ABSTRACT OF DISSERTATION

Patrick D. Quillen

The Graduate School

University of Kentucky

2005

GENERALIZATIONS OF AN INVERSE FREE KRYLOV SUBSPACE METHOD FOR THE SYMMETRIC GENERALIZED EIGENVALUE PROBLEM

ABSTRACT OF DISSERTATION

A dissertation submitted in partial fulfillment of the requirements of the degree of Doctor of Philosophy in the College of Arts and Sciences at the University of Kentucky

By

Patrick D. Quillen

Lexington, Kentucky

Director: Dr. Qiang Ye, Department of Mathematics

Lexington, Kentucky

2005

ABSTRACT OF DISSERTATION

GENERALIZATIONS OF AN INVERSE FREE KRYLOV SUBSPACE METHOD FOR THE SYMMETRIC GENERALIZED EIGENVALUE PROBLEM

Symmetric generalized eigenvalue problems arise in many physical applications and frequently only a few of the eigenpairs are of interest. Typically, the problems are large and sparse, and therefore traditional methods such as the QZ algorithm may not be considered. Moreover, it may be impractical to apply shift-and-invert Lanczos, a favored method for problems of this type, due to difficulties in applying the inverse of the shifted matrix.

With these difficulties in mind, Golub and Ye developed an inverse free Krylov subspace algorithm for the symmetric generalized eigenvalue problem. This method does not rely on shift-and-invert transformations for convergence acceleration, but rather a preconditioner is used. The algorithm suffers, however, in the presence of multiple or clustered eigenvalues. Also, it is only applicable to the location of extreme eigenvalues.

In this work, we extend the method of Golub and Ye by developing a block generalization of their algorithm which enjoys considerably faster convergence than the usual method in the presence of multiplicities and clusters. Preconditioning techniques for the problems are discussed at length, and some insight is given into how these preconditioners accelerate the method. Finally we discuss a transformation which can be applied so that the algorithm extracts interior eigenvalues. A preconditioner based on a QR factorization with respect to the B^{-1} inner product is developed and applied in locating interior eigenvalues.

KEYWORDS: symmetric generalized eigenvalue problem, block Krylov subspace methods, interior eigenvalues, preconditioning, incomplete factorizations

Patrick D. Quillen

27 October 2005

GENERALIZATIONS OF AN INVERSE FREE KRYLOV SUBSPACE METHOD FOR THE SYMMETRIC GENERALIZED EIGENVALUE PROBLEM

By

Patrick D. Quillen

Dr. Qiang Ye
Director of Dissertation

Dr. Serge Ochanine
Director of Graduate Studies

27 October 2005

DISSERTATION

Patrick D. Quillen

The Graduate School

University of Kentucky

2005

GENERALIZATIONS OF AN INVERSE FREE KRYLOV SUBSPACE METHOD FOR THE SYMMETRIC GENERALIZED EIGENVALUE PROBLEM

DISSERTATION

A dissertation submitted in partial fulfillment of the requirements of the degree of Doctor of Philosophy in the College of Arts and Sciences at the University of Kentucky

By

Patrick D. Quillen

Lexington, Kentucky

Director: Dr. Qiang Ye, Department of Mathematics

Lexington, Kentucky

2005

ACKNOWLEDGMENTS

I would first like to thank the members of my committee for their unending patience with me. They are Russell Brown, Michael Cavagnero, Sung Ha Kang, and Jun Zhang. Most of all, I thank my advisor Qiang Ye for the opportunity to work with him and the support he has given me during my time at Kentucky.

Secondly, I have been fortunate enough to receive some software from Miroslav Tůma, Michele Benzi, Menno Verbeek, and Jane Cullum. Without their software, I would have spent many nights writing my own implementations of their algorithms.

I have spent three summers under the capable tutelage of Noël Nachtigal at Sandia National Laboratories. He has given me more than I can imagine, and there are not words to express my gratitude.

Also, Professor Tom Hayden has given me continuous support during my time at Kentucky, both in and out of the classroom. I am grateful for a truly wonderful mentor and friend. My academic experience was highlighted and heavily influenced by Peter Hislop, John Lewis, and Ren-cang Li. I will carry the lessons I have learned in their classes throughout my career.

My friends, old and new, all deserve to be mentioned, but I'll mention just a few. My heartfelt thanks go to John Bagley, Todd Moody, Ziggy Hoffnung, Matt Ragland, Mike Dobranski, David Feinauer, Jakayla Robbins, and Jennifer Eli. They have always been there for me when I needed them.

Most of all, I must thank my family for supporting me and believing in me. Thank you. And I don't think I would have made it through this, especially the last bit, without the love and support of my wife Deddie, who has been more patient with me, and given me more than everyone else put together.

Portions of this work have been supported by NSF Grant CCR-0098133.

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Preliminaries

1.1 Introduction

We are interested in the computation of a few eigenpairs of the symmetric generalized pencil (A, B). That is, given two symmetric matrices, A and B, such that B is positive definite, we wish to compute a few scalar/vector pairs (λ, x) such that

$$Ax = \lambda Bx. \tag{1.1}$$

The symmetric generalized eigenvalue problems arises in many physical applications, such as the dynamic analysis of structures and electronic structure calculations. The matrices involved are typically very large and sparse; many of the entries in the matrices are zero. Due to the size of the matrices, classical methods such as the QZ algorithm are typically infeasible, since over the course of the algorithm, sparsity will generally be lost, resulting in the need to store much more data than is practical. Also, in some cases, the action of the matrix A or B may only be available as a function, and it may not be practical to construct the matrix representing the underlying linear operator. For these kinds of problems, algorithms which rely only on the ability to apply a matrix to a vector are very attractive.

The Lanczos algorithm is one of a class of methods called Krylov subspace methods which compute approximate eigenpairs of (A, B) while only requiring the actions of the matrices A and B on vectors. The algorithm works by projecting the eigenvalue problem onto the Krylov subspace

$$\mathcal{K}_m(H, x) = \operatorname{span}\{x, Hx, H^2x, \ldots, H^{m-1}x\} \tag{1.2}$$

where $H := B^{-1}A$, and x is some initial vector. The Lanczos process develops a basis Z of $\mathcal{K}_m(H, x)$ and then computes the eigenpairs (θ, u) of the projected problem (Z^*AZ, Z^*BZ). The eigenvalues θ are called Ritz values and are taken as approximate eigenvalues of (A, B) with corresponding approximate eigenvectors Zu, which are called Ritz vectors. In constructing Z, explicit formation of H is not necessary.

Instead, the Lanczos process may be carried out with respect to the B inner product to construct a basis of the subspace (1.2) while only requiring a multiplication by A and an application of B^{-1} at each step. The convergence behavior of the Lanczos algorithm has been widely studied, and it is known that the method produces good approximations to the well-separated extreme eigenpairs rather quickly. When the extreme eigenvalues are not well-separated, however, the rate of convergence of the Ritz values is typically rather poor.

