-zero block, $\mathbf{H}_{m+1,m} \in \mathbb{C}^{p\times p}$, and it is upper triangular. Following [57], the block harmonic Ritz problem for $\mathbb{K}_m(\mathbf{A}, \mathbf{R}_0)$ is to find pairs $(\mathbf{y}, \mu)$ such that

$$\mathbf{A}^{-1}\mathbf{y} - \mu\mathbf{y} \perp \mathbf{A}\mathbb{K}_m(\mathbf{A}, \mathbf{R}_0) \ \text{ for all } \ \mathbf{y} \in \mathbf{A}\mathbb{K}_m(\mathbf{A}, \mathbf{R}_0). \tag{4.26}$$

Proposition 4.4: Given the block Krylov subspace $\mathbb{K}_m(\mathbf{A}, \mathbf{R}_0)$, solving the harmonic Ritz problem is equivalent to solving the $mp \times mp$ eigenvalue problem

$$(\mathbf{H}_m + (\mathbf{H}_m^*)^{-1}\widehat{\mathbf{E}}(\mathbf{H}_{m+1,m}^*\mathbf{H}_{m+1,m})\widehat{\mathbf{E}}^*)\mathbf{t} = \tilde{\theta}\mathbf{t} \tag{4.27}$$

and then for a solution pair $(\mathbf{t}_i, \tilde{\theta}_i)$ assigning $\mathbf{y}_i = \mathbf{W}_m\mathbf{t}_i$, where the columns of $\widehat{\mathbf{E}} \in \mathbb{C}^{mp\times p}$ are columns $(mp - p + 1), \ldots, mp$ identity matrix of order mp, and $\tilde{\theta} = 1/\mu$.

It should be noted that, as a practical matter, the expression

$$\mathbf{H}_m + (\mathbf{H}_m^*)^{-1}\widehat{\mathbf{E}}(\mathbf{H}_{m+1,m}^*\mathbf{H}_{m+1,m})\widehat{\mathbf{E}}^*$$

simply means that the last p columns of $\mathbf{H}_m$ should be modified by the $mp \times p$ matrix

$$(\mathbf{H}_m^*)^{-1}\widehat{\mathbf{E}}(\mathbf{H}_{m+1,m}^*\mathbf{H}_{m+1,m}).$$

Proof. We can prove this through algebraic manipulation using the block Arnoldi relation. The orthogonality condition (4.26) is equivalent to

$$\begin{aligned}
(\mathbf{A}\mathbf{W}_m)^*(\mathbf{A}^{-1}\mathbf{y} - \mu\mathbf{y}) &= 0 \qquad (4.28)\\
(\mathbf{W}_{m+1}\overline{\mathbf{H}}_m)^*(\mathbf{A}^{-1}\mathbf{A}\mathbf{W}_m\mathbf{t} - \mu\mathbf{A}\mathbf{W}_m\mathbf{t}) &= 0\\
\overline{\mathbf{H}}_m^*\mathbf{W}_{m+1}^*(\mathbf{W}_m\mathbf{t} - \mu\mathbf{W}_{m+1}\overline{\mathbf{H}}_m\mathbf{t}) &= 0\\
\overline{\mathbf{H}}_m^*\mathbf{W}_{m+1}^*\mathbf{W}_m\mathbf{t} &= \mu\overline{\mathbf{H}}_m^*\overline{\mathbf{H}}_m\mathbf{t}\\
\mathbf{H}_m^*\mathbf{t} &= \mu(\mathbf{H}_m^*\mathbf{H}_m + \widehat{\mathbf{E}}(\mathbf{H}_{m+1,m}^*\mathbf{H}_{m+1,m})\widehat{\mathbf{E}}^*)\mathbf{t}\\
\tilde{\theta}\mathbf{t} &= (\mathbf{H}_m + (\mathbf{H}_m^*)^{-1}\widehat{\mathbf{E}}(\mathbf{H}_{m+1,m}^*\mathbf{H}_{m+1,m})\widehat{\mathbf{E}}^*)\mathbf{t} \qquad \square
\end{aligned}$$

This can be seen as a generalization of the harmonic Ritz computation in the case of a single-vector Krylov subspace, see, e.g., [58].

The computation in the case of the augmented subspace $\mathcal{R}(\widehat{\mathbf{W}}_M)$ is similar to the computation employed in the GCRODR [68].

Proposition 4.5: In a cycle of block GCRODR, if we have generated an augmented space $\mathcal{R}(\widehat{\mathbf{W}}_M)$, as in (4.17), then solving the associated harmonic Ritz problem is equivalent to solving the generalized eigenvalue problem

$$\overline{\mathbf{G}}_M^*\overline{\mathbf{G}}_M\mathbf{t} = \tilde{\theta}\overline{\mathbf{G}}_M^*\widetilde{\mathbf{W}}_{M+p}^*\widehat{\mathbf{W}}_M\mathbf{t} \qquad (4.29)$$

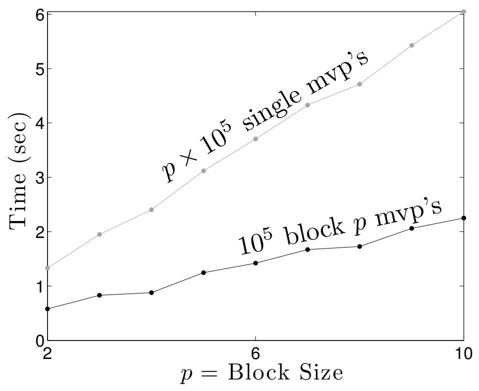
and then for a solution pair $(\mathbf{t}_i, \tilde{\theta}_i)$ assigning $\mathbf{y}_i = \widehat{\mathbf{W}}_M\mathbf{t}_i$.

Proof. The proof is identical to that in [68]. $\square$

4.3.4 Experiments

We implemented the block GCRODR algorithm in Matlab. We have also implemented and deployed block GCRODR into the Belos package of Sandia's Trilinos Project [2]. Our intention is to eventually incorporate this solver into the Tramonto DFT solver and run large scale simulations on a parallel machine. In the meantime, though, we were able to get promising preliminary results with our efficient Matlab implementation.

One point which must be discussed is how to compare the performance of the block GCRODR to algorithms that execute only a matrix-single-vector product per iteration. Iteration-for-iteration, block methods and single-vector methods have different costs for the dominant operation in the iteration, the matrix-vector product. However, a block size p matrix-vector product does not cost p times as much as a single matrix-vector product. In Figure 4.5, we demonstrate timing comparisons for performing single and block matrix-vector products for computing the action of a sparse matrix $\mathbf{A}$ on equal numbers of vectors. In this scenario, we see that the block matrix-vector product is able to outperform the single matrix-vector product in this case, in Matlab. However, we are more interested in an iteration for iteration performance comparison of a block Krylov subspace method versus a single-vector Krylov subspace method. We pose the question, do the benefits of convergence in fewer iterations outweigh the costs of the more expensive block matrix-vector product? In Figure 4.6, we compare the the time taken to perform matrix-single-vector products with the time take to compute matrix-block-vector products. We



mvp - matrix-vector product

Figure 4.5. Comparison of the time taken to perform $p \times 10^5$ matrix-vector products for the sparse `sherman5` matrix for the single matrix-vector product (**grey**) and the block size p matrix-vector product (**black**). Different block sizes were tested. The initial target vector was generated using Matlab's `rand()` function, and the matrix simply acted upon this vector repeatedly.

see that, though block matrix-vector products (block size $p > 1$) are more expensive to compute than the single-vector variety, they are not p times as expensive.

In Figure 4.7, we test the code's convergence properties as we increase the number of right-hand sides. As is predicted by the underlying theory, the increased number of right-hand sides generates a richer space from which to select our approximation updates and from which to recycle, though the benefit decreases for each additional right-hand side.

We extracted matrices from seven consecutive iterations of a Tramonto Newton iteration from the `POLY1_CMS_1D` test problem [3]. For each iteration, we precondition using `ILU(0)`. We compare the performance of GMRES, block GMRES with 3 right-hand sides, GCRODR, and our block GCRODR algorithm on all 7 systems. In the case of the block methods, one right-hand side generated by the Tramonto

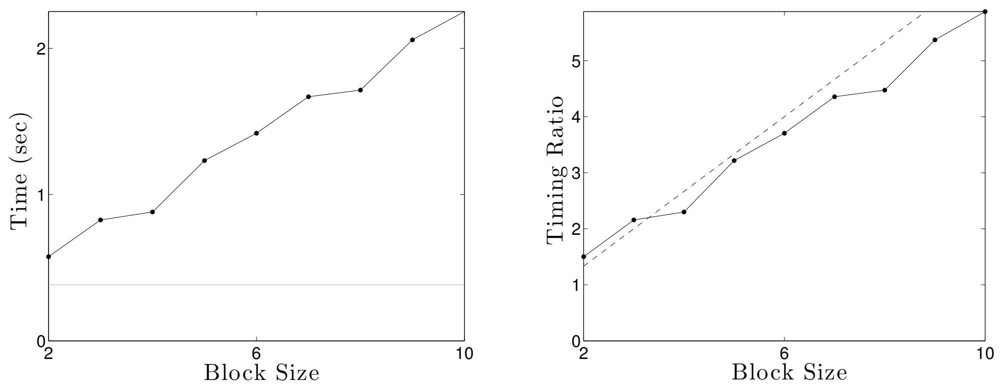


Figure 4.6. In the figure on the left, we compare the time taken to perform 10^5 single matrix-vector products (**grey**) with how much time is taken to perform the same number of block matrix-vector products for different block sizes p (**black**). The figure on the right shows the ratio of time taken for the block matrix- vector products over the time taken for single matrix-vector products (**black solid line**). As predicted, the block matrix-vector product is not p times as expensive. The **dashed line** shows the ratio if a block p matrix-vector product was $\frac{2}{3}p$ times as expensive as a single matrix-vector product.

package, and the other two right-hand sides were random, generated using Matlab's `rand()`. In the case of these Newton iterations for these relatively small systems (dimension $\approx$ 14000), convergence for the preconditioned system is quick enough that we are able to recycle the entire Krylov subspace when running GCRODR and block GCRODR for the first few systems in the sequence for properly chosen recycled subspace dimension. In Figure 4.8, we plot the number of matrix-vector products needed to solve each system. Observe that both for GCRODR and block GCRODR, recycling greatly reduces the number of matrix-vector products needed to solve later systems. Furthermore, we get a per-system reduction when moving from GCRODR to block GCRODR, particularly for systems appearing early in the sequence. In

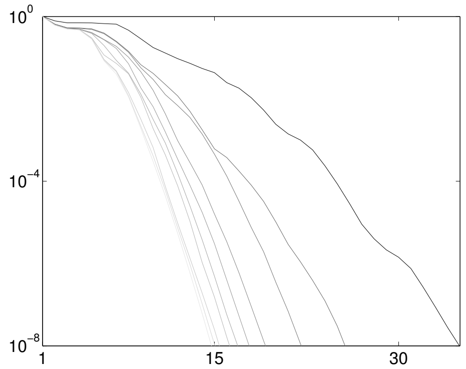


Figure 4.7. Performance of block GCRODR(100,50) on the `sherman5` matrix from Matrix Market [1] preconditioned with `ILU(0)`as we increase the number of right-hand sides. The first right-hand side is packaged with the matrix. The other right-hand sides are generated using Matlab's `rand()` function. Convergence is measured by computing the residual norm associated to the first right-hand side. Lightness of the convergence curves increases as we add more right-hand sides.

Figure 4.8, the benefit of the block method is most pronounced for the first system. This suggests that for some problems, we may gain the most benefit by applying block GCRODR only for the first system. This will yield a high-quality recycled subspace, and we apply single-vector GCRODR for the rest of the systems. We also see that, for large enough recycled subspace, we achieve a 30% reduction in overall matrix-vector products when moving from single right-hand side GCRODR to block GCRODR with two random right-hand sides.

4.3.5 Conclusion

We have presented an extension of existing subspace recycling techniques to work with block Krylov subspaces. We have presented numerical experiments demonstrating the efficacy of the method for a sequence of linear systems arising in computations

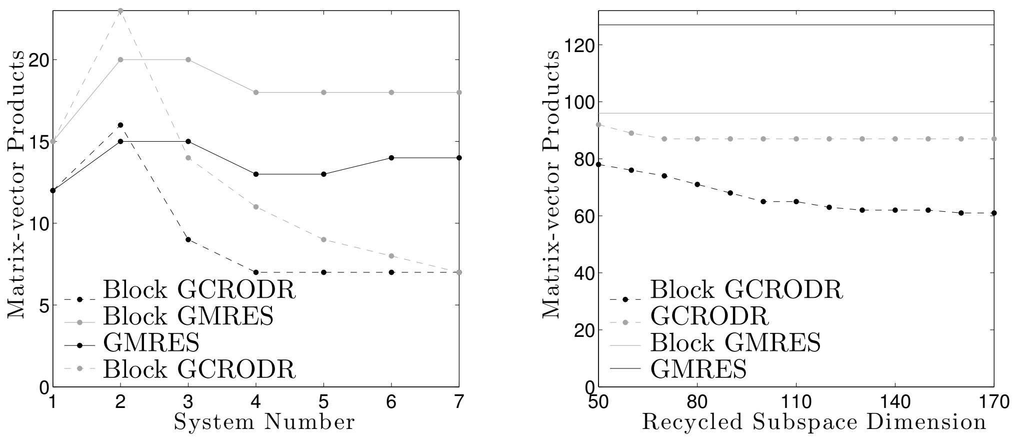


Figure 4.8. In the left-hand figure, we see the matrix-vector product count for block GCRODR solving linear systems involving the Jacobian for 7 sequential Newton steps of fluid DFT problem from Tramonto software package. ILU(0) preconditioning was used. The figure on the right shows the total matrix-vector product count for various recycle space dimensions Effectively, as we move to the right in this figure, we reach a point at which no recycled subspace information is ever discarded in the experiment.

from fluid density functional theory. Furthermore, we have presented some simple convergence results comparing the convergence of block GCRODR to that of single-vector GCRODR under certain assumptions.

4.4 GMRES with Subspace Recycling for Multiple Shifted Linear Systems

4.4.1 Introduction

We now shift to our our second project to adapt subspace recycling technology to solve sequences of families of linear systems where the coefficient matrices for the ith family come from the family $\mathcal{F}_i$, as defined in (4.3). The solution of such collections of systems is an important problem which arises, e.g., in computations associated to problems of lattice quantum chromodynamics (QCD) [34], an application from particle physics. It is thought that the theory of QCD completely describes the strong interaction between quarks. Lattice QCD is a discretized version of the theory living on a four dimensional space-time lattice.

For simplicity of discussion, we will make two assumptions. We will focus on one family of linear systems, and drop the index i. Second, without loss of generality, we can assume one matrix in the family $\mathcal{F}$ involves the base coefficient matrix with no shift. This yields the set of equations

$$\begin{aligned} \mathbf{A}\mathbf{x} &= \mathbf{b} && (4.30) \\ (\mathbf{A} + \sigma_j \mathbf{I})\mathbf{x}^{(\sigma_j)} &= \mathbf{b}, \quad \text{with} \quad j = 1, \ldots, L && (4.31) \end{aligned}$$

The coefficient matrices for these problems tend to be large (on the order of 10^7 or higher for QCD). For such problems, Krylov subspace methods are effective and widely used. One method for generating approximations for (4.30) and (4.31) would be to solve the systems sequentially using a Krylov subspace methods applied to each system. However, as we already mentioned in Section 2.3, this would not necessarily be the most efficient way to compute the solution. For nonsymmetric problems, each iteration of a Krylov method such as GMRES [76] requires the storage of an additional vector. For problems of this size, we are severely limited in the number of vectors we can store. Storage-efficient methods, such as restarted methods, are quite attractive for limiting memory usage. Further savings can be achieved by not solving (4.30) and (4.31) sequentially. In fact, note that the Krylov subspace generated by

$\mathbf{A}$ and $\mathbf{b}$, $\mathcal{K}_j(\mathbf{A}, \mathbf{b})$, is invariant under any shift, σ, i.e.,

$$\mathcal{K}_j(\mathbf{A}, \mathbf{b}) = \mathcal{K}_j(\mathbf{A} + \sigma\mathbf{I}, \mathbf{b}). \tag{4.32}$$

This suggests that we can generate a single Krylov subspace to solve the base system (4.30) and the shifted system (4.31). This shift invariance property indicates that restarted methods may be suitable for solving the equations of (4.30) and (4.31) simultaneously. However, we have already discussed some of the problems with restarted methods in Section 2.3, leading to our use of subspace recycling.

Note that the shift-invariance no longer holds if general preconditioning is used. It does hold, though, if the preconditioner commutes with the coefficient matrix $\mathbf{A}$. For example, polynomial preconditioning [49] would be appropriate in this setting. In general, any preconditioner which is simultaneously diagonalizable with $\mathbf{A}$ would preserve the shift invariance property [47, Theorem 1.3.19]. In this project, we focus on the unpreconditioned case, as in [33] and [59].

In the next section, we review some existing methods for solving (4.30) and (4.31). In Section 4.4.3, we introduce a new algorithm which combines ideas for solving (4.30) and (4.31) simultaneously while using subspace recycling techniques. This method, though, requires additional storage for each new shift. There are difficulties in applying subspace recycling techniques in this setting without requiring additional storage per shift, and we present a scheme in Section 4.4.4, which mitigates this problem by enforcing an **approximation** of the conditions presented in the algorithm from Section 4.4.3. This method is based on the shifted GMRES method [33]. However, for certain choices of the shift, the shifted GMRES algorithm is not guaranteed to produce approximations for this shift. Our method does not suffer from this problem. In Section 4.4.5, we present numerical results for a sequence of sample QCD matrices taken from [1], and for a discretized convection-diffusion equation.

4.4.2 Background

For any Krylov method, one can describe a restarted version of that method. The memory needed to store $\mathbf{V}_j$ for GMRES increases with j, and a restart is necessary when the size of $\mathbf{V}_j$ exceeds the limit of available memory. Recall that in restarted GMRES, often called GMRES(m), we run an m-step cycle of the GMRES method and compute an approximation $\mathbf{x}_m$. We then discard all Arnoldi vectors and use $\mathbf{x}_m$ as the initial approximation for the next cycle of GMRES. This process is repeated until we achieve convergence. Adaptions of restarted methods to solve (4.30) and (4.31) using the invariance property (4.32) have been previously proposed; see, e.g., [33],[80]. The extension of another type of non-optimal, short-term recurrence Krylov subspace method, BiCGStab, has also been proposed [32]. A recently proposed method called IDR [87], which has been shown to be a generalization of BiCGStab [86], has also recently been extended to solve (4.30) and (4.31) [53]. Since all the strategies discussed can be applied repeatedly for each additional shift, without loss of generality, it is enough to consider a single shift for the purposes of derivation. Therefore, unless otherwise indicated, we will restrict our discussion to a system with two equations, the base system and one shifted system with shifted $\sigma \in \mathbb{C}$.

Many methods use the fact that for any shift σ, the Krylov subspace generated by $\mathbf{A}$ and $\mathbf{b}$ is invariant under the shift, with a shifted Arnoldi relation similar to (2.15)

$$(\mathbf{A}+\sigma\mathbf{I})\mathbf{V}_m = \mathbf{V}_{m+1}\overline{\mathbf{H}}_m^{(\sigma)}. \tag{4.33}$$

This indicates that large savings in storage and time can be achieved by generating only one sequence of Krylov subspaces. At each step, we solve all shifted systems in one Krylov subspace simultaneously. We describe the algorithm proposed in [33] in more detail.

Frommer and Glässner [33] proposed a restarted GMRES method to solve (4.30) and (4.31) . Suppose, for the base system and the shifted systems, we have the same initial approximation $\mathbf{x}_0 = \mathbf{x}_0^{(\sigma)} = \mathbf{0}$. It is observed that, unlike restarted FOM [80], after applying a cycle of GMRES(m) to both systems, the residual of the base system and the shifted system will not be collinear. Upon restart, the systems will no longer share the same Krylov subspace. Therefore, a restarted GMRES method to solve (4.30) and (4.31) will require that we enforce collinearity upon the residuals at the end of each cycle.

This is what the authors propose in [33]. Suppose that our residuals for the shifted and base system satisfy the relationship

$$\mathbf{r}_0^{(\sigma)} = \beta_0\mathbf{r}_0 \tag{4.34}$$

at the beginning of a cycle. For the base system, we can represent the residual at the mth step as $\mathbf{r}_m = r_m(\mathbf{A})\mathbf{r}_0$ where $r_m(t)$ is a polynomial of degree less than or equal to m such that $r(0) = 1$. We enforce the condition that the residual for the shifted system is in the same direction as the residual of the base system, i.e.,

$$\mathbf{r}_m^{(\sigma)} = \beta_m\mathbf{r}_m. \tag{4.35}$$

By straightforward computations in [33], it is shown that if $\mathbf{r}_m$ is the mth GMRES residual and $\mathbf{z}_{m+1} = \|\mathbf{r}_0\| \mathbf{e}_1 - \overline{\mathbf{H}}_m \mathbf{y}_m$, then for (4.35) to hold, we must have

$$\overline{\mathbf{H}}_m^{(\sigma)} \mathbf{y}_m^{(\sigma)} + \mathbf{z}_{m+1}\beta_m = \beta_0 \|\mathbf{r}_0\| \mathbf{e}_1, \tag{4.36}$$

and $\mathbf{x}_m^{(\sigma)} = \mathbf{x}_0^{(\sigma)} + \mathbf{V}_m \mathbf{y}_m^{(\sigma)}$ Thus, we can compute both $\mathbf{y}_m^{(\sigma)}$ and β_m by solving the augmented linear system,

$$\begin{bmatrix} \overline{\mathbf{H}}_m^{(\sigma)} & \mathbf{z}_{m+1} \end{bmatrix} \begin{bmatrix} \mathbf{y}_m^{(\sigma)} \\ \beta_m \end{bmatrix} = \beta_0 \|\mathbf{r}_0\| \mathbf{e}_1. \tag{4.37}$$

Note that we only compute the minimum residual solution using a Petrov-Galerkin condition for the base system. We enforce collinearity of the residual to compute the approximation for the shifted system. We can equivalently think of this as applying an oblique rather than an orthogonal projection to the residual of the shifted system. It is shown in [33, Lemmas 2.1 and 2.4] that a solution to (4.37) exists if and only if the residual polynomial $r_m(t)$ satisfies $r_m(-\sigma) \neq 0$; otherwise, the augmented system is singular. Therefore, this condition on (4.30) can easily be verified by looking at the last row of $\mathbf{R}_{m+1}$ and investigating if the diagonal entry is close to zero, indicating numerical singularity, where $\begin{bmatrix} \overline{\mathbf{H}}_m^{(\sigma)} & \mathbf{z}_{m+1} \end{bmatrix} = \mathbf{Q}_{m+1}\mathbf{R}_{m+1}$ is the QR-factorization. In the case that the solution does not exist, we simply perform one more iteration and check this condition again. It is pointed out in [33], though, that for real-positive matrices (all eigenvalues have positive real part), restarted GMRES for shifted linear systems computes solutions at every iteration for all shifts $\sigma^{(i)} > 0$. The shifts applied in the setting of QCD yield a family of coefficient matrices which are usually real-positive [33]. However, it has been pointed out that statistical aberrations arising from the generation of these matrices by Monte Carlo methods can result in the need to solve systems which are not real-positive, even when application of the shifts should, in theory, yield only real-positive systems [32]. This is an efficient method for solving families of shifted linear systems, but it is based on restarted GMRES, meaning it inherits all of its deficiencies. In Figure 4.9, we demonstrate the convergence of the shifted GMRES method for simultaneously solving a family of five shifted linear systems.

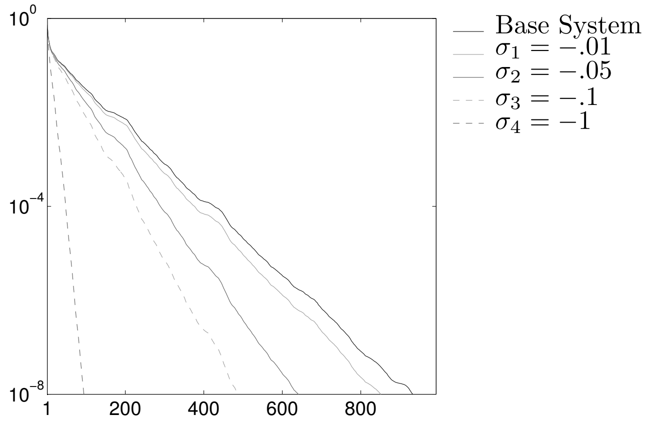


Figure 4.9. The simultaneous solution of a base system and four shifted systems using the shifted GMRES method. The base system $\mathbf{A}$ is constructed from the matrix `conf5.0-00l4x4-1000` from Matrix Market [1], which we call $\mathbf{D}$. We form $\mathbf{A} = \mathbf{D} - (1/\kappa_c + .001)\mathbf{I}$ for a value κ_c provided with the matrix. The matrix $\mathbf{A}$ is real-positive. GMRES(30) is used to solve the base system, and the shifted GMRES method produces approximations for the four shifted systems. The shifts $\{\sigma_i\}_{i=1}^4$ are listed in the convergence plot.

An attractive idea would be to combine Frommer and Glässner's method with a subspace augmentation or deflation method. Morgan's GMRES-DR [59] is an attractive candidate, and in recent work [19], this method has been extended to solve

(4.30) and (4.31). However, it is restricted to the use of approximate invariant subspaces spanned by harmonic Ritz vectors. We cannot select other subspaces using different criteria. As a result, we cannot use the recycled subspace from one linear system to deflate to the next one. In a situation in which we are running multiple QCD simulations, we would like to accelerate convergence by deflating between systems.

4.4.3 GCRODR For Shifted Systems

The GMRES method for shifted systems [33] is a method for solving (4.30) and (4.31) that computes the minimum residual solution for the base system at the end of each cycle, and then computes solutions in the same subspace for the shifted system such that the residual of the shifted system is collinear with the residual of the base system. However, since it is a method based on restarted GMRES, it inherits the unpredictable convergence properties discussed in Section 2.3. Subspace recycling has been shown to greatly improve the convergence of restarted methods, in many cases, without dramatically increasing the memory costs. Therefore, if we can incorporate GMRES for shifted linear systems into the recycling framework described in [68] we will have a storage-efficient method which will solve all shifted systems simultaneously but is less likely to suffer from problems which plague restarted GMRES. We will denote this method *GCRODR for shifted systems*. In terms of extending the work of Frommer and Glässner, this is our ideal method. However, we will show that such a method is memory-consuming for a large number of shifted systems. It may be useful when there are only a small number of shifts, and it is the basis from which we develop a more practical algorithm in Section 4.4.4.

Consider the pair of linear systems (4.30) and (4.31). For initial approximations, we choose $\tilde{\mathbf{x}}_0$ and $\tilde{\mathbf{x}}_0^{(\sigma)}$ so that the initial residuals are collinear, i.e., $\tilde{\mathbf{r}}_0 = \tilde{\beta}_0\tilde{\mathbf{r}}_0^{(\sigma)}$. In GCRODR, the initial projection of the residual (4.5) and resulting update of the solution rely on the relationship (4.4). This means that even though we can orthogonally project the residual of the shifted system onto $\mathcal{R}(\mathbf{C})^{\perp}$, we cannot easily update the approximation associated to the shifted system. In order to project the residual for the shifted system, we would need $\mathbf{U}^{(\sigma)}$ such that

$$\mathbf{C} = (\mathbf{A} + \sigma\mathbf{I})\mathbf{U}^{(\sigma)}. \tag{4.38}$$

This would allow us to project the residual for the shifted system and update the shifted solution using (4.38) in the same manner that we use (4.4) to project the residual for the base system. This will require an additional k vectors of storage for each shift. We will show that we can obtain $\mathbf{U}^{(\sigma)}$ without solving (4.38).

Let us suppose for the moment that for a given $\mathbf{C} \in \mathbb{C}^{n\times k}$ with orthonormal columns, we already have matrices $\mathbf{U}$ and $\mathbf{U}^{(\sigma)}$ satisfying the relationships (4.4) and (4.38), respectively. We make a simple observation about the relationship between the subspaces $\mathcal{K}_m((\mathbf{I}-\mathbf{C}\mathbf{C}^*)\mathbf{A},\mathbf{v})$ and $\mathcal{K}_m((\mathbf{I} - \mathbf{C}\mathbf{C}^*)(\mathbf{A} + \sigma\mathbf{I}),\mathbf{v})$, for any vector $\mathbf{v}$ orthogonal to the range of $\mathbf{C}$.

Proposition 4.6: Let $\mathbf{C}$ be a matrix with orthonormal columns. If $\mathbf{v}$ is in the orthogonal complement of the range of $\mathbf{C}$, i.e., $\mathbf{C}\mathbf{C}^*\mathbf{v} = \mathbf{0}$, then we have

$$\mathcal{K}_m((\mathbf{I}-\mathbf{C}\mathbf{C}^*)\mathbf{A},\mathbf{v}) = \mathcal{K}_m((\mathbf{I}-\mathbf{C}\mathbf{C}^*)(\mathbf{A}+\sigma\mathbf{I}),\mathbf{v}) \tag{4.39}$$

Proof. Observe that since $\mathbf{CC}^*\mathbf{v} = \mathbf{0}$, we have

$$(\mathbf{I} - \mathbf{CC}^*)(\mathbf{A} + \sigma\mathbf{I})\mathbf{v} = (\mathbf{I} - \mathbf{CC}^*)\mathbf{Av} + \sigma(\mathbf{I} - \mathbf{CC}^*)\mathbf{v} = (\mathbf{I} - \mathbf{CC}^*)\mathbf{Av} + \sigma\mathbf{v}.$$

Therefore, when restricted to vectors orthogonal to the range of $\mathbf{C}$, we have that

$$(\mathbf{I} - \mathbf{CC}^*)(\mathbf{A} + \sigma\mathbf{I}) = (\mathbf{I} - \mathbf{CC}^*)\mathbf{A} + \sigma\mathbf{I}.$$

Furthermore, since any $\mathbf{u} \in \mathcal{R}((\mathbf{I} - \mathbf{CC}^*)(\mathbf{A} + \sigma\mathbf{I}))$ is orthogonal to the range of $\mathbf{C}$, we have

$$[(\mathbf{I} - \mathbf{CC}^*)(\mathbf{A} + \sigma\mathbf{I})]^j = [(\mathbf{I} - \mathbf{CC}^*)\mathbf{A} + \sigma\mathbf{I}]^j$$

when applied to any $\mathbf{v} \perp \mathcal{R}(\mathbf{C})$. Thus,

$$\mathcal{K}_m((\mathbf{I} - \mathbf{CC}^*)(\mathbf{A} + \sigma\mathbf{I}), \mathbf{v}) = \mathcal{K}_m((\mathbf{I} - \mathbf{CC}^*)\mathbf{A} + \sigma\mathbf{I}, \mathbf{v}) = \mathcal{K}_m((\mathbf{I} - \mathbf{CC}^*)\mathbf{A}, \mathbf{v}),$$

where the last equality follows from the shift invariance property of Krylov subspaces. ☐