To accelerate convergence, robust implementations of the Lanczos algorithm use a shift-and-invert technique to get better spectral separation. That is, the Lanczos algorithm is applied to the pencil

$$B(A - \mu B)^{-1} Bx = \hat{\lambda} Bx, \tag{1.3}$$

and the eigenvalues $\hat{\lambda}$ are then transformed back to the eigenvalues of (A, B). Here μ is some selected target for the location of the eigenvalues. This algorithm requires the solution of the shifted linear system $(A - \mu B)y = Bx$ at each step, but for that price, it typically enjoys very rapid convergence. Solving the shifted linear systems by direct methods, however, may be infeasible due to the size of the problem at hand. In these cases, iterative methods may be employed to compute approximate solutions of the shifted linear systems. It has been observed, however, that the Lanczos algorithm is extremely sensitive to perturbations, especially in the early stages of the iteration, and therefore the shifted systems must be solved very accurately. This limits the practicality of the shift-and-invert approach. We remark that application of the shift-and-invert transformation is sometimes called preconditioning. Generally, the term preconditioning refers to applying some transformation to the original problem which results in a different problem which is more easily handled by a given method.

Other methods, such as the JDQZ method [17], also make use of a shift-and-invert strategy, however, they tolerate much lower accuracy approximations to the solutions of the shifted linear systems. The JDQZ method in particular, takes an initial approximation to the desired eigenpair and successively solves correction equations to expand the search subspace. The idea is that the correction equations assist in choosing the "right" vector for expansion and thereby refine the approximations to the eigenpair in an optimal way. Here, preconditioning techniques are applicable in approximating solutions to the correction equations. Frequently, Krylov subspace methods such as GMRES or BiCG are used, and preconditioning schemes for these methods have been widely studied [5], [52]. In this framework, the preconditioner has little to do with the eigenvalue problem.

On the other hand, gradient type methods, such as the steepest descent method may be used to locate the algebraically smallest eigenvalues. Such methods do not require inversion of B in any form, however, they frequently suffer from extremely slow convergence when the extreme eigenvalues are not well-separated. Some gradient methods that have been studied include the LOBPCG method [34], the DACG method [19], and a variation of inverse iteration called preconditioned inverse iteration [42]. Here, instead of applying a shift-and-invert transformation, a different kind of preconditioning scheme is used to accelerate the iterations. Frequently, preconditioners which work well for solving linear systems with the matrix A are chosen and applied in the iterations for the solution of the eigenproblem (A, B). Although in many cases these methods work well, the role of the preconditioner in accelerating the eigensolver is unclear.

With these considerations in mind, Golub and Ye [25] presented and analyzed an inverse-free Krylov subspace method for the computation of the extreme eigenvalues of a symmetric generalized pencil (A, B). As with gradient type methods, the algorithm does not require application of B^{-1} at anytime, however, unlike many gradient methods, this algorithm takes advantage of the optimality offered by Krylov subspace approximations. Moreover, the analysis in [25] indicates how a transformation may be applied to accelerate the convergence of the method to the extreme eigenvalues. Again, we will refer to such transformations as preconditioning. Due to the nature of vector iterations, the inverse free Krylov subspace algorithm suffers from poor convergence in the presence of multiple and tightly clustered eigenvalues, though. Often, the presence of such clusters will also limit the effectiveness of preconditioning transformations.

The goals of this present work are as follows.

1. Develop a block generalization of the method of [25] which enjoys similar convergence characteristics, even in the presence of multiple and clustered eigenvalues. Moreover, the block generalization should be able to be accelerated by a well-defined preconditioning scheme.

2. Gain a deeper understanding of exactly how purely algebraic preconditioners such as those stemming from incomplete factorizations accelerate the convergence of the method.

3. Devise a scheme for the computation of interior eigenvalues, or more specifically, those eigenvalues nearest to some target μ.

Regarding the interior eigenvalue problem, we note that Ritz approximations from a given subspace are frequently very good approximations for extreme eigenvalues and very poor for interior ones. Even if a Ritz value is near an interior eigenvalue, it may be more proper to view that Ritz value as one which is on its way to the extreme of the spectrum, but simply has not yet made it. As a result, for large eigenvalue problems, the only robust method for computing interior eigenvalues currently is the shift-and-invert Lanczos method.

The remainder of this thesis is organized as follows. Chapter 1 contains necessary background material. Some definitions, notation, and classical results regarding the symmetric generalized eigenvalue problem are presented in Section 1.2. Krylov subspace methods for eigenvalue approximation are the focus of Section 1.3.

Chapter 2 begins with a description of the inverse-free Krylov subspace algorithm of Golub and Ye. In Section 2.3 we develop a block generalization of the inverse free method and we discuss many pertinent implementation details. We derive this new method by first generalizing the method of Golub and Ye on a vector-by-vector basis, and by observing an Arnoldi-like relationship, we arrive at the block generalization. We provide many numerical examples showing the ability of the block algorithm to compute eigenvalues in clusters and with multiplicities.

Chapter 3 is concerned with accelerating the convergence of the algorithms described in Chapter 2 by means of preconditioning. The chapter opens with a review of some common preconditioning techniques, followed by a discussion about how these techniques may be applied to solving the eigenvalue problem. In particular, we demonstrate how to apply preconditioning transformations to the block algorithm of Chapter 2 and we detail the effects of three different preconditioning schemes on the rates of convergence of the inverse free algorithms. Ideas from preconditioning for linear systems regarding the accuracy and stability of incomplete factorization preconditioners are applied here in the context of our eigensolvers. Chapter 3 closes with many numerical examples showing the favorable effects of the preconditioning schemes discussed.

Chapter 4 discusses the application of the algorithms described in Chapter 2 to the problem of finding eigenvalues of the pencil (A, B) nearest to some target $\mu \in \mathbb{R}$. We apply a simple transformation to the original problem to make the eigenvalues of interest extreme eigenvalues. Here, due to the conditioning of the transformed problem, preconditioning becomes necessary for these methods to be considered viable at all, and common preconditioning schemes falter. Motivated by the normal equations like structure of the matrices under consideration, we develop a new algorithm for computing the QR factorization of a given matrix with respect to the B^{-1} inner product via Householder like reflectors. We briefly discuss an approach to using this factorization as an incomplete factorization preconditioner, and we apply the incomplete factor as a preconditioner for the interior formulation. Examples showing the viability of these approaches are given.

Finally, Chapter 5 contains concluding remarks and insights into directions for future work in this subject.

1.2 Classical Results

Most of the material presented here can be found in any standard text. See, for example, [15], [23], [29], [46].

1.2.1 Simultaneous diagonalization of two quadratic forms

Here we recall a result which echoes the spectral theorem for Hermitian matrices. A major difference when dealing with symmetric pencils is that the eigenvectors are no longer orthogonal, but rather B-orthogonal.