Thus, if we construct an augmented subspace for the base system and minimize the residual as described in [68], this will be the same subspace which would be constructed when using GCRODR to solve the shifted system. Furthermore, as in [33], at the end of a cycle we can minimize the residual over the augmented subspace using GCRODR, and then enforce a collinearity condition on the residual of the shifted system to simultaneously compute an approximation for the shifted system. Recall that, we have the two Arnoldi relations

$$\mathbf{AV}_m = \mathbf{V}_{m+1}\overline{\mathbf{H}}_m \text{ and } (\mathbf{A} + \sigma\mathbf{I})\mathbf{V}_m = \mathbf{V}_{m+1}\overline{\mathbf{H}}_m^{(\sigma)},$$

respectively for the base and shifted systems, where $\overline{\mathbf{H}}_m^{(\sigma)} = \overline{\mathbf{H}}_m + \begin{bmatrix} \sigma \mathbf{I}_m \\ \mathbf{0} \end{bmatrix}$. From here it is straightforward to construct an Arnoldi-like relation for the shifted system,

$$(\mathbf{A} + \sigma \mathbf{I}) \begin{bmatrix} \mathbf{U}^{(\sigma)} & \mathbf{V}_{m-k} \end{bmatrix} = \begin{bmatrix} \mathbf{C} & \mathbf{V}_{m-k+1} \end{bmatrix} \begin{bmatrix} \mathbf{I}_k & \mathbf{B} \\ \mathbf{0} & \overline{\mathbf{H}}_{m-k}^{(\sigma)} \end{bmatrix}. \tag{4.40}$$

We note, the same matrix $\mathbf{B}$ appears in (4.40) as in (4.7) since

$$\begin{aligned} (\mathbf{I} - \mathbf{C}\mathbf{C}^*)(\mathbf{A} + \sigma \mathbf{I})\mathbf{V}_{m-k} &= \mathbf{V}_{m-k+1}\overline{\mathbf{H}}_{m-k}^{(\sigma)} \\ (\mathbf{A} + \sigma \mathbf{I})\mathbf{V}_{m-k} - \mathbf{C}\mathbf{C}^*(\mathbf{A} + \sigma \mathbf{I})\mathbf{V}_{m-k} &= \mathbf{V}_{m-k+1}\overline{\mathbf{H}}_{m-k}^{(\sigma)} \\ (\mathbf{A} + \sigma \mathbf{I})\mathbf{V}_{m-k} - \mathbf{C}\mathbf{C}^*\mathbf{A}\mathbf{V}_{m-k} + \sigma \mathbf{C}\mathbf{C}^*\mathbf{V}_{m-k} &= \mathbf{V}_{m-k+1}\overline{\mathbf{H}}_{m-k}^{(\sigma)} \\ (\mathbf{A} + \sigma \mathbf{I})\mathbf{V}_{m-k} - \mathbf{C}\mathbf{C}^*\mathbf{A}\mathbf{V}_{m-k} &= \mathbf{V}_{m-k+1}\overline{\mathbf{H}}_{m-k}^{(\sigma)} \\ (\mathbf{A} + \sigma \mathbf{I})\mathbf{V}_{m-k} &= \mathbf{C}\mathbf{B} + \mathbf{V}_{m-k+1}\overline{\mathbf{H}}_{m-k}^{(\sigma)} \quad (4.41) \end{aligned}$$

where $\mathbf{B} = \mathbf{C}^*\mathbf{A}\mathbf{V}_{m-k}$ and where we used the fact that $\sigma \mathbf{C}\mathbf{C}^*\mathbf{V}_{m-k} = \mathbf{0}$. This is due to the fact that the columns of $\mathbf{V}_{m-k}$ are orthogonal to $\mathcal{R}(\mathbf{C})$. We can follow the same arguments as in [33] and construct an approximation for the shifted system such that the residual is collinear to the one produced by GCRODR for the base system,

$$\begin{aligned}
\mathbf{r}_m^{(\sigma)} &= \beta_m \mathbf{r}_m \qquad (4.42)\\
\mathbf{b} - (\mathbf{A} + \sigma\mathbf{I})\left(\mathbf{x}_0^{(\sigma)} + \begin{bmatrix}\mathbf{U}^{(\sigma)} & \mathbf{V}_{m-k}\end{bmatrix}\mathbf{y}_m^{(\sigma)}\right) &= \beta_m \begin{bmatrix}\mathbf{C} & \mathbf{V}_{m-k+1}\end{bmatrix}\mathbf{z}_m\\
\mathbf{r}_0^{(\sigma)} - \begin{bmatrix}\mathbf{C} & \mathbf{V}_{m-k+1}\end{bmatrix}\begin{bmatrix}\mathbf{I}_k & \mathbf{B}\\ \mathbf{0} & \overline{\mathbf{H}}_{m-k}^{(\sigma)}\end{bmatrix}\mathbf{y}_m^{(\sigma)} &= \beta_m \begin{bmatrix}\mathbf{C} & \mathbf{V}_{m-k+1}\end{bmatrix}\mathbf{z}_m\\
\beta_0\mathbf{r}_0 - \begin{bmatrix}\mathbf{C} & \mathbf{V}_{m-k+1}\end{bmatrix}\begin{bmatrix}\mathbf{I}_k & \mathbf{B}\\ \mathbf{0} & \overline{\mathbf{H}}_{m-k}^{(\sigma)}\end{bmatrix}\mathbf{y}_m^{(\sigma)} &= \beta_m \begin{bmatrix}\mathbf{C} & \mathbf{V}_{m-k+1}\end{bmatrix}\mathbf{z}_m\\
\beta_0\left\|\mathbf{r}_0\right\|\mathbf{e}_{k+1}^{(m+1)} - \begin{bmatrix}\mathbf{I}_k & \mathbf{B}\\ \mathbf{0} & \overline{\mathbf{H}}_{m-k}^{(\sigma)}\end{bmatrix}\mathbf{y}_m^{(\sigma)} &= \beta_m\mathbf{z}_m,
\end{aligned}$$

which yields the linear system

$$\begin{bmatrix}\overline{\mathbf{G}}_m^{(\sigma)} & \mathbf{z}_m\end{bmatrix}\begin{bmatrix}\mathbf{y}_m^{(\sigma)}\\ \beta_m\end{bmatrix} = \beta_0\left\|\mathbf{r}_0\right\|\mathbf{e}_{k+1}^{(m+1)}, \qquad (4.43)$$

where

$$\overline{\mathbf{G}}_m^{(\sigma)} = \begin{bmatrix}\mathbf{I}_k & \mathbf{B}\\ \mathbf{0} & \overline{\mathbf{H}}_{m-k}^{(\sigma)}\end{bmatrix}.$$

We then update

$$\mathbf{x}_m^{(\sigma)} = \mathbf{x}_0^{(\sigma)} + \begin{bmatrix}\mathbf{U}^{(\sigma)} & \mathbf{V}_{m-k}\end{bmatrix}\mathbf{y}_m^{(\sigma)}.$$

As it can be appreciated, this is a straightforward extension of the work of Frommer and Glässner [33].

We now return to the question of practicality. Given a matrix $\widetilde{\mathbf{U}} \in \mathbb{C}^{n\times k}$ with columns spanning our recycled space, how do we compute $\mathbf{U}$ and $\mathbf{U}^{(\sigma)}$ such that both

$$\mathbf{A}\mathbf{U} = \mathbf{C} \quad \text{and} \quad (\mathbf{A} + \sigma\mathbf{I})\mathbf{U}^{(\sigma)} = \mathbf{C} \qquad (4.44)$$

hold, with $\mathbf{C}^*\mathbf{C} = \mathbf{I}_k$? In GCRODR, we compute $\mathbf{A}\widetilde{\mathbf{U}} = \widetilde{\mathbf{C}}$. Then we compute the QR-factorization $\widetilde{\mathbf{C}} = \mathbf{C}\mathbf{R}$ and let $\mathbf{U} = \widetilde{\mathbf{U}}\mathbf{R}^{-1}$. One way to interpret this process is to consider that to solve one system, we need one recycled space satisfying (4.4) which costs k matrix-vector products plus one basis orthogonalization to acquire.

Now suppose we have two coefficient matrices $\mathbf{A}$ and $(\mathbf{A} + \sigma\mathbf{I})$. If we have $\mathbf{U}$ such that $\mathbf{A}\mathbf{U} = \mathbf{C}$, there is no efficient way to acquire $\mathbf{U}^{(\sigma)}$. We cannot compute the image of $\widetilde{\mathbf{U}}$ under $\mathbf{A}$ and $\mathbf{A}+\sigma\mathbf{I}$ separately because this would not yield one matrix $\mathbf{C}$. To get a pair of matrices $\mathbf{U}$ and $\mathbf{U}^{(\sigma)}$ satisfying (4.44), we premultiply $\widetilde{\mathbf{U}}$ by both $\mathbf{A}$ and $\mathbf{A}+\sigma\mathbf{I}$. In other words, let $\widetilde{\mathbf{C}} = \mathbf{A}(\mathbf{A}+\sigma\mathbf{I})\widetilde{\mathbf{U}}$ and compute the QR-factorization $\widetilde{\mathbf{C}} = \mathbf{C}\mathbf{R}$. Then we have $\mathbf{U} = (\mathbf{A} + \sigma\mathbf{I})\widetilde{\mathbf{U}}\mathbf{R}^{-1}$ and $\mathbf{U}^{(\sigma)} = \mathbf{A}\widetilde{\mathbf{U}}\mathbf{R}^{-1}$. Since matrices differing by multiples of the identity commute, we have $\mathbf{A}\mathbf{U} = (\mathbf{A} + \sigma\mathbf{I})\mathbf{U}^{(\sigma)} = \mathbf{C}$. This easily can be generalized to any number of shifts. Given a sequence of shifts $\{\sigma^i\}_{i=1}^{s+1} \subset \mathbb{C}$, where we consider the base matrix as a system with shift $\sigma_1 = 0$, the matrices

$$\mathbf{U}^{(\sigma_j)} = \left[\prod_{i=1, i\neq j}^{s+1} (\mathbf{A} + \sigma_i)\right] \widetilde{\mathbf{U}}\mathbf{R}^{-1} \quad \text{and} \quad \mathbf{C} = \left[\prod_{i=1}^{s+1} (\mathbf{A} + \sigma_i)\right] \widetilde{\mathbf{U}}\mathbf{R}^{-1}$$

satisfy our requirements with $\widetilde{\mathbf{C}} = \mathbf{C}\mathbf{R}$ being the QR-factorization of $\widetilde{\mathbf{C}}$. Since matrices differing by a multiple of the identity commute, we have that $(\mathbf{A}+\sigma_i\mathbf{I})\mathbf{U}^{(\sigma_i)} = \mathbf{C}$ for all i. This strategy has clear stability issues which must be addressed. Furthermore, we wish to execute as few calls to the operator $\mathbf{A}$ as possible. Therefore, we propose a more stable implementation of this procedure.

Though these matrices do commute, once we choose a particular multiplication order, e.g., if $\mathbf{C} = \mathbf{A}(\mathbf{A} + \sigma\mathbf{I})\widetilde{\mathbf{U}}\mathbf{R}^{-1}$, we must proceed intelligently to recover

$\mathbf{U}^{(\sigma)} = \mathbf{A}\widetilde{\mathbf{U}}\mathbf{R}^{-1}$ without doing additional matrix vector products. To deal with this issue, we first introduce a straightforward proposition whose proof follows by simple algebraic manipulation.

Proposition 4.7: Let $\widehat{\mathbf{U}} = (\mathbf{A} + \sigma_1\mathbf{I})\widetilde{\mathbf{U}}$. Then $(\mathbf{A} + \sigma_2\mathbf{I})\widetilde{\mathbf{U}} = \widehat{\mathbf{U}} + (\sigma_2 - \sigma_1)\widetilde{\mathbf{U}}$. Furthermore, let $\widehat{\mathbf{U}} = \mathbf{C}\mathbf{R}$ be the QR-factorization. Then

$$(\mathbf{A} + \sigma_2\mathbf{I})\widetilde{\mathbf{U}}\mathbf{R}^{-1} = \mathbf{C} + (\sigma_2 - \sigma_1)\widetilde{\mathbf{U}}\mathbf{R}^{-1}.$$

For clarity, we present a small example to motivate a general algorithm for this process. Suppose we have a family of three shifted systems $\mathbf{A} + \sigma_1\mathbf{I}$, $\mathbf{A} + \sigma_2\mathbf{I}$, and $\mathbf{A} + \sigma_3\mathbf{I}$. Our goal is, given $\widehat{\mathbf{U}} \in \mathbb{C}^{n\times k}$, to generate $\mathbf{U}^{(\sigma_1)}, \mathbf{U}^{(\sigma_2)}, \mathbf{U}^{(\sigma_3)} \in \mathbb{C}^{n\times k}$ such that for some $\mathbf{C} \in \mathbb{C}^{n\times k}$ such that $\mathbf{C}^*\mathbf{C} = \mathbf{I}_k$, we have

$$(\mathbf{A} + \sigma_1\mathbf{I})\mathbf{U}^{(\sigma_1)} = \mathbf{C} \quad \text{and} \quad (\mathbf{A} + \sigma_2\mathbf{I})\mathbf{U}^{(\sigma_2)} = \mathbf{C} \quad \text{and} \quad (\mathbf{A} + \sigma_3\mathbf{I})\mathbf{U}^{(\sigma_3)} = \mathbf{C}.$$

Furthermore, we want to do this using only $3k$ matrix-vector products (or three block k matrix-vector products) and k-dimensional basis orthogonalizations. This is a three-step process, and at each step, we will compute k matrix-vector products (or one block k matrix-vector product) and then apply Proposition 4.7 to implicitly compute other matrix-vector products needed in that step. To begin, let

$$\mathbf{U}_0^{(\sigma_1)} = \mathbf{U}_0^{(\sigma_2)} = \mathbf{U}_0^{(\sigma_3)} = \widehat{\mathbf{U}}.$$

We want to

1. Compute $\widehat{\mathbf{U}}_1^{(\sigma_1)} = (\mathbf{A} + \sigma_2\mathbf{I})\mathbf{U}_0^{(\sigma_1)}$, $\widehat{\mathbf{U}}_1^{(\sigma_2)} = (\mathbf{A} + \sigma_3\mathbf{I})\mathbf{U}_0^{(\sigma_2)}$, and $\widehat{\mathbf{U}}_1^{(\sigma_3)} = (\mathbf{A} + \sigma_1\mathbf{I})\mathbf{U}_0^{(\sigma_3)}$

2. Compute the QR-factorization $\widehat{\mathbf{U}}_1^{(\sigma_3)} = \mathbf{U}_1^{(\sigma_3)}\mathbf{R}_1$
3. Compute $\mathbf{U}_1^{(\sigma_1)} = \widehat{\mathbf{U}}_1^{(\sigma_1)}\mathbf{R}_1^{-1}$ and $\mathbf{U}_1^{(\sigma_2)} = \widehat{\mathbf{U}}_1^{(\sigma_2)}\mathbf{R}_1^{-1}$.

However, we do not need to do these computations explicitly. If we compute $\widehat{\mathbf{U}}_1^{(\sigma_3)} = (\mathbf{A} + \sigma_1\mathbf{I})\mathbf{U}_0^{(\sigma_2)}$ and the QR-factorization $\widehat{\mathbf{U}}_1^{(\sigma_3)} = \mathbf{U}_1^{(\sigma_3)}\mathbf{R}_1$, then by Proposition 4.7, we know that

$$\mathbf{U}_1^{(\sigma_1)} = \mathbf{U}_1^{(\sigma_3)} + (\sigma_2 - \sigma_1)\mathbf{U}_0^{(\sigma_1)}\mathbf{R}_1^{-1} \tag{4.45}$$

which then implies that, again by Proposition 4.7,

$$\mathbf{U}_1^{(\sigma_2)} = \mathbf{U}_1^{(\sigma_1)} + (\sigma_3 - \sigma_2)\mathbf{U}_0^{(\sigma_2)}\mathbf{R}_1^{-1}.$$

For the next step, explicitly, we want to

1. Compute $\widehat{\mathbf{U}}_2^{(\sigma_1)} = (\mathbf{A} + \sigma_3\mathbf{I})\mathbf{U}_1^{(\sigma_1)}$, $\widehat{\mathbf{U}}_2^{(\sigma_2)} = (\mathbf{A} + \sigma_1\mathbf{I})\mathbf{U}_1^{(\sigma_2)}$, and $\widehat{\mathbf{U}}_2^{(\sigma_3)} = (\mathbf{A} + \sigma_2\mathbf{I})\mathbf{U}_1^{(\sigma_2)}$
2. Compute the QR-factorization $\widehat{\mathbf{U}}_2^{(\sigma_3)} = \mathbf{U}_2^{(\sigma_3)}\mathbf{R}_2$
3. Compute $\mathbf{U}_2^{(\sigma_1)} = \widehat{\mathbf{U}}_2^{(\sigma_1)}\mathbf{R}_2^{-1}$ and $\mathbf{U}_2^{(\sigma_2)} = \widehat{\mathbf{U}}_2^{(\sigma_2)}\mathbf{R}_2^{-1}$.

Again, however, we do not need to do all these computations explicitly. If we compute $\widehat{\mathbf{U}}_2^{(\sigma_3)} = (\mathbf{A} + \sigma_2\mathbf{I})\mathbf{U}_1^{(\sigma_2)}$ and the QR-factorization $\widehat{\mathbf{U}}_2^{(\sigma_3)} = \mathbf{U}_2^{(\sigma_3)}\mathbf{R}_2$, then we have

$$\begin{aligned}
\widehat{\mathbf{U}}_2^{(\sigma_1)} &= (\mathbf{A} + \sigma_3\mathbf{I})\mathbf{U}_1^{(\sigma_1)} \qquad (4.46)\\
&= (\mathbf{A} + \sigma_3\mathbf{I})(\mathbf{U}_1^{(\sigma_3)} + (\sigma_2 - \sigma_1)\mathbf{U}_0^{(\sigma_1)}\mathbf{R}_1^{-1})\\
&= (\mathbf{A} + \sigma_2\mathbf{I})\mathbf{U}_1^{(\sigma_3)} + (\sigma_3 - \sigma_2)\mathbf{U}_1^{(\sigma_3)} + (\sigma_2 - \sigma_1)(\mathbf{A} + \sigma_3\mathbf{I})\mathbf{U}_0^{(\sigma_1)}\mathbf{R}_1^{-1}\\
&= \widehat{\mathbf{U}}_2^{(\sigma_3)} + (\sigma_3 - \sigma_2)\mathbf{U}_1^{(\sigma_3)} + (\sigma_2 - \sigma_1)(\mathbf{A} + \sigma_3\mathbf{I})\mathbf{U}_0^{(\sigma_1)}\mathbf{R}_1^{-1}.
\end{aligned}$$

By Proposition 4.7,

$$(\mathbf{A}+\sigma_3\mathbf{I})\mathbf{U}_0^{(\sigma_1)}\mathbf{R}_1^{-1} = \mathbf{U}_1^{(\sigma_3)} + (\sigma_3-\sigma_1)\mathbf{U}_0^{(\sigma_1)}\mathbf{R}_1^{-1},$$

yielding

$$\begin{aligned}
\widehat{\mathbf{U}}_2^{(\sigma_1)} &= \widehat{\mathbf{U}}_2^{(\sigma_3)} + (\sigma_3-\sigma_2)\mathbf{U}_1^{(\sigma_3)} + (\sigma_2-\sigma_1)(\mathbf{U}_1^{(\sigma_3)} + (\sigma_3-\sigma_1)\mathbf{U}_0^{(\sigma_1)}\mathbf{R}_1^{-1}) \\
&= \widehat{\mathbf{U}}_2^{(\sigma_3)} + (\sigma_3-\sigma_2)\mathbf{U}_1^{(\sigma_3)} + (\sigma_2-\sigma_1)\mathbf{U}_1^{(\sigma_3)} + (\sigma_2-\sigma_1)(\sigma_3-\sigma_1)\mathbf{U}_0^{(\sigma_1)}\mathbf{R}_1^{-1} \\
&= \widehat{\mathbf{U}}_2^{(\sigma_3)} + (\sigma_3-\sigma_2)\mathbf{U}_1^{(\sigma_3)} + (\sigma_2-\sigma_1)\mathbf{U}_1^{(\sigma_3)} + (\sigma_2-\sigma_1)(\sigma_3-\sigma_1)\mathbf{U}_0^{(\sigma_1)}\mathbf{R}_1^{-1} \\
&= \widehat{\mathbf{U}}_2^{(\sigma_3)} + (\sigma_3-\sigma_1)\mathbf{U}_1^{(\sigma_3)} + (\sigma_2-\sigma_1)(\sigma_3-\sigma_1)\mathbf{U}_0^{(\sigma_1)}\mathbf{R}_1^{-1} \\
&= \widehat{\mathbf{U}}_2^{(\sigma_3)} + (\sigma_3-\sigma_1)(\mathbf{U}_1^{(\sigma_3)} + (\sigma_2-\sigma_1)\mathbf{U}_0^{(\sigma_1)}\mathbf{R}_1^{-1}).
\end{aligned}$$

Finally, by (4.45), we have that

$$\widehat{\mathbf{U}}_2^{(\sigma_1)} = \widehat{\mathbf{U}}_2^{(\sigma_3)} + (\sigma_3-\sigma_1)\mathbf{U}_1^{(\sigma_1)},$$

which, when we post multiply by $\mathbf{R}_2^{-1}$, gives us

$$\mathbf{U}_2^{(\sigma_1)} = \mathbf{U}_2^{(\sigma_3)} + (\sigma_3-\sigma_1)\mathbf{U}_1^{(\sigma_1)}\mathbf{R}_2^{-1}.$$

A similar derivation shows us that

$$\mathbf{U}_2^{(\sigma_2)} = \mathbf{U}_2^{(\sigma_1)} + (\sigma_1-\sigma_2)\mathbf{U}_1^{(\sigma_2)}\mathbf{R}_2^{-1}.$$

From here, we can compute the QR-factorization $(\mathbf{A}+\sigma_3\mathbf{I})\mathbf{U}_2^{(\sigma_3)} = \mathbf{C}\mathbf{R}_3$ and let

$$\mathbf{U}^{(\sigma_1)} = \mathbf{U}_2^{(\sigma_1)}\mathbf{R}_3^{-1} \quad \text{and} \quad \mathbf{U}^{(\sigma_2)} = \mathbf{U}_2^{(\sigma_2)}\mathbf{R}_3^{-1} \quad \text{and} \quad \mathbf{U}^{(\sigma_3)} = \mathbf{U}_2^{(\sigma_3)}\mathbf{R}_3^{-1}.$$

For the general case, in which we have $s+1$ shifts, this suggests an, $(s+1)$-step algorithm for the computation $\mathbf{U}^{(\sigma_i)}$ for all i and $\mathbf{C}$, such that $\mathbf{C}^*\mathbf{C} = \mathbf{I}_k$.

We begin with some notation. For any shift σ_i, $\mathbf{U}^{(\sigma_i)}$ is such that $(\mathbf{A} + \sigma_i \mathbf{I})\mathbf{U}^{(\sigma_i)} = \mathbf{C}$. At the jth step, we compute k matrix-vector products and generate an intermediate matrix $\mathbf{U}_j^{(\sigma_i)}$ for each shift, and we denote the initial recycled space associated to σ_i, $\mathbf{U}_0^{(\sigma_i)} = \widetilde{\mathbf{U}}$. At step j, we compute the product

$$\widehat{\mathbf{U}}_j^{(\sigma_i)} = (\mathbf{A} + \sigma_{\rho_j}\mathbf{I})\mathbf{U}_{j-1}^{(\sigma_i)},$$

where $\rho_j = (j+1 \mod s+1) + \lfloor \frac{j+1}{s+1} \rfloor (s+1)$. This formula for ρ_j allows us to cyclically index through all the shifts using arithmetic on the group $\mathbb{Z}_{s+1} = \{1, 2, \ldots, s+1\}$. For stability, we need to maintain orthogonality of one of the bases, and it does not matter which one. Therefore, without loss of generality, we compute the QR-factorization $\widehat{\mathbf{U}}_j^{(\sigma_{s+1})} = \mathbf{U}_j^{(\sigma_{s+1})}\mathbf{R}_j$ and update $\mathbf{U}_j^{(\sigma_i)} = \widehat{\mathbf{U}}_j^{(\sigma_i)}\mathbf{R}_j^{-1}$ for each $i \neq s+1$. In each step of this process, we only compute k matrix-vector products explicitly (the one associated to σ_{s+1}). The others are computed using Proposition 4.7. At the end of the process, we have computed sk matrix-vector products (or s block k matrix-vector products). We finish by computing $\widehat{\mathbf{C}} = (\mathbf{A} + \sigma_{s+1}\mathbf{I})\mathbf{U}_s^{(\sigma_{s+1})}$ and one final QR-factorization $\widehat{\mathbf{C}} = \mathbf{C}\mathbf{R}$. We then update $\mathbf{U}^{(\sigma_i)} = \mathbf{U}_s^{(\sigma_i)}\mathbf{R}^{-1}$ for all i. We present these steps more precisely in Algorithm 4.4.1.

In Figure 4.10, we demonstrate the performance of Algorithm 4.4.1 for a set of shifts ranging from 10^{-8} to 10^{8}. We see that for shifts which are $\mathcal{O}(1)$ or smaller in magnitude, the algorithm performs quite well. However, for larger shifts, we see a dramatic loss of accuracy, where we define accuracy to mean the largest principal angle between $\mathcal{R}((\mathbf{A} + \sigma\mathbf{I})\mathbf{U}^{(\sigma)})$ and $\mathcal{R}(\mathbf{C})$. Further experiments show that it is not just the magnitude of the shifts that matter. Experimentally, we observe that three other criteria on the set of shifts also seem to effect the stability of the algorithm.

For a collection of real shifts, let $\{\sigma_i\}_{i=1}^{s} \subset [a,b]$ with $a < b$ where a and b are the smallest such numbers for which the interval contains the shifts. We have observed that $|b-a|$ effects stability. However, this effect is relative to the distance of the set of shifts from the origin, where we define distance as $|b+a|/2$, the distance of the midpoint from the origin. The further the distance of the set from the origin, the larger $|b-a|$ can be before instability onsets. This is demonstrated by experiments shown in Figure 4.11. What determines the performance of Algorithm 4.4.1 is how

Algorithm 4.4.1: Computing a Family of Recycled Spaces for Multiple Shifted Linear Systems

```
   Input  : $\mathbf{A} \in \mathbb{C}^{n\times n}$, $\{\sigma_i\}_{i=1}^{s+1} \subset \mathbb{C}$, $\widetilde{\mathbf{U}} \in \mathbb{C}^{n\times k}$
   Output: $\mathbf{C} \in \mathbb{C}^{n\times k}$ such that $\mathbf{C}^*\mathbf{C} = \mathbf{I}_k$, $\left\{\mathbf{U}^{(\sigma_i)}\right\}_{i=1}^{s+1}$ such that
           $(\mathbf{A} + \sigma_i\mathbf{I})\mathbf{U}^{(\sigma_i)} = \mathbf{C}$ for all $i$
1  for $i = 1$ to $s+1$ do
2      $\mathbf{U}^{(\sigma_i)} = \widetilde{\mathbf{U}}$
3  for $i = 1$ to $s$ do
4      $\mathbf{U}^{(\sigma_{s+1})} \leftarrow \mathbf{A}\mathbf{U}^{(\sigma_{s+1})} + \sigma_i\mathbf{U}^{(\sigma_{s+1})}$
5      Compute the QR-factorization $\mathbf{U}^{(\sigma_{s+1})} = \mathbf{QR}$
6      $\mathbf{U}^{(\sigma_{s+1})} \leftarrow \mathbf{Q}$
7      for $j = 1$ to $s$ do
8          $\widehat{\mathbf{U}} \leftarrow \mathbf{U}^{(\sigma_j)}\mathbf{R}^{-1}$
9          $k_1 \leftarrow (j-1 \mod s+1) + (1 - \lceil \frac{j-1}{s+1} \rceil)(s+1)$
10         $k_2 \leftarrow (j+1 \mod s+1) + \lfloor \frac{j+1}{s+1} \rfloor (s+1)$
11         $\mathbf{U}^{(\sigma_j)} \leftarrow \mathbf{U}^{(\sigma_{k_1})} + (\sigma_{k_2} - \sigma_j)\widehat{\mathbf{U}}$
12 $\mathbf{C} = \mathbf{A}\mathbf{U}^{(\sigma_{s+1})} + \sigma_{s+1}\mathbf{U}^{(\sigma_{s+1})}$
13 Compute the QR-factorization $\mathbf{C} = \mathbf{QR}$
14 $\mathbf{C} \leftarrow \mathbf{Q}$
15 for $i = 1$ to $s+1$ do
16     $\mathbf{U}^{(\sigma_i)} \leftarrow \mathbf{U}^{(\sigma_i)}\mathbf{R}^{-1}$
```

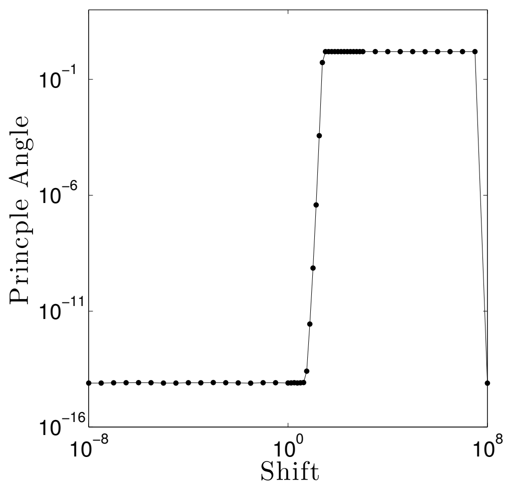


Figure 4.10. The performance of Algorithm 4.4.1 for a collection of shifts taken from the interval $[10^{-8}, 10^{8}]$. For this figure, both axes are on a logarithmic scale. We compute the largest principle angle between $\mathcal{R}(\mathbf{C})$ and $\mathcal{R}((\mathbf{A}+\sigma\mathbf{I})\mathbf{U}^{(\sigma)})$ to examine the accuracy.

closely clustered the shifts are, where "close" is defined relative to the distance of the midpoint from the origin.

Suppose the initial recycled space $\widetilde{\mathbf{U}}$ is an approximate invariant subspace of $\mathbf{A}$ associated to small-magnitude eigenvalues. The process described in Algorithm 4.4.1 might yield a matrix $\mathbf{C}$ with columns spanning a new approximate invariant subspace. This new subspace might not be associated to small magnitude eigenvalues, though. Because $\mathbf{C}$ is constructed implicitly by premultiplying $\widetilde{\mathbf{U}}$ by different members of the family of shifted matrices, it is not clear what eigenspace $\mathcal{R}(\mathbf{C})$ might approximate.