Definition 1.2.1. *Let $x, y \in \mathbb{R}^n$. If $B \in \mathbb{R}^{n\times n}$ is symmetric positive definite The quadratic form*

$$\langle x, y\rangle_B := x^* B y$$

induces an inner product on $\mathbb{R}^n$ called the B inner product. Two vectors $x, y \in \mathbb{R}^n$ are said to be B-orthogonal if $\langle x, y\rangle_B = 0$. The notation $x \perp_B y$ indicates that x and y are B-orthogonal. With this inner product there is an associated norm called the B-norm:

$$\|x\|_B := \sqrt{\langle x, x\rangle_B}$$

Finally, a collection of vectors is called B-orthonormal if the vectors are pairwise B-orthogonal and have unit B-norm.

We may also consider B-unitary and B-orthogonal matrices.

Definition 1.2.2. *A matrix Q is said to be B-unitary if*

$$Q^* B Q = B \tag{1.4}$$

That is, B-unitary matrices are those which preserve the B-inner product. A matrix Q is said to be B-orthogonal if

$$Q^* B Q = D \tag{1.5}$$

where D is some diagonal matrix. If $D = I$, we say that Q is B-orthonormal.

Throughout, we will discuss B-orthogonal matrices, and we will consider specific applications of B-unitary operators in Chapter 4.

With these definitions in mind, we present the simultaneous diagonalization theorem. See [46] for a proof of this theorem.

Theorem 1.2.3. *Suppose A, B are symmetric matrices and B is positive definite. Then there exists an invertible matrix X (in fact many matrices) such that*

$$X^* A X \quad \text{and} \quad X^* B X$$

are both real and diagonal.

We may, in fact, choose X such that $X^*BX = I$. That is, the eigenvectors of (A, B) are B-orthonormal and not merely B-orthogonal. Throughout this work the eigenpairs of (A, B) will be denoted by $(\lambda_i, x_i), 1 \leq i \leq n$ and ordered such that $\lambda_1 \leq \lambda_2 \leq \cdots \leq \lambda_n$. The eigenvectors will be scaled to have unit B-norm unless otherwise specified.

1.2.2 The Rayleigh Quotient and Courant-Fischer

Of extreme importance in the study of the symmetric eigenvalue problem is a quantity called the Rayleigh Quotient.

Definition 1.2.4. *The Rayleigh quotient of x with (A, B) is defined by*

$$\rho(x; A, B) = \frac{x^*Ax}{x^*Bx}.$$

We frequently shorten the notation to $\rho(x)$ when A, B are clear from context.

The Rayleigh Quotient enjoys many properties, which will be discussed as necessary. The following properties will be exploited throughout this work.

Lemma 1.2.5. *Let (A, B) be a definite pencil with eigenpairs $(\lambda_i, x_i), i = 1, \ldots, n$ such that $\lambda_1 \leq \cdots \leq \lambda_n$, and $\|x_i\|_B = 1$.*

1. $\lambda_1 \leq \rho(x; A, B) \leq \lambda_n$
2. *Let $\mathcal{X}_j = \operatorname{span}\{x_1, x_2, \ldots, x_j\}$. Then*

$$\lambda_j = \max_{x \in \mathcal{X}_j} \rho(x; A, B) = \min_{y \perp_B \mathcal{X}_{j-1}} \rho(y; A, B)$$

The above facts about the Rayleigh quotient together with an observation about subspaces of a finite dimensional vector space lead to the stunning Courant-Fischer Minmax principle. The principle holds for self-adjoint operators in more general Hilbert space, however, we will be confined to finite dimensional space throughout the course of this work. One may consult [29], [46], or one of many other texts for a proof of this result.

Theorem 1.2.6 (Courant-Fischer Minmax Principle). *Let $\{\lambda_j\}_{j=1}^n$ be the eigenvalues of the pencil (A, B) ordered such that $\lambda_1 \leq \cdots \leq \lambda_n$. Letting $\mathcal{S}$ and $\mathcal{C}$ denote subspaces of $\mathbb{R}^n$ with dimensions denoted by subscript, we have*

$$\lambda_j = \min_{\mathcal{S}_j} \max_{x \in \mathcal{S}_j} \rho(x; A, B) = \max_{\mathcal{C}_{j-1}} \min_{y \perp_B \mathcal{C}_{j-1}} \rho(y; A, B)$$

1.2.3 The Rayleigh-Ritz Method

The Rayleigh-Ritz procedure is a classical method for approximating the eigenvalues of a symmetric matrix, or symmetric definite pencil, and is motivated by the following two lemmas. The first states that the distance between some eigenvalue of the pencil (A, B) and a given scalar σ is bounded by the B^{-1}-norm of the residual and the second shows that among all scalars σ, the Rayleigh quotient minimizes the residual. Again, both of these results may be found in standard texts such as [15], [46], [50].

Lemma 1.2.7. *There exists an eigenvalue λ of (A, B) such that for any $u \in \mathbb{R}^n$ and any $\sigma \in \mathbb{R}$ we have*

$$|\lambda - \sigma| \leq \frac{\|(A - \sigma B)u\|_{B^{-1}}}{\|Bu\|_{B^{-1}}} \tag{1.6}$$

Lemma 1.2.8. *For any vector $u \in \mathbb{R}^n$ and scalar $\sigma \in \mathbb{R}$*

$$\|(A - \rho(u)B)u\|_{B^{-1}} \leq \|(A - \sigma B)u\|_{B^{-1}}$$

Note further that $\|(A - \rho(u)B)u\|^2_{B^{-1}} = \|Au\|^2_{B^{-1}} - |\rho(u)|^2\|Bu\|^2_{B^{-1}}$

The Rayleigh-Ritz procedure is presented in Algorithm 1.1.

Algorithm 1.1 Rayleigh-Ritz Procedure

```
Input: Symmetric $A$, symmetric positive definite $B$, a subspace $\mathcal{S}_m$.
1: If necessary, construct a basis $Z$ of $\mathcal{S}_m$.
2: Form $A_m = Z^*AZ$ and $B_m = Z^*BZ$.
3: Compute the eigenpairs $(\theta_i, v_i)$ of $(A_m, B_m)$.
4: Set $y_i = Zv_i$. Approximate eigenpairs of $(A, B)$ are $(\theta_i, y_i)$.
```

The eigenvalues $\{\theta_i\}_{i=1}^m$ of the pencil (Z^*AZ, Z^*BZ) are called *Ritz values* of (A, B) with respect to $\mathcal{S}_m$. Similarly, the vectors $y_i = Zv_i, 1 \leq i \leq m$ are called *Ritz vectors* and the pair (θ_i, y_i) is called a *Ritz pair* of (A, B) with respect to $\mathcal{S}_m$.

Should a B-orthogonal basis be constructed in step one of Algorithm 1.1, the need for the computation of B_m in step two is obviated and the problem in step three is reduced to the standard symmetric eigenvalue problem for A_m.