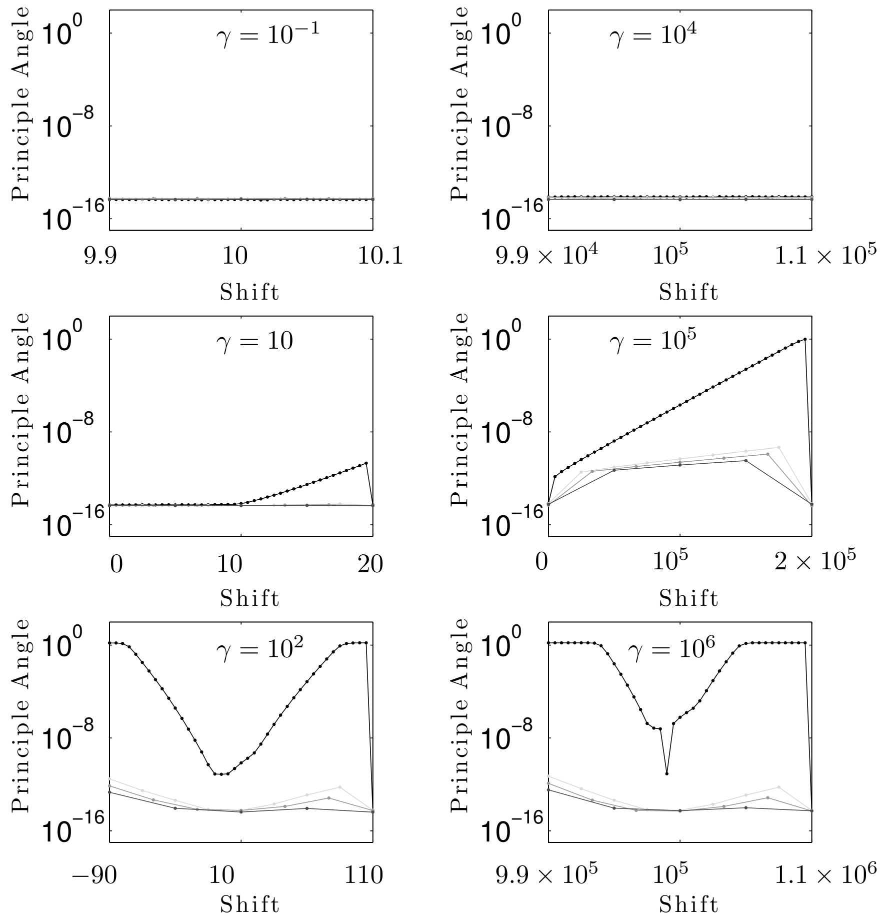


Figure 4.11. The performance of Algorithm 4.4.1 for collections of shifts. Different lines depict the algorithm's performance for different numbers of shifts. Line darkness increases when more shifts are used (meaning the algorithm requires more iterations). We compute the largest principle angle between $\mathcal{R}(\mathbf{C})$ and $\mathcal{R}((\mathbf{A} + \sigma\mathbf{I})\mathbf{U}^{(\sigma)})$ to examine the accuracy. In the figures in the left column, shifts come from the interval $[10 - \gamma, 10 + \gamma]$. In the right column, shifts come from the interval $[10^5 - \gamma, 10^5 + \gamma]$. Observe that the Algorithm 4.4.1 performs well for larger values of γ when the center is located at 10^5.

4.4.4 An Approximate Collinearity Scheme

The algorithm presented in Section 4.4.3 is the most natural extension of the recycling framework to GMRES for shifted systems, but in terms of storage, it is not efficient since storage requirements increase for each new shift. Furthermore, this process is computationally expensive. It costs sk matrix-vector products (or s block k matrix-vector products), and the orthogonalizations at each step are a source of additional computational cost. Therefore, to develop an algorithm with fixed-storage requirements regardless of the number of shifts, we propose to only store the matrix $\mathbf{U} \in \mathbb{C}^{n\times k}$ whose columns span the recycled space associated to the base system.

If we do not compute $\mathbf{U}^{(\sigma)}$ and substitute $\mathbf{U}$ in its place, the simple appearance of the shift in the Arnoldi relation no longer holds in the Arnoldi- like relation (4.9). Observe that for $\widehat{\mathbf{V}}_m$ and $\widehat{\mathbf{W}}_{m+1}$, defined as in (4.7),

$$(\mathbf{A} + \sigma\mathbf{I})\widehat{\mathbf{V}}_m = \widehat{\mathbf{W}}_{m+1} \begin{bmatrix} \mathbf{I}_k & \mathbf{B} \\ \mathbf{0} & \overline{\mathbf{H}}^{(\sigma)}_{m-k} \end{bmatrix} + \sigma\widehat{\mathbf{V}}_m.$$

If we have

$$\mathcal{R}(\widehat{\mathbf{V}}_m) \subset \mathcal{R}(\widehat{\mathbf{W}}_{m+1}), \tag{4.47}$$

the relation could be easily modified so that we could derive a method similar to GMRES for shifted systems which incorporates subspace recycling [33]. However, the inclusion (4.47) in general does not hold; $\mathbf{U}$ represents an approximate invariant subspace of $\mathbf{A}$, not a true invariant subspace. In general, the span of the columns of $\mathbf{U}$ will be different than that of $\mathbf{C}$. The same is true of $\widehat{\mathbf{V}}_m$ and $\widehat{\mathbf{W}}_{m+1}$, respectively. However, there is one scenario in which this does hold.

Consider the situation in which we begin with no starting recycled space and compute harmonic Ritz vectors at the end of each cycle to pass into the next cycle. We run an m-step cycle of shifted GMRES, and at the end of that cycle, we compute k harmonic Ritz vectors, compute $\mathbf{U}$ and $\mathbf{C}$ as before, and restart. Morgan [58] showed that for a harmonic Ritz pair $(\mathbf{g}, \theta^{(G)})$, the eigenvector residual $\mathbf{A}\mathbf{g} - \theta^{(G)}\mathbf{g}$ is a multiple of the GMRES residual $\mathbf{r}_m$. At the end of a cycle, if we compute k harmonic Ritz vectors and store them as the columns of $\widetilde{\mathbf{U}}$, then we know that $\mathcal{R}(\mathbf{A}\widetilde{\mathbf{U}} - \widetilde{\mathbf{U}}\mathbf{D}) = \operatorname{span}(\mathbf{r}_m)$ where $\mathbf{D} = \operatorname{diag}(\theta_1^{(G)}, \ldots, \theta_k^{(G)})$, the diagonal matrix containing the respective harmonic Ritz values associated to the columns of $\widetilde{\mathbf{U}}$. If we compute the QR-factorization $\mathbf{A}\widetilde{\mathbf{U}} = \mathbf{C}\mathbf{R}$ and let $\mathbf{U} = \widetilde{\mathbf{U}}\mathbf{R}^{-1}$, then for $\mathbf{T}_1 = \mathbf{R}^{-1}$ and $\mathbf{T}_2 = \mathbf{D}\mathbf{R}^{-1}$, we have $\mathcal{R}(\mathbf{C}\mathbf{T}_1 - \mathbf{U}\mathbf{T}_2) = \operatorname{span}(\mathbf{r}_m)$. At the beginning of the next cycle, we take $\mathbf{v}_1 = \mathbf{r}_m / \|\mathbf{r}_m\|$ as the first Krylov vector; and in this case, the containment (4.47) holds. This is the same fact exploited in [19].

In general, we do not have such luck. What is the best we can do? Before we proceed with the mathematical details, it will be helpful to provide an overview of the strategy we are proposing and encode this into a schematic algorithm. This algorithm will allow us to get a good approximation at low cost. As before, the base system will be solved using cycles of GCRODR. For the shifted system, though we can initially orthogonally project the residual away from $\mathcal{R}(\mathbf{C})$, we cannot perform the corresponding update of the solution. Therefore, we compute this step approximately. We will show that for the shifted system we cannot enforce a residual collinearity condition, as in (4.37), at the end of each cycle. However, neglecting a term from the collinearity equation allows us to solve a nearby approximate collinearity condition directly. We use the solution from this nearby equation to update the approximation for the shifted system. Experimentally, we have observed that these corrections to the shifted system solution improve the residual but do not lead to convergence for the shifted system. Therefore, we only monitor the the residual norm of the base system and terminate when the base system has converged to tolerance. We then correct the approximation for the shifted system using GCRODR. We present a schematic of this process in Algorithm 4.4.2.

Algorithm 4.4.2: Schematic of Shifted GCRODR with an Approximate Collinearity Condition

```
Input : $\mathbf{A} \in \mathbb{C}^{n\times n}$; $\sigma \in \mathbb{C}$; $\mathbf{U}, \mathbf{C} \in \mathbb{C}^{n\times k}$ such that $\mathbf{AU} = \mathbf{C}$ and
        $\mathbf{C}^*\mathbf{C} = \mathbf{I}_k$; Initial Approximations $\mathbf{x}_0$ and $\mathbf{x}_0^{(\sigma)}$ such that
        residuals are collinear; $\varepsilon > 0$
1  $\mathbf{x} \leftarrow \mathbf{x}_0$, $\mathbf{r} = \mathbf{b} - \mathbf{Ax}$
2  $\mathbf{x} \leftarrow \mathbf{x} + \mathbf{UC}^*\mathbf{r}$, $\mathbf{r} \leftarrow \mathbf{r} - \mathbf{CC}^*\mathbf{r}$;            Project base residual
3  $\mathbf{x}^{(\sigma)} \leftarrow \mathbf{x}_0^{(\sigma)}$, $r^{(\sigma)} = \mathbf{b} - \mathbf{Ax}^{(\sigma)}$
4  $\mathbf{x}^{(\sigma)} \leftarrow \mathbf{x}^{(\sigma)} + \mathbf{UC}^*\mathbf{r}^{(\sigma)}$;      Approximately update shifted solution
5  while $\|\mathbf{r}\| > \varepsilon$ do
6      Construct subspace $\mathcal{K}_m((\mathbf{I} - \mathbf{CC}^*)\mathbf{A}, \mathbf{r})$
7      Compute update $\mathbf{t} \in \mathcal{R}(\mathbf{U}) + \mathcal{K}_m((\mathbf{I} - \mathbf{CC}^*)\mathbf{A}, \mathbf{r})$ using GCRODR
       minimization
8      $\mathbf{x} \leftarrow \mathbf{x} + \mathbf{t}$; $\mathbf{r} \leftarrow \mathbf{b} - \mathbf{Ax}$
9      Compute update $\mathbf{t}^{(\sigma)} \in \mathcal{R}(\mathbf{U}) + \mathcal{K}_m((\mathbf{I} - \mathbf{CC}^*)\mathbf{A}, \mathbf{r})$ according to the
       approximate collinearity condition
10     $\mathbf{x}^{(\sigma)} \leftarrow \mathbf{x}^{(\sigma)} + \mathbf{t}^{(\sigma)}$
11     Compute updated recycled subspace information $\mathbf{U}$ and $\mathbf{C}$
12 Apply GCRODR to $\mathbf{x}^{(\sigma)}$ yielding shifted system approximation that
   achieves tolerance $\varepsilon$
```

We now present a more detailed description of Algorithm 4.4.2. Let $\mathbf{E}$ be a matrix whose columns form a basis for the orthogonal completion of $\mathcal{R}(\mathbf{C}) \oplus \mathcal{R}(\mathbf{V}_{m-k+1})$

in $\mathbb{C}^n$. We note that $\mathbf{E}$ needs not be computed; we use it here as a theoretical tool to develop our algorithm. Then we can write

$$\mathbf{U} = \mathbf{C}\mathbf{Y} + \mathbf{V}_{m-k+1}\mathbf{Z} + \mathbf{E}\mathbf{F}, \tag{4.48}$$

where $\mathbf{Y} \in \mathbb{C}^{k\times k}$, $\mathbf{Z} \in \mathbb{C}^{(m-k+1)\times k}$, and $\mathbf{F} \in \mathbb{C}^{(n-m+1)\times k}$. This yields the following *imperfect* Arnoldi-like relation for the shifted system,

$$(\mathbf{A} + \sigma\mathbf{I})\begin{bmatrix}\mathbf{U} & \mathbf{V}_{m-k}\end{bmatrix} = \begin{bmatrix}\mathbf{C} & \mathbf{V}_{m-k+1}\end{bmatrix}\begin{bmatrix}\mathbf{I}_k + \sigma\mathbf{Y} & \mathbf{B} \\ \sigma\mathbf{Z} & \overline{\mathbf{H}}_{m-k}^{(\sigma)}\end{bmatrix} + \sigma\begin{bmatrix}\mathbf{E}\mathbf{F} & \mathbf{0}\end{bmatrix}. \tag{4.49}$$

If we let

$$\overline{\mathbf{G}}_m^{(\sigma)} = \begin{bmatrix}\mathbf{I}_k + \sigma\mathbf{Y} & \mathbf{B} \\ \sigma\mathbf{Z} & \overline{\mathbf{H}}_{m-k}^{(\sigma)}\end{bmatrix}, \tag{4.50}$$

together with (4.9), then the Arnoldi-like relation (4.49) can be rewritten as

$$(\mathbf{A} + \sigma\mathbf{I})\widehat{\mathbf{V}}_m = \widehat{\mathbf{W}}_{m+1}\overline{\mathbf{G}}_m^{(\sigma)} + \sigma\begin{bmatrix}\mathbf{E}\mathbf{F} & \mathbf{0}\end{bmatrix}.$$

At the end of the cycle, we minimize the residual of the base system accordingly using GCRODR, obtaining $\mathbf{y}_m$ as in (4.10), so that

$$\begin{aligned}\mathbf{r}_m &= \mathbf{r}_0 - \widehat{\mathbf{W}}_{m+1}\overline{\mathbf{G}}_m\mathbf{y}_m \\ &= \|\mathbf{r}\|\,\widehat{\mathbf{W}}_{m+1}\mathbf{e}_{k+1} - \widehat{\mathbf{W}}_{m+1}\overline{\mathbf{G}}_m\mathbf{y}_m \\ &= \widehat{\mathbf{W}}_{m+1}\mathbf{z}_{m+1},\end{aligned}$$

where

$$\mathbf{z}_{m+1} = \|\mathbf{r}_0\|\,\mathbf{e}_{k+1} - \overline{\mathbf{G}}_m\mathbf{y}_m. \tag{4.51}$$

Now, for the shifted system, we would like to enforce the collinearity condition. If a collinear residual were to exist for the shifted system, then it would satisfy

$$\begin{aligned} \mathbf{r}^{(\sigma)} &= \beta_m \mathbf{r} \qquad (4.52) \\ \mathbf{b} - (\mathbf{A} + \sigma\mathbf{I})(\mathbf{x}^{(\sigma)} + \widehat{\mathbf{V}}_m \mathbf{y}_m^{(\sigma)}) &= \beta_m \widehat{\mathbf{W}}_{m+1}\mathbf{z}_{m+1} \\ \mathbf{r}^{(\sigma)} - (\mathbf{A} + \sigma\mathbf{I})\widehat{\mathbf{V}}_m \mathbf{y}_m^{(\sigma)} &= \widehat{\mathbf{W}}_{m+1}\mathbf{z}_{m+1}\beta_m \\ \beta_0\mathbf{r} - (\widehat{\mathbf{W}}_{m+1}\overline{\mathbf{G}}_m^{(\sigma)} + \sigma\begin{bmatrix}\mathbf{EF} & \mathbf{0}\end{bmatrix})\mathbf{y}_m^{(\sigma)} &= \widehat{\mathbf{W}}_{m+1}\mathbf{z}_{m+1}\beta_m \\ \beta_0\mathbf{r} &= \widehat{\mathbf{W}}_{m+1}(\mathbf{z}_{m+1}\beta_m + \overline{\mathbf{G}}_m^{(\sigma)}\mathbf{y}_m^{(\sigma)}) \\ &\quad +\sigma\begin{bmatrix}\mathbf{EF} & \mathbf{0}\end{bmatrix}\mathbf{y}_m^{(\sigma)}. \end{aligned}$$

However, such a solution cannot exist. Observe that $\mathbf{r} \in \mathcal{R}(\mathbf{C}) \oplus \mathcal{R}(\mathbf{V}_{m-k+1})$ while the right-hand side of (4.52) has a non-zero component in $(\mathcal{R}(\mathbf{C}) \oplus \mathcal{R}(\mathbf{V}_{m-k+1}))^{\perp}$. We can instead solve a nearby problem. As we previously stated, $\mathbf{EF}$ is generally non-zero. However, if it is relatively small (e.g., in the sense of the 2-norm) perhaps we can disregard it. This yields an augmented linear system which can be solved directly,

$$\begin{aligned} \mathbf{z}_{m+1}\tilde{\beta}_m + \overline{\mathbf{G}}_m^{(\sigma)}\tilde{\mathbf{y}}_m^{(\sigma)} &= \beta_0\left\|\mathbf{r}\right\|\mathbf{e}_{k+1}, \quad \text{or} \qquad (4.53) \\ \begin{bmatrix}\overline{\mathbf{G}}_m^{(\sigma)} & \mathbf{z}_{m+1}\end{bmatrix}\begin{bmatrix}\tilde{\mathbf{y}}_m^{(\sigma)} \\ \tilde{\beta}_m\end{bmatrix} &= \beta_0\left\|\mathbf{r}\right\|\mathbf{e}_{k+1}. \end{aligned}$$

Thus, we proceed by solving this nearby problem and updating the shifted solution,

$$\mathbf{x}_m^{(\sigma)} = \mathbf{x}_0^{(\sigma)} + \widehat{\mathbf{V}}_m\tilde{\mathbf{y}}_m^{(\sigma)}. \qquad (4.54)$$

For each restart cycle, we repeat this process for the shifted system. We measure convergence criteria only for the base system, and stop when the base residual norm is below tolerance. As experiments show in Section 4.4.5, the update (4.54) at each cycle improves the solution for the shifted system at very little additional cost. When the residual norm for the base system reaches tolerance, the residual norm of the shifted system will have been reduced but not to tolerance. We then apply GCRODR algorithm [68] to the shifted system until the shifted residual norm is reduced to tolerance. Observe that for any number of shifts, we can easily form $\overline{\mathbf{G}}_m^{(\sigma)}$ for each i at little additional cost. The matrices $\mathbf{Y}$ and $\mathbf{Z}$ in (4.48) must be computed only once, regardless of the number of shifted systems we are solving. However, the number of correction cycles we must execute at the end to solve the shifted systems will increase with each additional shift.