It is straightforward to show that $\lambda_i \leq \theta_i$ for $i : 1 \leq i \leq m$. Moreover, it can be shown that there exist m eigenvalues of (A, B) such that the distance from each of those eigenvalues to each of the Ritz values is smaller than a constant times the residual norm. This result echoes a result found in [46].

Theorem 1.2.9. *Let Z be any $n \times m$ B-orthonormal matrix and define $H = Z^*AZ$ and $R = AZ - BZH$. There are m eigenvalues of (A, B) denoted $\lambda_{j(i)}$, $1 \leq i \leq m$ such that*

$$|\theta_i - \lambda_{j(i)}| \leq \left(\|B^{-1}\|_2\right)^{\frac{1}{2}} \|R\|_2$$

In fact, if something more is known about the pencil (A, B), we may show something stronger than Lemma 1.2.7 and Theorem 1.2.9. The result that follows is similar to one found in [46] and also in [63].

Theorem 1.2.10. *Let $\lambda_1 \leq \cdots \leq \lambda_n$ be the eigenvalues of a symmetric definite pencil (A, B) and let $Y \in \mathbb{R}^{n \times p}$ be a collection of B-orthonormal Ritz vectors associated with Ritz values $\theta_1 \leq \cdots \leq \theta_p$. If $\theta_p < \lambda_{p+1}$ then*

$$|\theta_i - \lambda_i| \leq \|B^{-1}\|_2 \frac{\|R\|_2^2}{|\theta_p - \lambda_{p+1}|}$$

where $R = AY - BY\Theta$.

We point out that Theorem 1.2.9 and Theorem 1.2.10 are included as justification for the use of residual norms as stopping criteria in the algorithms presented later.

1.3 Krylov Subspace Methods

Krylov subspace methods for the approximation of eigenvalues simply apply the Rayleigh-Ritz method (Algorithm 1.1) to a Krylov subspace. A Krylov subspace of dimension less than or equal to m associated with a matrix A and a vector $x \neq 0$ is defined by

$$\mathcal{K}_m(A, x) := \operatorname{span}\{x, Ax, A^2x, \ldots, A^{m-1}x\}. \tag{1.7}$$

Each vector $u \in \mathcal{K}_m(A, x)$ may be represented as $u = p(A)x$ where p is a polynomial of degree less than or equal to $m - 1$. Of course, the dimension of the Krylov space depends on the initial vector x; namely, if x is an eigenvector of A, then the Krylov space $\mathcal{K}_m(A, x)$ has dimension one, regardless of the value of m. Furthermore, the dimension of the Krylov subspace $\mathcal{K}_m(A, x)$ is always less than or equal to the degree of the minimal polynomial of x with respect to A. It should also be pointed out that the basis $\{x, Ax, A^2x, \ldots, A^{m-1}x\}$ is never used in practice since the vectors become increasingly dependent, as the sequence $\{A^i x\}_{i=0}^{\infty}$ converges to the dominant eigenvector for most choices of x. Instead, the Arnoldi process or the Lanczos process is used to develop an orthonormal basis of the Krylov subspace $\mathcal{K}_m(A, x)$ with respect to a certain inner product. Once a suitable basis Q_m is constructed and the projected pencil $(A_m, B_m) \equiv (Q_m^* A Q_m, Q_m^* B Q_m)$ is formed, Ritz values and Ritz vectors must be extracted. We discuss the construction of basis and the quality of the extracted Ritz values, as these topics are pertinent to the discussions in the following chapters. We omit a discussion concerning the computation of the eigenvalues of the pencil (A_m, B_m) and instead point the interested reader to any one of the valuable references [23], [46], [53].

1.3.1 The Arnoldi Process

Given an inner product, $\langle \cdot, \cdot \rangle$, the Arnoldi process develops an orthonormal basis $\{q_1, q_2, \ldots, q_m\}$ of the Krylov subspace $\mathcal{K}_m(A, x)$. Each vector $q_{j+1}, 1 \leq j \leq m$ is generated by the following recurrence

$$h_{j+1,j}q_{j+1} = Aq_j - h_{1j}q_1 - h_{2j}q_2 - \cdots - h_{jj}q_j \tag{1.8}$$

where $h_{ij} = \langle q_i, Aq_j \rangle$. Letting H_m be the square matrix with entries h_{ij} and Q_m be the matrix whose j^{th} column is q_j, we may rewrite (1.8) to obtain

$$AQ_m = Q_m H_m + h_{m+1,m} q_{m+1} e_m^* \tag{1.9}$$

where e_m is the m^{th} canonical basis vector of $\mathbb{R}^m$. Equation (1.9) entirely defines the Arnoldi process. By construction, Q_m is orthogonal with respect to the inner product $\langle \cdot, \cdot \rangle$ and H_m is upper Hessenberg.

To see that the columns of Q_m produced by the Arnoldi process do indeed form a basis for $\mathcal{K}_m(A, x)$ we proceed inductively. Clearly q_1 forms a basis for $\mathcal{K}_1(A, x)$. Let us assume that $q_j \neq 0$ for $j : 1 \leq j \leq m$. Suppose now that $Q_j = \begin{pmatrix} q_1 & \cdots & q_j \end{pmatrix}$ forms a basis for $\mathcal{K}_j(A, x)$, and note that q_{j+1} is defined as a linear combination of the columns of Q_j and the vector Aq_j. It is easy to see therefore that $\operatorname{span}\{q_1, \ldots, q_{j+1}\} \subseteq \mathcal{K}_{j+1}(A, x)$. Now, since the collection $\{q_1, \ldots, q_{j+1}\}$ is orthonormal with respect to some inner product, then the collection is linearly independent and so it must follow that $\operatorname{span}\{q_1, \ldots, q_{j+1}\} = \mathcal{K}_{j+1}(A, x)$.

When $h_{j+1,j} = 0$ for $j : 1 \leq j \leq m$, the Arnoldi process is said to suffer breakdown. Encountering such a breakdown is fortuitous as equation (1.9) implies that the Krylov subspace is A-invariant. Ritz values extracted from the Krylov subspace will therefore be exact eigenvalues.

Because of the explicit orthogonalization of each new Arnoldi vector against all previous Arnoldi vectors, the Arnoldi process can be computationally expensive.

1.3.2 The Lanczos process

When the matrix A is symmetric with respect to the inner product used in the Arnoldi process, that is, when

$$\langle x, Ay \rangle = \langle Ax, y \rangle$$

the recurrence expressed by (1.8) is greatly simplified. Specifically, for $i \leq j - 2$, the entries h_{ij} in the upper Hessenberg matrix H_m are zero. To see this, note that due to our assumption of symmetry

$$h_{ij} = \langle q_i, Aq_j \rangle = \langle Aq_i, q_j \rangle.$$

Furthermore, as $Aq_i = \sum_{k=1}^{i+1} h_{ki} q_k$ we see that

$$\begin{aligned} h_{ij} &= \langle Aq_i, q_j \rangle \\ &= \left\langle \left(\sum_{k=1}^{i+1} h_{ki} q_k \right), q_j \right\rangle \\ &= \sum_{k=1}^{i+1} h_{ki} \langle q_k, q_j \rangle. \end{aligned}$$

Since $\langle q_k, q_j \rangle = 0$ for $k \leq j-1$, it follows that $h_{ij} = 0$ provided that $i+1 \leq j-1$. With our symmetry assumption, the defining recurrence for the Arnoldi process becomes

$$h_{j+1,j} q_{j+1} = Aq_j - h_{j,j} q_j - h_{j-1,j} q_{j-1}.$$

Noting that q_{j+1} has unit length and is orthogonal to all previous Arnoldi vectors we see that

$$h_{j+1,j} = h_{j+1,j} \langle q_{j+1}, q_{j+1} \rangle = \langle Aq_j, q_{j+1} \rangle = \langle q_j, Aq_{j+1} \rangle = h_{j,j+1}$$

and thus the matrix H_m defined by the Arnoldi process is a symmetric tridiagonal matrix; the symmetry here is with respect to the Euclidean inner product. In the case of such symmetry, the algorithm is called the Lanczos algorithm and the defining recurrence is written as

$$\beta_{j+1} q_{j+1} = Aq_j - \alpha_j q_j - \beta_j q_{j-1}. \tag{1.10}$$

Throughout this work we will refer to the method of generating a basis of a Krylov subspace by the name Lanczos if the basis may be generated by a short (three-term) recurrence. Reference to the Arnoldi process will be reserved for a method relying on a long recurrence for the generation of a basis of a Krylov subspace. Rewriting (1.10), we obtain

$$AQ_m = Q_m T_m + \beta_{j+1} q_{j+1} e_m^* \tag{1.11}$$

where Q_m is the matrix whose j^{th} column is q_j and T_m is the matrix defined by

$$T_m = \begin{pmatrix} \alpha_1 & \beta_2 & & \\ \beta_2 & \alpha_2 & \ddots & \\ & \ddots & \ddots & \beta_m \\ & & \beta_m & \alpha_m \end{pmatrix}. \tag{1.12}$$

Again, the matrix Q_m generated by the Lanczos process is orthogonal with respect to the inner product $\langle \cdot, \cdot \rangle$. Most often, the Lanczos process is applied to matrices which are symmetric with respect to the Euclidean inner product, and therefore the matrix Q_m is an orthogonal matrix with respect to the Euclidean inner product and thus

$$Q_m^* A Q_m = Q_m^* Q_m T_m + \beta_{m+1} Q_m^* q_{m+1} e_m^* = T_m.$$

That is, executing the Rayleigh-Ritz procedure for the pencil (A, I) with respect to the Krylov subspace $\mathcal{K}_m(A, x)$ reduces to computing the eigenvalues of an $m \times m$ symmetric tridiagonal matrix, which can be done very efficiently.

We are, however, interested in computing the eigenvalues of the pencil (A, B) where A and B are symmetric with respect to the Euclidean inner product. If A is also symmetric with respect to the inner product induced by the operator B, we may obtain a B-orthonormal basis of the space $\mathcal{K}_m(A, x)$ via a short-term recurrence. Again, extraction of the Ritz values of interest relies on computing some eigenvalues of the pencil (T_m, I) where T_m is symmetric tridiagonal. In general, however, the operator A will not be symmetric with respect to the B inner product and we will therefore need to extract eigenvalues of $(Q_m^* A Q_m, Q_m^* B Q_m)$ where both matrices may be dense symmetric matrices. We will discuss strategies for this problem later, however, we raise this point now to motivate a Lanczos method for (A, B) that builds a B-orthonormal basis which reduces A to tridiagonal form.

In order to generate such a basis, let M be the symmetric square root of B, and note that the pencil (A, B) is equivalent to the pencil $(M^{-1}AM^{-1}, I)$. Applying the Lanczos procedure with the Euclidean inner product to the matrix $M^{-1}AM^{-1}$ generates orthonormal Q_m and symmetric tridiagonal T_m such that

$$M^{-1}AM^{-1}Q_m = Q_m T_m + \beta_{m+1} q_{m+1} e_m^*.$$

Defining $z_j := M^{-1} q_j, 1 \leq j \leq m+1$ we have that

$$\begin{aligned} AZ_m = AM^{-1}Q_m &= MQ_m T_m + \beta_{m+1} M q_{m+1} e_m^* \\ &= M^2 Z_m T_m + \beta_{m+1} M^2 z_{m+1} e_m^* \\ &= BZ_m T_m + \beta_{m+1} B z_{m+1} e_m^* \end{aligned}$$

where we note that

$$Z_m^* B Z_m = Q_m^* M^{-1} B M^{-1} Q_m = Q_m^* Q_m = I.$$

It is readily seen, therefore, that the recurrence is simply

$$\beta_{j+1} B z_{j+1} = A z_j - \alpha_j B z_j - \beta_j B z_{j-1} \tag{1.13}$$

where

$$\alpha_j = \langle z_j, A z_j \rangle_B \quad \beta_{j+1} = \sqrt{\langle z_{j+1}, z_{j+1} \rangle_B}.$$

Notice that to obtain the Lanczos vector z_{j+1} we must apply B^{-1} either explicitly or implicitly. The need to apply B^{-1} may be a huge drawback in the implementation of this method. For example, the matrix B, though symmetric positive definite, may be difficult to factor, or, may not be available at all. That is, B may exist only as a function which takes a vector x and returns Bx. In this case, one can utilize iterative methods to approximate $B^{-1}y$ for a given vector y, but much more work than is feasible may be required to compute a sufficient approximation. Indeed, methods based on an inexact solution of $Bx = y$ have been studied, (see [10],[26]) and the Lanczos process has been found to be especially sensitive to perturbations in its early stages. With this difficulty in mind, Golub and Ye [25] have developed a Krylov subspace method for approximation of the eigenpairs of the pencil (A, B) which does not require implicit or explicit application of B^{-1}. This method is the subject of Chapter 2 and is the method we seek to extend.

1.3.3 The Quality of extracted Ritz pairs

Regardless of the process used to construct a basis of the Krylov space, we may obtain the following bound on the quality of the Ritz pairs extracted from the space. The following well-known result can be found in [14], [46], [48], [50], for example.