We make two observations about this new algorithm. Once the approximation for the base system has converged to tolerance, we can make one of the shifted systems into the new base system. We can then reapply the algorithm with this new base system, allowing for possible improvement to the remaining shifted systems. This strategy can be repeated until there is only one system left unsolved. Then we can apply GCRODR to that system.

Unlike the shifted GMRES algorithm, our shifted GCRODR algorithm will always produce approximations which have converged below tolerance. This is because we only monitor the residual norm of the base system approximation. When that has converged, we correct the shifted systems. Therefore, even if (4.53) is highly ill-conditioned for some shift σ, producing unhelpful approximations, the algorithm will still correct the approximation associated to σ at the end of the process. Note that the we have not developed a theory of when the approximate collinearity condition (4.53) might fail. However, in our experiments, we were able to apply the approximate collinearity condition in situations where the shifted GMRES method was unable to produce solutions.

Analysis of the Approximate Collinearity Condition. We provide a simple analysis which justifies why the approximate collinearity condition will produce improved approximations to the solutions of the shifted systems and to understand how well we can expect the algorithm to perform.

Theorem 4.8: Suppose we begin the cycle as in (4.52), with approximate collinearity between the base and shifted residuals, satisfying the relationship

$$\mathbf{r}_0^{(\sigma)} = \tilde{\beta}_0 \mathbf{r}_0 + \mathbf{s}. \tag{4.55}$$

Then if we run a cycle of GCRODR to reduce the residual of the base system and apply the approximate collinearity condition (4.53), we have the relationship

$$\tilde{\mathbf{r}}_m^{(\sigma)} = \tilde{\beta}_m \mathbf{r}_m + \sigma \mathbf{E}\mathbf{F} \left(\tilde{\mathbf{y}}_m^{(\sigma)} \right)_{1:k} + s \tag{4.56}$$

Proof. We can write the residual produced by the approximate collinearity procedure for the shifted system,

$$\begin{aligned}
\tilde{\mathbf{r}}_m^{(\sigma)} &= \mathbf{r}_0^{(\sigma)} - (\mathbf{A} + \sigma\mathbf{I})\mathbf{x}_m^{(\sigma)} && (4.57)\\
&= \tilde{\beta}_0\mathbf{r}_0 + \mathbf{s} - (\mathbf{A} + \sigma\mathbf{I})\widehat{\mathbf{V}}_m\tilde{\mathbf{y}}_m^{(\sigma)} && (4.58)\\
&= \tilde{\beta}_0\mathbf{r}_0 - \left(\widehat{\mathbf{W}}_{m+1}\overline{\mathbf{G}}_m^{(\sigma)} + \sigma\begin{bmatrix}\mathbf{EF} & \mathbf{0}\end{bmatrix}\right)\tilde{\mathbf{y}}_m^{(\sigma)} + \mathbf{s} && (4.59)\\
&= \tilde{\beta}_0\left\|\mathbf{r}_0\right\|\widehat{\mathbf{W}}_{m+1}\mathbf{e}_{k+1} - \widehat{\mathbf{W}}_{m+1}\overline{\mathbf{G}}_m\tilde{\mathbf{y}}_m^{(\sigma)} + \sigma\begin{bmatrix}\mathbf{EF} & \mathbf{0}\end{bmatrix}\tilde{\mathbf{y}}_m^{(\sigma)} + \mathbf{s}\\
&= \tilde{\beta}_0\left\|\mathbf{r}_0\right\|\widehat{\mathbf{W}}_{m+1}\mathbf{e}_{k+1} - \widehat{\mathbf{W}}_{m+1}\overline{\mathbf{G}}_m\tilde{\mathbf{y}}_m^{(\sigma)} - \tilde{\beta}_m\widehat{\mathbf{W}}_{m+1}\mathbf{z}_{m+1} + \tilde{\beta}_m\widehat{\mathbf{W}}_{m+1}\mathbf{z}_{m+1}\\
&\quad +\sigma\begin{bmatrix}\mathbf{EF} & \mathbf{0}\end{bmatrix}\tilde{\mathbf{y}}_m^{(\sigma)} + \mathbf{s}\\
&= \widehat{\mathbf{W}}_{m+1}\left(\tilde{\beta}_0\left\|\mathbf{r}_0\right\|\mathbf{e}_{k+1} - \overline{\mathbf{G}}_m\tilde{\mathbf{y}}_m^{(\sigma)} - \tilde{\beta}_m\mathbf{z}_{m+1}\right) + \tilde{\beta}_m\widehat{\mathbf{W}}_{m+1}\mathbf{z}_{m+1}\\
&\quad +\sigma\begin{bmatrix}\mathbf{EF} & \mathbf{0}\end{bmatrix}\tilde{\mathbf{y}}_m^{(\sigma)} + \mathbf{s}.
\end{aligned}$$

Now using the approximate collinearity condition (4.53) and the fact that by definition $\mathbf{r}_m = \widehat{\mathbf{W}}_{m+1}\mathbf{z}_{m+1}$, we have that

$$\tilde{\mathbf{r}}_m^{(\sigma)} = \tilde{\beta}_m\mathbf{r}_m + \sigma\begin{bmatrix}\mathbf{EF} & \mathbf{0}\end{bmatrix}\tilde{\mathbf{y}}_m^{(\sigma)} + \mathbf{s}.$$

Then we can write

$$\tilde{\mathbf{r}}_m^{(\sigma)} = \tilde{\beta}_m\mathbf{r}_m + \sigma\mathbf{EF}\left(\tilde{\mathbf{y}}_m^{(\sigma)}\right)_{1:k} + \mathbf{s}. \qquad \square$$

Through the use of simple inequalities, this leads to an important corollary describing the amount of residual norm reduction we can expect for the shifted systems.

Corollary 4.9: Under the same assumptions as in Theorem 4.8, if we apply the approximate collinearity condition (4.53) to obtain a correction $\mathbf{x}_m^{(\sigma)}$ for the shifted system, then the associated residual satisfies the following inequality,

$$\left\|\tilde{\mathbf{r}}_m^{(\sigma)}\right\| \leq \left|\tilde{\beta}_m\right| \left\|\mathbf{r}_m\right\| + |\sigma| \left\|\mathbf{EF}\right\| \left\|\left(\tilde{\mathbf{y}}_m^{(\sigma)}\right)_{1:k}\right\| + \|s\| .$$

This gives us important information about performance of this method. We will see decrease in $\left\|\tilde{\mathbf{r}}_m^{(\sigma)}\right\|$ as long as $\left|\tilde{\beta}_m\right| \|\mathbf{r}_m\|$ is larger than $|\sigma| \|\mathbf{EF}\| \left\|\left(\tilde{\mathbf{y}}_m^{(\sigma)}\right)_{1:k}\right\| + \|s\|$. Furthermore, we see that the amount of improvement we can expect from computing corrections for the shifted system according to (4.53) is affected by $|\sigma|$, $\|\mathbf{EF}\|$, and $\left\|\left(\tilde{\mathbf{y}}_m^{(\sigma)}\right)_{1:k}\right\|$. We cannot control $\left\|\left(\tilde{\mathbf{y}}_m^{(\sigma)}\right)_{1:k}\right\|$, and σ is dictated by the problem. However, the size of $\|\mathbf{EF}\|$ is connected to the quality of $\mathbf{U}$ as an approximation to an invariant subspace of $\mathbf{A}$. This can seen by writing

$$\mathbf{EF} = \mathbf{U} - \left(\mathbf{CY} + \mathbf{V}_{m-k+1}\mathbf{Z}\right)$$

and observing that the norm of this difference decreases as $\mathbf{U}$ becomes a better approximation of an invariant subspace. This can also be formulated in terms of a quantity called the *containment gap* [14], which quantifies how close a smaller subspace is to being contained in a larger subspace. It is a generalization of the notion of the *gap* between two subspaces of equal dimension [89].

We should note that, unlike the shifted GMRES method, the approximate collinearity condition may be satisfied even if the bases system is not real-positive or one of the shifts is negative. Therefore, there are situations in which our method can produce approximations for which the shifted GMRES method cannot. This will be demonstrated with numerical experiments. We do not know, however, when the approximate collinearity condition might fail. If it does, though, we can simply not compute the approximation at that step and continue to the next iteration.

4.4.5 Experiments

For these tests, we constructed recycled subspaces from harmonic Ritz vectors of the coefficient matrix associated with the base system (1.1) with respect to the augmented subspace. This is by no means the only way to generate a recycled space. A strength of the GCRODR method is the ability to use any recycled subspace. We based our choice on the report in [68] that the use of harmonic Ritz vectors in the recycled space has given good results. In fact, we used the harmonic Ritz vector generation procedures provided as part of the GCRODR codes available on Michael Parks' website [67].

Our first experiment, in Figure 4.12, demonstrates the performance of the GCRODR method for shifted linear systems on a sequence of four bidiagonal matrices. The first matrix, $\mathbf{B}_1$, used in [19], is a bidiagonal matrix with the numbers $\{.1, 1, 2, \ldots, 998, 999\}$ on the diagonal and ones on the first superdiagonal and the other matrices are random bidiagonal perturbation so of $\mathbf{B}_1$. We see that, as predicted by Corollary 4.8, the amount of residual reduction achieved for the shifted systems is effected by the size of the shift. For the shift $\sigma_1 = 10^{-2}$, the relative residual is reduced to $\mathcal{O}(10^{-4})$ during the solving of the base system while for $\sigma_4 = 10$, the relative residual is only reduced to $\mathcal{O}(10^{-1})$. This experiment is more for illustrative purposes than to demonstrate superior performance. Observe that based on the performance for System 1, it may make more sense to apply the method repeatedly with no intra-system recycling, i.e., apply GMRES-DR to each system independently.

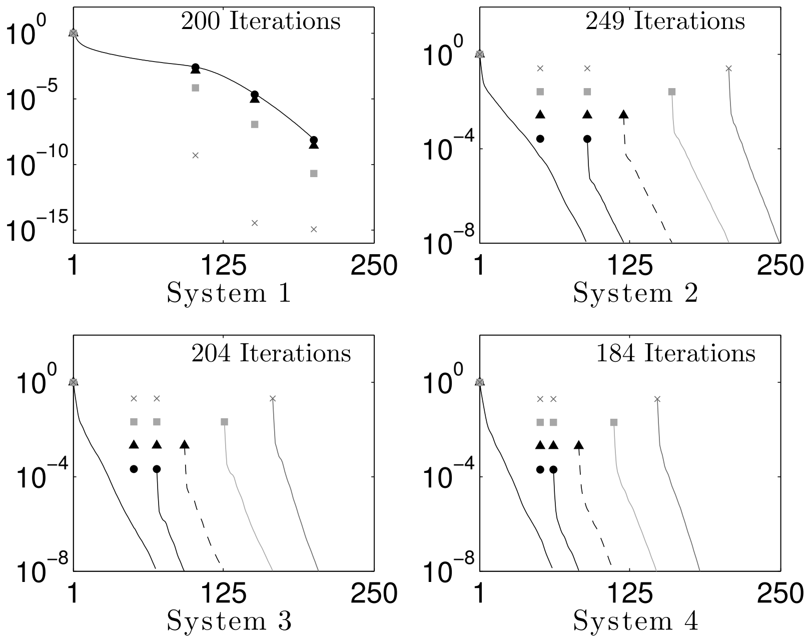


Figure 4.12. The performance of GCRODR for shifted systems. We performed simple tests on a sequence of four 1000×1000 bidiagonal matrices. The first matrix is the bidiagonal matrix used in [19]. The other systems are constructed by applying $\mathcal{O}(1)$ bidiagonal random perturbations to the first system using the `sprand()` Matlab function. The four shifts are 10^{-2}, 10^{-1}, 1, and 10. For the shifted systems, the residuals are only computed at the end of each cycle, with residual norms represented by the **circle**, **triangle**, **square**, and **cross**, respectively. The curves originating from these symbols are the convergence curves for the GCRODR iterations executed for each shift system in serial at the end of the process.