Theorem 1.3.1. *Consider the pencil (A, I) with eigenpairs (λ_i, q_i) such that $\lambda_1 \leq \cdots \leq \lambda_n$. Let $\theta_1 \leq \cdots \leq \theta_m$ be the Ritz values extracted from $\mathcal{K}_m(A, x)$. If $\mathcal{Q}_j = \operatorname{span}\{x_1, \ldots, x_j\}$ then for each $j = 1, \ldots, m$ we have*

$$0 \leq \theta_j - \lambda_j \leq (\lambda_n - \lambda_j) \left[\frac{\sin \angle(x, \mathcal{Q}_j)}{\cos \angle(x, q_j)} K_j \right]^2 \epsilon_{m,j}^2 \tag{1.14}$$

where

$$K_j := \prod_{i=1}^{j-1} \frac{\theta_i - \lambda_n}{\theta_i - \lambda_j}, (K_1 := 1) \quad \textit{and} \quad \epsilon_{m,j} := \min_{p \in \mathcal{P}_{m-j}; p(\lambda_j)=1} \max_{\lambda \in [\lambda_{j+1}, \lambda_n]} |p(\lambda)|.$$

Here, $\mathcal{P}_{m-j}$ is the collection of polynomials of degree less than or equal to $m - j$.

For each Ritz value, the bound (1.14) depends on the gap between λ_j and λ_{j+1}. Thus convergence can suffer when the eigenvalues at the ends of the spectrum are tightly clustered. This is part of the motivation for block Lanczos methods, which construct bases for the block Krylov subspace $\mathcal{K}_m(A, X)$, with X being an $n \times p$ orthonormal matrix, as opposed to a single vector. The major advantage of block Krylov subspace methods over their single vector counterparts is the ability to determine multiple eigenvalues. Indeed, due to the unreduced tridiagonal structure of the projection obtained from a Lanczos method, Ritz values may only have multiplicity one. In the case of clustered eigenvalues, block methods offer better convergence rates. This well-known property is summarized by the following theorem which can be found in [48] and [57], to name just two sources. The result we quote is from [57].

Theorem 1.3.2. *Consider the pencil (A, I) with eigenpairs (λ_i, q_i) such that $\lambda_1 \leq \cdots \leq \lambda_n$. Let $\theta_1 \leq \cdots \leq \theta_p$ be the Ritz values extracted from $\mathcal{K}_m(A, X)$ where $X^*X = I_p$. If $\mathcal{Q}_p = \operatorname{span}\{q_1, \ldots, q_p\}$, and $\mathcal{X} = \operatorname{span}\{x_1, \ldots, x_p\}$ then for each $j = 1, \ldots, p$ we have*

$$0 \leq \theta_j - \lambda_j \leq (\lambda_n - \lambda_j) \tan^2 \angle(\mathcal{X}, \mathcal{Q}_p) \epsilon_{m,j}^2 \tag{1.15}$$

where

$$\epsilon_{m,j} := \min_{r \in \mathcal{P}_{m-1}; r(\lambda_j)=1} \max_{\lambda \in [\lambda_{p+1}, \lambda_n]} |r(\lambda)|. \tag{1.16}$$

Here, $\mathcal{P}_{m-1}$ is the collection of polynomials of degree less than or equal to $m - 1$.

Notably, for the block Lanczos method with block size p, the rate of convergence (of the first Ritz value to the first eigenvalue) depends on the relative spectral gap $\frac{\lambda_{p+1}-\lambda_1}{\lambda_n-\lambda_1}$, thus showing the advantage of block algorithms in the presence of clustered eigenvalues.

Chapter 2

Inverse Free Algorithms for (A, B)

The goal of this chapter is the development of a block generalization of the inverse free preconditioned algorithm of Golub and Ye [25] for the simultaneous computation of p algebraically smallest eigenpairs for the pencil (A, B). The block generalization overcomes difficulties experienced by the Golub-Ye algorithm in the presence of multiple or clustered eigenvalues. Numerical examples are provided to illustrate properties of the block algorithm.

2.1 A Generic Eigensolver and Properties

Many modern iterative methods for the computation of eigenvalues of the large sparse symmetric pencils are built upon iterated Rayleigh-Ritz. That is, for each $k = 1, 2, \ldots$ a subspace $\mathcal{S}^{(k)}$ is constructed and the Rayleigh-Ritz method is performed to obtain Ritz pairs $\left(\theta_i^{(k+1)}, x_i^{(k+1)}\right)$ for $1 \leq i \leq p$. The Ritz vectors are then used as a starting point for the construction of the next subspace $\mathcal{S}^{(k+1)}$ and the process is iterated. Inspired by the discussion of [2], this process will be referred to as the generic eigensolver and a template is presented as Algorithm 2.1. Sometimes we will

Algorithm 2.1 Generic Eigensolver

```
Input: Symmetric $A$, symmetric positive definite $B$.
1: for $k = 1, 2, \ldots$ do
2:     Construct a subspace $\mathcal{S}^{(k)}$ of dimension $m \ll n$.
3:     Perform Rayleigh-Ritz on $(A, B)$ with respect to $\mathcal{S}^{(k)}$ to extract the $p$ desired
       approximate eigenpairs.
4: end for
```

refer to this process as an inner-outer iteration process as the construction of the subspace $\mathcal{S}^{(k+1)}$ will usually be performed iteratively.

Provided that each of the iterates $x_1^{(k)}, \ldots, x_p^{(k)}$ are included in the space $\mathcal{S}^{(k)}$, it is not difficult to establish the monotonicity and boundedness of the sequences of the

Ritz values.

Lemma 2.1.1. *Let $\lambda_1 \leq \cdots \leq \lambda_n$ denote the eigenvalues of (A, B). Let the Ritz values obtained from Algorithm 2.1 at the k^{th} iteration be denoted by $\theta_1^{(k+1)} \leq \cdots \leq \theta_p^{(k+1)}$ with corresponding Ritz vectors $x_1^{(k+1)}, \ldots, x_p^{(k+1)}$. Suppose further that*

$$\{x_1^{(k)}, \ldots, x_p^{(k)}\} \subseteq \mathcal{S}^{(k)}.$$

The following relations hold

$$\lambda_i \leq \theta_i^{(k+1)} \leq \theta_i^{(k)}, \quad 1 \leq i \leq p \tag{2.1}$$

$$(A - \theta_i^{(k)} B) x_i^{(k)} \perp \mathcal{S}^{(k)}, \quad 1 \leq i \leq p \tag{2.2}$$

Proof. We begin by establishing (2.2). Suppose that Z is a basis for $\mathcal{S}^{(k)}$. By construction (the Rayleigh-Ritz process), each Ritz vector $x_i^{(k)}$ is given by $x_i^{(k)} = Zu_i$, where $(\theta_i^{(k)}, u_i),\ 1 \leq i \leq p$ are eigenpairs of the pencil (Z^*AZ, Z^*BZ). Therefore,

$$Z^*AZu_i - \theta_i^{(k)} Z^*BZu_i = 0$$

Noting that any vector $y \in \mathcal{S}^{(k)}$ may be written as $y = Zc, c \in \mathbb{R}^m$, we have that

$$y^*(A - \theta_i^{(k)} B) x_i^{(k)} = c^*(Z^*AZu_i - \theta_i^{(k)} Z^*BZu_i) = 0$$

showing (2.2).

To establish (2.1) recall that by the Courant-Fischer minmax principle (Theorem 1.2.6) we have that

$$\theta_i^{(k+1)} = \min_{\mathcal{S}^i} \max_{u \in \mathcal{S}^i, u \neq 0} \frac{u^*Au}{u^*Bu} = \min_{\mathcal{S}^i} \max_{u \in \mathcal{S}^i, u \neq 0} \frac{u^*(A - \theta_i^{(k)} B)u}{u^*Bu} + \theta_i^{(k)}$$

for all subspaces $\mathcal{S}^i \subseteq \mathcal{S}^{(k)}$ of dimension i. Letting $x \in \mathcal{X} = \mathrm{span}\{x_1^{(k)}, \ldots, x_i^{(k)}\} \subseteq \mathcal{S}^{(k)}$ we have

$$x^*(A - \theta_i^{(k)} B) x_j^{(k)} = (\theta_j^{(k)} - \theta_i^{(k)}) x^* B x_j^{(k)} \quad 1 \leq j \leq i$$

As the collection $\{x_1^{(k)}, \ldots, x_i^{(k)}\}$ is a B-orthogonal set, it follows that

$$x^*(A - \theta_i^{(k)} B)x = \sum_{j=1}^{i} c_j^2 (\theta_j^{(k)} - \theta_i^{(k)}) {x_j^{(k)}}^* B x_j^{(k)} \leq 0$$

where the c_j is the j^{th} coordinate of x with respect to the basis $\{x_1^{(k)}, \ldots, x_i^{(k)}\}$. Therefore,

$$\theta_i^{(k+1)} \leq \max_{u \in \mathcal{X}, u \neq 0} \frac{u^*(A - \theta_i^{(k)} B)u}{u^*Bu} + \theta_i^{(k)} \leq \theta_i^{(k)}$$

establishing monotonicity.

That each sequence $\{\theta_i^{(k)}\}, 1 \leq i \leq p$ is bounded below by λ_i is a consequence of the Rayleigh-Ritz method. □

If we additionally assume that the residual vectors $r_i^{(k)} := Ax_i^{(k)} - \theta_i^{(k)} Bx_i^{(k)}, 1 \leq i \leq p$ are contained in $\mathcal{S}^{(k)}$, then we can show that each of the sequences $\{\theta_i^{(k)}\}_{k=1}^{\infty}$ converges to an eigenvalue of (A, B).

Lemma 2.1.2. *Take the assumptions of Lemma 2.1.1 and additionally, assume that the Ritz vectors $x_1^{(k)}, \dots, x_p^{(k)}$ are scaled such that $\|x_i^{(k)}\|_B = 1$ and assume that the residuals $r_i^{(k)} = Ax_i^{(k)} - \theta_i^{(k)} Bx_i^{(k)}$ are elements of $\mathcal{S}^{(k)}$ for $1 \leq i \leq p$.*

Then for each $j = 1, 2, \dots, p$, the sequence $\left\{\theta_j^{(k)}\right\}_{k=1}^{\infty}$ converges to some eigenvalue $\hat{\lambda}$ of (A, B) and the sequence $\left\{\|(A - \hat{\lambda}B)x_j^{(k)}\|\right\}_{k=1}^{\infty}$ converges to zero.

Proof. We follow the proof technique of Golub and Ye [25]. Begin by defining the matrices

$$\Theta^{(k)} := \operatorname{diag}(\theta_1^{(k)}, \dots, \theta_j^{(k)}) \quad \text{and} \quad X^{(k)} := \begin{pmatrix} x_1^{(k)} & \cdots & x_j^{(k)} \end{pmatrix}. \tag{2.3}$$

We note that for each k,

$$X^{(k)^*} A X^{(k)} = \Theta^{(k)} \quad \text{and} \quad X^{(k)^*} B X^{(k)} = I. \tag{2.4}$$

Since each sequence $\left\{x_i^{(k)}\right\}, 1 \leq i \leq j$ is bounded, it follows that we may develop convergent subsequences $\left\{x_i^{(k_\ell)}\right\}, 1 \leq i \leq j$ such that

$$x_i^{(k_\ell)} \to \hat{x}_i \quad \text{as } \ell \to \infty \tag{2.5}$$

and we define the matrix $\hat{X} := \begin{pmatrix} \hat{x}_1 & \cdots & \hat{x}_j \end{pmatrix}$. Similarly, each of the sequences $\left\{\theta_i^{(k)}\right\}, 1 \leq i \leq j$ are convergent since they are bounded monotonic sequences, and we define

$$\hat{\lambda}_i := \lim_{k \to \infty} \theta_i^{(k)} \tag{2.6}$$

and $\hat{\Lambda} := \operatorname{diag}(\hat{\lambda}_1, \dots, \hat{\lambda}_j)$. From Lemma 2.1.1 it follows readily that

$$\hat{X}^* A \hat{X} = \hat{\Lambda} \quad \text{and} \quad \hat{X}^* B \hat{X} = I. \tag{2.7}$$

Define $\hat{r} := (A - \hat{\lambda}_j B)\hat{x}_j$ and assume that $\hat{r} \neq 0$. Otherwise, $(\hat{\lambda}_j, \hat{x}_j)$ is an eigenpair of (A, B) since $\hat{r}_j = 0$ implies that $A\hat{x}_j = \hat{\lambda}_j B\hat{x}_j$. We also, define

$$\hat{A} = \begin{pmatrix} \hat{X} & \hat{r} \end{pmatrix}^* A \begin{pmatrix} \hat{X} & \hat{r} \end{pmatrix} = \begin{pmatrix} \hat{\Lambda} & \hat{X}^* A \hat{r} \\ \hat{r}^* A \hat{X} & \hat{r}^* A \hat{r} \end{pmatrix} \tag{2.8}$$

and

$$\hat{B} = \begin{pmatrix} \hat{X} & \hat{r} \end{pmatrix}^* B \begin{pmatrix} \hat{X} & \hat{r} \end{pmatrix} = \begin{pmatrix} I & \hat{X}^* B \hat{r} \\ \hat{r}^* B \hat{X} & \hat{r}^* B \hat{r} \end{pmatrix}. \tag{2.9}$$

Without loss of generality, let us assume that $\hat{\lambda}_1 < \cdots < \hat{\lambda}_j$. Otherwise, we may consider only the unique instances of $\hat{\lambda}_i$ for $1 \leq i \leq p$. Under this assumption, Sturm sequencing can be used to show that the matrix

$$\hat{A} - \hat{\lambda}_j \hat{B} = \begin{pmatrix} \hat{\Lambda} - \hat{\lambda}_j I & f \\ f^* & \hat{r}^*(A - \hat{\lambda}_j B)\hat{r} \end{pmatrix}$$

$$= \begin{pmatrix} \hat{\lambda}_1 - \hat{\lambda}_j & & & & * \\ & \ddots & & & \vdots \\ & & \hat{\lambda}_{j-1} - \hat{\lambda}_j & & * \\ & & & 0 & \hat{r}^*\hat{r} \\ * & \cdots & * & \hat{r}^*\hat{r} & \hat{r}^*(A - \hat{\lambda}_j B)\hat{r} \end{pmatrix} \tag{2.10}$$

has j negative eigenvalues. Denoting the j^{th} eigenvalue of $(\hat{A}, \hat{B})$ by $\hat{\theta}_j$, it therefore follows that $\hat{\theta}_j < \hat{\lambda}_j$.