In this experiment, we ran the algorithm until we achieved convergence for the base system. The shifted systems were each solved with GCRODR. After convergence for the base system, we can also take one of the shifted systems as our new base system and reapply the algorithm for the smaller family of systems. In Figure 4.4.5, we see that this strategy offers a small improvement in iteration counts, a low-cost boost in performance. For comparison, we present in Figure 4.4.5

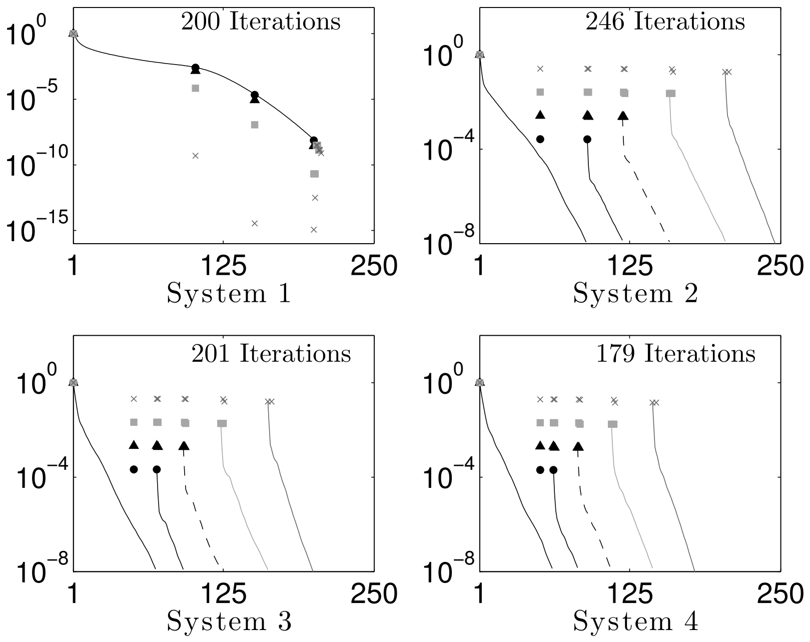


Figure 4.13. An experiment for the same sequence of systems and shifts as in Figure 4.12. Here, after the base system converges, we choose a new base system and apply the algorithm recursively on a smaller family of systems. We observe that this strategy does yield an 11 iteration improvement.

the convergence curves if we simply apply GCRODR to each shifted system in serial. For our second experiment, we take the sequence of seven 3072×3072 sample matrices (called $\mathbf{D}_1$ through $\mathbf{D}_7$) from the QCD collection from the Matrix Market website [1] with filename prefix `conf5.0-00l4x4`. We can construct the coefficient matrix $\mathbf{A}_i = \mathbf{I} - \kappa^{(i)}\mathbf{D}_i$ where $\kappa^{(i)}$ is a parameter associated to the QCD problem. For each matrix, there exists some critical value $\kappa_c^{(i)}$ such that for

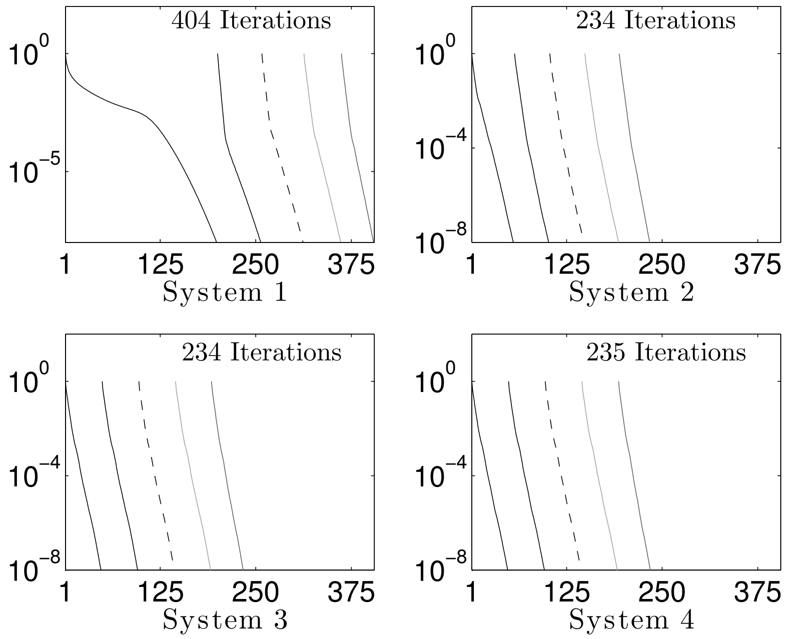


Figure 4.14. Convergence curves when we apply GCRODR to each shifted bidiagonal system in serial.

$0 \leq \kappa^{(i)} < \kappa_c^{(i)}$, $\mathbf{A}_i$ is a real-positive matrix. Equivalently, for each $\mathbf{A}_i$, we can write $\mathbf{A}_i = \frac{1}{\kappa^{(i)}}\mathbf{I} - \mathbf{D}_i$ where $\frac{1}{\kappa_c^{(i)}} < \frac{1}{\kappa^{(i)}} < \infty$, and we can scale any right-hand-side so that we are solving the same problem. For each $\mathbf{D}_i$, $\kappa_c^{(i)}$ is listed on the Matrix Market website, and in these experiments, all are in the interval $[0.20, 0.22]$. Frequently in QCD computations, we wish to solve with multiple parameters, and we can solve all these systems simultaneously using an algorithm for shifted linear systems.

We chose $\{.001, .002, .003, -.6, -.5\}$ as our family of shifts. Observe that by the definition of $\mathbf{A}_i$ and $\kappa_c^{(i)}$, the two shifted coefficient matrices associated to the two negative shifts are not real-positive. In fact, the method of Frommer and Gläsner [33]

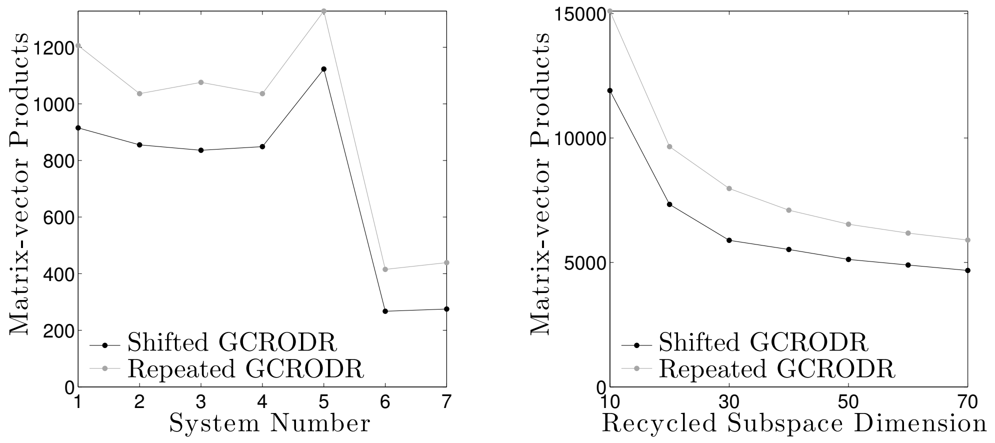


Figure 4.15. The performance of GCRODR for shifted systems on a sequence of seven small Wilson fermion matrices where for each matrix, we solve a linear system with the base system and ones associated to the shifts, .001 , .002, .003, $-.6$, and $-.5$. In the left figure, we demonstrate the performance of GCRODR(50,50) for shifted systems as compared to repeated applications of GCRODR(50,50). In the figure on the right, we compare performance for different size recycled subspaces.

is not guaranteed to produce solutions for the shifted systems satisfying the collinearity condition, due to the negativity of these shifts. In this experiment, the method frequently failed to produce shifted systems for long sequences of iterations. Since the shifted GMRES method did not converge for some systems, its performance was not included in the figure. It is realistic to include such shifts, as it was observed, e.g., in [32], that such shifts may arise due to statistical aberrations in generating these matrices using a Monte Carlo method. This highlights that fact that the satisfaction of the approximate collinearity condition is not reliant to the real-positivity of the base system or positivity of the shifts. We simply apply GCRODR to the base system, and attempt to cheaply improve the approximate solutions to the shifted systems. We compared with another strategy, repeated applications of GCRODR

[68] for the base and shifted system. In Figure 4.15 we present the matrix-vector product counts for each system for a particular recycled subspace dimension as well as the totals over seven systems for various recycled subspace dimensions. We see that our method is able to produce a 20% reduction is the number of matrix-vector products needed to solve these systems, when compared to repeated applications of GCRODR. Again, we note that the shifted GMRES method was unable to produce approximation for some of the shifted systems being solved in this experiment. In Figure 4.16, we present the convergence curves for the first six QCD matrices.

4.4.6 Conclusion

We have presented two extensions of the GCRODR algorithm which can be used to solve a family of linear systems for which the coefficient matrices differ by multiples of the identity. The first algorithm is a direct extension of the GMRES method for shifted systems of Frommer and Gläsaner [33], but this method is impractical and becomes increasingly expensive (computationally and in terms of storage) as the number of shifts grows. Therefore, we introduced a different method which applies an approximation of the residual collinearity condition used in the development of GMRES for shifted systems [33]. This approximate collinearity condition allows us to improve the approximation for the shifted systems while solving the base system using the GCRODR algorithm. These improved approximations come at little additional computational cost. At the end, we apply cycles of GCRODR to improve the shifted system approximations to tolerance. Thus, our method does increase in computational cost as more shifts are introduced. The amount of improvement we can achieve is governed by the size of the shifts and the quality of the recycled

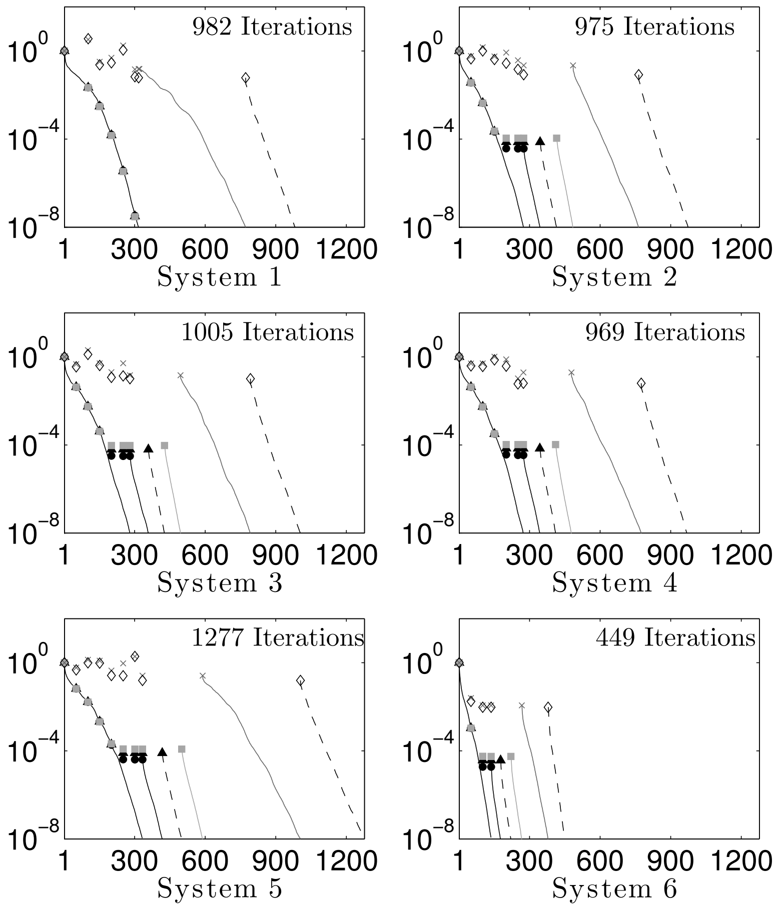


Figure 4.16. Convergence curves from the same experiment as in Figure 4.15. Observe that two of the shifted systems (corresponding to the negative shifts) require more work than the others for each base system **including** the first system, in which we started with no recycled subspace. These are the two shifts for which shifted GMRES (or shifted GMRES-DR) would fail. Notice that in this case, we are still able to converge by applying GCRODR at the end.

subspace as an invariant subspace of the coefficient matrix. We demonstrated that this method can solve problems for which GMRES for shifted linear systems fails to produce solutions. These experiments show our algorithm to be superior for solving linear systems when compared to the strategy of applying GCRODR to each shifted system in sequence.

CHAPTER 5

CONCLUSIONS

We have shown that producing storage-efficient Krylov subspace methods which are effective is not always an easy task. For nearly-Hermitian linear systems, the progressive GMRES algorithm is a clever algorithm, exploiting the structure of the matrix to limit storage of the GMRES algorithm. Unfortunately, in exploiting this structure, the algorithm also becomes unstable. We were able to document the causes of the instability and demonstrate them through well-constructed numerical experiments. As an alternative, we presented the Schur complement method. This method has similar storage characteristics as progressive GMRES but does not suffer from the same instabilities. We demonstrated its effectiveness using numerical experiments. The algorithm is also parallelizable, and we show that the method is competitive for large problems when compared to other state-of-the-art fixed-storage Krylov subspace methods.

For general non-Hermitian linear systems, subspace recycling has been shown to be an effective subspace augmentation strategy, allowing for the selection of optimal corrections over arbitrary augmented Krylov subspaces. We have extended this method to the block Krylov subspace setting. We were able to compare performance of this algorithm with that of GCRODR using existing convergence theory comparing GMRES with block GMRES. We then tested the algorithm on sequence of seven Jacobian matrices arising from a Newton iteration associated to fluid density functional theory. We have codes in both Matlab and Sandia's Trilinos C++ library, and we are continuing work on the Trilinos codes in order to perform large scale experiments and scaling studies.

We also extended the method of subspace recycling to simultaneously solve a family of shifted linear systems. We found that the equivalence of the Krylov subspace under a shift of the coefficient matrix does not hold for augmented Krylov subspaces. Thus we could not directly extend methods such as the shifted GMRES method to use subspace recycling. Instead, we enforced an approximation of the residual collinearity condition using the in the shifted GMRES method to obtain corrections for the shifted system approximations. We proved that such a procedure will reduce the norm of the residuals for the shifted systems, though usually not to tolerance. Since the base system is being solved using GCRODR, that approximation will converge to tolerance, and we can then take one of the shifted system as the new base system to apply the method recursively on this smaller family of systems. We demonstrated the effectiveness of the method on a sequence of bidiagonal families of coefficient matrices as well as a sequence families where each base coefficient matrix is a Wilson fermion matrix from a QCD problem. This method also has the benefit of being guaranteed to produce approximations for any coefficient matrix and any set of shifts, whereas the shifted GMRES method does not.

BIBLIOGRAPHY

[1] Matrix Market website, 2011. `http://math.nist.gov/MatrixMarket/`.

[2] The Trilinos Project website., Accessed February 24, 2012 . `http://trilinos.sandia.gov/packages/`.

[3] Description of the poly1-cms-1d problems, Accessed February 24, 2012. `https://software.sandia.gov/tramonto/src_docs/POLY1__CMS__1D_2README.html`.

[4] Kapil Ahuja, Michael L. Parks, Eric T. Phipps, Andy G. Salinger, and Eric de Sturler. Krylov recycling for climate modeling and uncertainty quantification. Technical Report SAND2010-8783P, Sandia National Laboratories Computer Science Research Institute, 2010.

[5] Walter E. Arnoldi. The principle of minimized iteration in the solution of the matrix eigenvalue problem. *Quarterly of Applied Mathematics*, 9:17–29, 1951.

[6] James Baglama, Daniela Calvetti, Gene H. Golub, and Lothar Reichel. Adaptively preconditioned GMRES algorithms. *SIAM Journal on Scientific Computing*, 20(1):243–269, 1998.

[7] Allison H. Baker, Elizabeth R. Jessup, and Thomas Manteuffel. A technique for accelerating the convergence of restarted GMRES. *SIAM Journal on Matrix Analysis and Applications*, 26(4):962–984, 2005.

[8] Bernard Beckermann and Lothar Reichel. The Arnoldi process and GMRES for nearly symmetric matrices. *SIAM Journal on Matrix Analysis and Applications*, 30(1):102–120, 2008.

[9] Suzanne C. Brenner and L. Ridgway Scott. *The mathematical theory of finite element methods*, volume 15 in series Texts in Applied Mathematics. Springer, New York, 2008.

[10] Claude Brezinski. Schur complements and applications in numerical analysis. In Fuzhen Zhang, editor, *The Schur Complement and Its Applications*, pages 227–258. Springer, New York, 2005.

[11] Kevin Burrage, Jocelyne Erhel, Bert Pohl, and Alan B. Williams. A deflation technique for linear systems of equations. *SIAM Journal on Scientific Computing*, 19(4):1245–1260, 1998.

[12] Stephen L. Campbell, Ilse C. F. Ipsen, C. Tim Kelley, and Carl D. Meyer. GMRES and the minimal polynomial. *BIT*, 36(4):664–675, 1996.

[13] Andrew Chapman and Yousef Saad. Deflated and augmented Krylov subspace techniques. *Numerical Linear Algebra with Applications*, 4(1):43–66, 1997.

[14] Francoise Chatelin. *Eigenvalues of Matrices*. Wiley, Chichester, UK, 1993.

[15] Yunmei Chen and Vladimir Rokhlin. On the inverse scattering problem for the Helmholtz equation in one dimension. *Inverse Problems*, 8:365–391, 1992.

[16] Paul Concus and Gene H. Golub. A generalized conjugate gradient method for nonsymmetric systems of linear equations. In *Computing Methods in Applied Sciences and Engieering, Part I*, volume 134 of *Lecture Notes in Economics and Mathematical Systems*, pages 56–65. Springer, Berlin, 1976.

[17] Richard W. Cottle. Manifestations of the Schur complement. *Linear Algebra and its Applications*, 8:189–211, 1974.

[18] James Daniel, William B. Gragg, Linda Kaufman, and Gilbert W. Stewart. Reorthogonalization and stable algorithms for updating the Gram-Schmidt QR factorization. *Mathematics of Computation*, 30(136):772–795, 1976.

[19] Dean Darnell, Ronald B. Morgan, and Walter Wilcox. Deflated GMRES for systems with multiple shifts and multiple right-hand sides. *Linear Algebra and its Applications*, 429(10):2415–2434, 2008.

[20] Ron S. Dembo, Stanley C. Eisenstat, and Trond Steihaug. Inexact Newton methods. *SIAM Journal on Numerical Analysis*, 19(2):400–408, 1982.

[21] Tobin A. Driscoll, Kim-Chuan Toh, and Lloyd N. Trefethen. From potential theory to matrix iterations in six steps. *SIAM Review*, 40:547–578, 1998.

[22] Iain. S. Duff, Albert M. Erisman, and John K. Reid. *Direct Methods for Sparse Matrices*. Clarendon Press, Oxford, 1986.

[23] Michael Eiermann, Oliver G. Ernst, and Olaf Schneider. Analysis of acceleration strategies for restarted minimal residual methods. *Journal of Computational and Applied Mathematics*, 123(1-2):261–292, 2000.

[24] Stanley C. Eisenstat, Howard C. Elman, and Martin H. Schultz. Variational iterative methods for nonsymmetric systems of linear equations. *SIAM Journal on Numerical Analysis*, 20(2):345–357, 1983.

[25] Mark Embree. The tortoise and the hare restart GMRES. *SIAM Review*, 45(2):259–266, 2003.

[26] Jocelyne Erhel, Kevin Burrage, and Bert Pohl. Restarted GMRES preconditioned by deflation. *Journal of Computational and Applied Mathematics*, 69(2):303–318, 1996.

[27] Vance Faber and Thomas Manteuffel. Necessary and sufficient conditions for the existence of a conjugate gradient method. *SIAM Journal on Numerical Analysis*, 21(2):352–362, 1984.

[28] Bernd Fischer. *Polynomial Based Iteration Methods for Symmetric Linear Systems*. Chichester, New York, 1996. SIAM (Reprinted) 2011.

[29] Bernd Fischer, Allison Ramage, David J. Silvester, and Andy J. Wathen. Minimum residual methods for augmented systems. *BIT. Numerical Mathematics*, 38:527–543, 1998.

[30] Roland W. Freund. Quasi-kernel polynomials and their use in non-Hermitian matrix iterations. *Journal of Computational and Applied Mathematics*, 43(1-2):135–158, 1992.

[31] Roland W. Freund and Noël M. Nachtigal. QMR: a quasi-minimal residual method for non-Hermitian linear systems. *Numerische Mathematik*, 60(3):315–339, 1991.

[32] Andreas Frommer. BiCGStab(l) for families of shifted linear systems. *Computing*, 70(2):87–109, 2003.

[33] Andreas Frommer and Uwerice Glässner. Restarted GMRES for shifted linear systems. *SIAM Journal on Scientific Computing*, 19(1):15–26, 1998.

[34] Andreas Frommer, Stephan Güsken, Thomas Lippert, Bertold Nöckel, and Klaus Schilling. Many masses on one stroke: Economic computation of quark propagators. *International Journal of Modern Physics C*, 6:627–638, 1995.

[35] Alan George and Joseph W. Liu. *Computer Solution of Large Sparse Positive Definite Systems. 1981.* Prentice-Hall, Englewood Cliffs, New Jersey, 1981.

[36] Gene H. Golub and Charles F. Van Loan. *Matrix Computations.* Johns Hopkins University Press, Baltimore, third edition, 1996.

[37] Serge Goossens and Dirk Roose. Ritz and harmonic Ritz values and the convergence of FOM and GMRES. *Numerical Linear Algebra with Applications*, 6(4):281–293, 1999.

[38] Anne Greenbaum. *Iterative Methods for Solving Linear Systems.* SIAM, Philadelphia, 1997.

[39] Anne Greenbaum, Vlastimil Pták, and Zdeněk Strakoš. Any nonincreasing convergence curve is possible for GMRES. *SIAM Journal on Matrix Analysis and Applications*, 17:465–469, 1996.

[40] Anne Greenbaum, Miroslav Rozložník, and Zdeněk Strakoš. Numerical behaviour of the modified Gram–Schmidt GMRES implementation. *BIT. Numerical Mathematics*, 37:706–719, 1997.

[41] Martin H. Gutknecht. Block Krylov space methods for linear systems with multiple right-hand sides: an introduction. In Abul Hasan Siddiqi, Iain S. Duff, and Ole Christensen, editors, *Modern Mathematical Models, Methods and Algorithms for Real World Systems*, pages 420–447, New Delhi, 2007. Anamaya Publishers.

[42] Martin H. Gutknecht and Thomas Schmelzer. Updating the QR decomposition of block tridiagonal and block Hessenberg matrices. *Applied Numerical Mathematics*, 58(6):871–883, 2008.

[43] William W. Hager. Updating the inverse of a matrix. *SIAM Review*, 31(2):221–239, 1989.

[44] Michael A. Heroux, Andrew G. Salinger, and Laura J. D. Frink. Parallel segregated Schur complement methods for fluid density functional theories. *SIAM Journal on Scientific Computing*, 29(5):2059–2077, 2007.

[45] Magnus R. Hestenes and Eduard Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49:409–436 (1953), 1952.

[46] Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, Philadelphia, second edition, 2002.

[47] Roger A. Horn and Charles R. Johnson. *Matrix Analysis*. Cambridge University Press, Cambridge, 1985.

[48] Marko Huhtanen. A Hermitian Lanczos method for normal matrices. *SIAM Journal on Matrix Analysis and Applications*, 23(4):1092–1108, 2002.

[49] Beat Jegerlehner. Krylov space solvers for sparse linear systems. Technical Report IUHET-353, Indiana University, 1996.

[50] Wayne Joubert. On the convergence behavior of the restarted GMRES algorithm for solving nonsymmetric linear systems. *Numerical Linear Algebra with Applications*, 1(5):427–447, 1994.

[51] S. A. Kharchenko and Aleksey Yu. Yeremin. Eigenvalue translation based preconditioners for the GMRES(k) method. *Numerical Linear Algebra with Applications*, 2(1):51–77, 1995.

[52] Misha E. Kilmer and Eric de Sturler. Recycling subspace information for diffuse optical tomography. *SIAM Journal on Scientific Computing*, 27(6):2140–2166, 2006.

[53] Sabrina Kirchner. IDR-Verfahren zur Lösung von Familien geshifteter linearer Gleichungssysteme. Master's thesis, Bergische Universität Wuppertal, Department of Mathematics, Wuppertal, Germany, 2011.

[54] Cornelius Lanczos. Solution of systems of linear equations by minimized-iterations. *Journal of Research of the National Bureau of Standards*, 49:33–53, 1952.

[55] Mathworks. MATLAB, 2011. Version R2011b.

[56] Ronald B. Morgan. Computing interior eigenvalues of large matrices. *Linear Algebra and its Applications*, 154/156:289–309, 1991.

[57] Ronald B. Morgan. A restarted GMRES method augmented with eigenvectors. *SIAM Journal on Matrix Analysis and Applications*, 16(4):1154–1171, 1995.

[58] Ronald B. Morgan. Implicitly restarted GMRES and Arnoldi methods for nonsymmetric systems of equations. *SIAM Journal on Matrix Analysis and Applications*, 21(4):1112–1135, 2000.

[59] Ronald B. Morgan. GMRES with deflated restarting. *SIAM Journal on Scientific Computing*, 24(1):20–37, 2002.

[60] Ronald B. Morgan and Min Zeng. Harmonic projection methods for large nonsymmetric eigenvalue problems. *Numerical Linear Algebra with Applications*, 5(1):33–55, 1998.

[61] Roy A. Nicolaides. Deflation of conjugate gradients with applications to boundary value problems. *SIAM Journal on Numerical Analysis*, 24(2):355–365, 1987.

[62] A. A. Nikishin and Aleksey Yu. Yeremin. Variable block CG algorithms for solving large sparse symmetric positive definite linear systems on parallel computers, I: general iterative scheme. *SIAM Journal on Matrix Analysis and Applications*, 16:1135, 1995.

[63] Dianne P. O'Leary. The block conjugate gradient algorithm and related methods. *Linear Algebra and its Applications*, 29:293–322, 1980.

[64] Chris C. Paige and Michael A. Saunders. Solutions of sparse indefinite systems of linear equations. *SIAM Journal on Numerical Analysis*, 12(4):617–629, 1975.

[65] Christopher C. Paige. A useful form of unitary matrix obtained from any sequence of unit 2-norm n-vectors. *SIAM Journal on Matrix Analysis and Applications*, 31:565–583, 2009.

[66] Christopher C. Paige, Miroslav Rozložník, and Zdeněk Strakoš. Modified Gram–Schmidt (MGS), least squares, and backward stability of MGS-GMRES. *SIAM Journal on Matrix Analysis and Applications*, 28:264–284, 2006.

[67] Michael Parks. Sandia national lab professional website: Software page, 2011. Available at `http://www.sandia.gov/~mlparks/software.html`.

[68] Michael L. Parks, Eric de Sturler, Greg Mackey, Duane D. Johnson, and Spandan Maiti. Recycling Krylov subspaces for sequences of linear systems. *SIAM Journal on Scientific Computing*, 28(5):1651–1674, 2006.

[69] Beresford N. Parlett. *The symmetric eigenvalue problem.* Prentice-Hall, Englewood Cliffs, New Jersey, 1980. SIAM (Reprinted) 1998.

[70] Sergio Pissanetzky. *Sparse matrix technology.* Academic Press London, New York, 1984.

[71] Axel Ruhe. Implementation aspects of band Lanczos algorithms for computation of eigenvalues of large sparse symmetric matrices. *Mathematics of Computation*, 33(146):680–687, 1979.

[72] Yousef Saad. On the Lanczos method for solving symmetric linear systems with several right-hand sides. *Mathematics of Computation*, 48(178):651–662, 1987.

[73] Yousef Saad. A flexible inner-outer preconditioned GMRES algorithm. *SIAM Journal on Scientific Computing*, 14(2):461–469, 1993.

[74] Yousef Saad. Analysis of augmented Krylov subspace methods. *SIAM Journal on Matrix Analysis and Applications*, 18(2):435–449, 1997.

[75] Yousef Saad. *Iterative methods for sparse linear systems.* SIAM, Philadelphia, Second edition, 2003.

[76] Yousef Saad and Martin H. Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 7:856–869, 1986.

[77] Jack Sherman and Winifred J. Morrison. Adjustment of an inverse matrix corresponding to a change in one element of a given matrix. *The Annals of Mathematical Statistics*, 21:124–127, 1950.

[78] Josef Sifuentes. *Preconditioned Iterative Methods for Inhomogeneous Acoustic Scattering Applications.* PhD thesis, Rice University, Department of Computational and Applied Mathematics, Houston, April 2010.

[79] Valeria Simoncini. On the convergence of restarted Krylov subspace methods. *SIAM Journal on Matrix Analysis and Applications*, 22(2):430–452, 2000.

[80] Valeria Simoncini. Restarted full orthogonalization method for shifted linear systems. *BIT. Numerical Mathematics*, 43(2):459–466, 2003.

[81] Valeria Simoncini and Efstratios Gallopoulos. An iterative method for nonsymmetric systems with multiple right-hand sides. *SIAM Journal on Scientific Computing*, 16(4):917–933, 1995.

[82] Valeria Simoncini and Efstratios Gallopoulos. Convergence properties of block GMRES and matrix polynomials. *Linear Algebra and its Applications*, 247:97–119, 1996.

[83] Valeria Simoncini and Daniel B. Szyld. The effect of non-optimal bases on the convergence of Krylov subspace methods. *Numerische Mathematik*, 100(4):711–733, 2005.

[84] Valeria Simoncini and Daniel B. Szyld. On the occurrence of superlinear convergence of exact and inexact Krylov subspace methods. *SIAM Review*, 47(2):247–272, 2005.

[85] Valeria Simoncini and Daniel B. Szyld. Recent computational developments in Krylov subspace methods for linear systems. *Numerical Linear Algebra with Applications*, 14(1):1–59, 2007.

[86] Valeria Simoncini and Daniel B. Szyld. Interpreting IDR as a Petrov-Galerkin method. *SIAM Journal on Scientific Computing*, 32:1898–1912, 2010.

[87] Peter Sonneveld and Martin B. van Gijzen. IDR(s): a family of simple and fast algorithms for solving large nonsymmetric systems of linear equations. *SIAM Journal on Scientific Computing*, 31(2):1035–1062, 2008.

[88] Gilbert W. Stewart. *Matrix Algorithms: Eigensystems*, volume 2 of 5 in series. SIAM, Philadelphia, 2001.

[89] Gilbert W. Stewart and Ji-guang Sun. *Matrix Perturbation Theory.* Academic Press, San Diego, 1990.

[90] Eric de Sturler. Nested Krylov methods based on GCR. *Journal of Computational and Applied Mathematics*, 67(1):15–41, 1996.

[91] Eric de Sturler. Truncation strategies for optimal Krylov subspace methods. *SIAM Journal on Numerical Analysis*, 36(3):864–889, 1999.

[92] Eric de Sturler. Convergence bounds for approximate invariant subspace recycling for sequences of linear systems. In *Program of the Householder Symposium XVIII on Numerical Linear Algebra*, pages 51–52, 2011. Abstractbecker.

[93] Daniel B. Szyld and Olof B. Widlund. Variational analysis of some conjugate gradient methods. *East-West Journal of Numerical Mathematics*, 1:51–74, 1993.

[94] Lloyd N. Trefethen and David Bau. *Numerical Linear Algebra*. SIAM, Philadelphia, 1997.

[95] Henk A. van der Vorst. Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing*, 13(2):631–644, 1992.

[96] Henk A. van der Vorst and Kees Vuik. The superlinear convergence behaviour of GMRES. *Journal of Computational and Applied Mathematics*, 48(3):327–341, 1993.

[97] Martin B. van Gijzen. IDR(s) website, Last Accessed February 29, 2012. `http://ta.twi.tudelft.nl/nw/users/gijzen/IDR.html`.

[98] Raf Vandebril and Gianna M. Del Corso. An implicit multishift QR-algorithm for Hermitian plus low rank matrices. *SIAM Journal on Scientific Computing*, 32:2190–2212, 2010.

[99] Brigitte Vital. *Etude de quelques méthodes de résolution de problemes linéaires de grande taille sur multiprocesseur*. PhD thesis, Université de Rennes, 1990.

[100] Olof B. Widlund. A Lanczos method for a class of nonsymmetric systems of linear equations. *SIAM Journal on Numerical Analysis*, 15:801–812, 1978.

[101] Max A. Woodbury. *Inverting modified matrices*. Statistical Research Group, Memo. Rep. no. 42. Princeton University, Princeton, 1950.

[102] Elizabeth L. Yip. A note on the stability of solving a rank-p modification of a linear system by the Sherman-Morrison-Woodbury formula. *SIAM Journal on Scientific and Statistical Computing*, 7(2):507–513, 1986.