On the other hand, for each k, we define $r_j^{(k)} := (A - \theta_j^{(k)} B) x_j^{(k)}$ and

$$\hat{A}^{(k)} := \begin{pmatrix} X^{(k)} & r_j^{(k)} \end{pmatrix}^* A \begin{pmatrix} X^{(k)} & r_j^{(k)} \end{pmatrix} \quad \text{and} \quad \hat{B}^{(k)} := \begin{pmatrix} X^{(k)} & r_j^{(k)} \end{pmatrix}^* B \begin{pmatrix} X^{(k)} & r_j^{(k)} \end{pmatrix}.$$

We note readily that $(\hat{A}^{(k_\ell)}, \hat{B}^{(k_\ell)}) \to (\hat{A}, \hat{B})$ as $\ell \to \infty$ and therefore letting $\hat{\theta}_j^{(k_\ell+1)}$ denote the j^{th} eigenvalue of $(\hat{A}^{(k)}, \hat{B}^{(k)})$, we see that $\hat{\theta}_j^{(k_\ell+1)} \to \hat{\theta}_j$ by continuity of eigenvalues. As the subspace $\mathcal{S}^{(k)}$ contains at least the vectors $\left\{ x_1^{(k)}, \ldots, x_j^{(k)}, r_j^{(k)} \right\}$, we note that $\theta_j^{(k_\ell+1)} \leq \hat{\theta}_j^{(k_\ell+1)}$ and therefore

$$\hat{\lambda}_j = \lim_{\ell \to \infty} \theta_j^{(k_\ell+1)} \leq \lim_{\ell \to \infty} \hat{\theta}_j^{(k_\ell+1)} = \hat{\theta}_j < \hat{\lambda}_j \tag{2.11}$$

which is a contradiction. It follows that $\hat{r}_j = 0$ and therefore $(\hat{\lambda}_j, \hat{x}_j)$ is an eigenpair of (A, B).

The remainder of the proof may be carried out in exactly the same way as the proof of Proposition 3.1 in [25] in order to show that the Ritz vectors converge in direction to an eigenvector. □

A simple algorithm of the form of our generic eigensolver is the method of steepest descent. At each outer iteration k, the constructed subspace $\mathcal{S}^{(k)}$ is the two dimensional subspace spanned by the Ritz vector $x_1^{(k)}$ and the residual $r_1^{(k)} = (A - \theta_1^{(k)} B) x_1^{(k)}$. In the language of traditional steepest descent, the residual vector may be referred to as the *search direction*. The method of steepest descent is convergent as each of Lemma 2.1.1 and Lemma 2.1.2 apply, however, convergence is usually not rapid in the least. An algorithm of Knyazev called LOBPCG [34] may be seen as an extension of steepest descent. At each outer iteration k of LOBPCG, the subspace $\mathcal{S}^{(k)}$ is the span of the current Ritz vector, the current residual, and the *previous* Ritz vector. This algorithm enjoys considerably faster convergence than the method of steepest descent. For more details concerning this method, see [34].

2.2 The Inverse Free method of Golub and Ye

In their 2002 paper [25], Golub and Ye introduced and analyzed an inverse free preconditioned Krylov subspace algorithm for the symmetric generalized eigenvalue problem. The principle on which this algorithm relies is the same as that of the method of steepest descent. The fundamental difference here, however, is in the choice of the search direction. Instead of simply choosing the residual r, the method of [25] seeks the search direction in the Krylov subspace $\mathcal{K}_m(A - \rho B, r)$, where $\rho = \rho(x)$. The next iterate is therefore selected from the subspace

$$K_m(A - \rho B, x) := \operatorname{span}\{x\} + \mathcal{K}_m(A - \rho B, r)$$

The use of the Krylov subspace $\mathcal{K}_m(A - \rho B, r)$ is further motivated by inexact inverse iteration as in [24]. The basic procedure is given in Algorithm 2.2. In the language

Algorithm 2.2 Inverse Free Krylov Subspace Algorithm for (A, B)

```
Input: Symmetric $A$, s.p.d. $B$, $x^{(1)}$ with $\|x^{(1)}\| = 1$, and $m \geq 1$.
1: $\rho^{(1)} = \rho(x^{(1)}; A, B)$
2: for $k = 1, 2, \ldots$ do
3:     Construct a basis $Z_m$ of $K_m(A - \rho^{(k)}B, x^{(k)})$
4:     Form projection $A_m = Z_m^*(A - \rho^{(k)}B)Z_m$, $B_m = Z_m^* B Z_m$
5:     Compute smallest eigenpair $(\theta, u)$ of $(A_m, B_m)$
6:     $\rho^{(k+1)} = \rho^{(k)} + \theta$; $x^{(k+1)} = Z_m u$
7: end for
```

of our generic eigensolver, the constructed subspace at step k of the iteration is $\mathcal{S}^{(k)} = K_m(A - \rho^{(k)}B, x^{(k)})$ and lines four through six perform the Rayleigh-Ritz procedure. It should also be noted that given an iterate $x^{(k)}$, the next iterate $x^{(k+1)}$ is defined by

$$x^{(k+1)} = p_k(A - \rho^{(k)}B)x^{(k)}$$

where $p_k(z)$ is some polynomial with degree not exceeding m. Therefore

$$x^{(k+1)} = \left(\prod_{j=1}^{k} p_j(A - \rho^{(j)}B) \right) x^{(1)} = p(A, B)x^{(1)}$$

where $p(z, w)$ is a bivariate polynomial of degree at most mk, showing this method to be a preconditioned eigensolver by the definition of Knyazev [32, 33].

The construction of the basis of the Krylov subspace is accomplished using either the Lanczos method or the Arnoldi method with the B inner product. With the choice of the Lanczos method, the projection $Z_m^*(A - \rho^{(k)}B)Z_m$ is freely available as the tridiagonal matrix of recurrence coefficients while $Z_m^* B Z_m$ must be explicitly formed. If m is sufficiently large, however, orthogonality will be gradually lost forcing either the need for reorthogonalization or explicit formation of A_m as well. Similarly, with the choice of B-orthonormal Arnoldi, we must explicitly form A_m while $B_m = I$. This comes at the price of a longer recurrence and the need for reorthogonalization or explicit formation of B_m with larger values of m. See [25] for a more detailed discussion concerning these issues.

The sequence of iterates, $\{\rho^{(k)}\}$ generated by Algorithm 2.2 converges to some eigenvalue λ of the pencil (A, B) while the sequence $\{x^{(k)}\}$ converges in direction to an eigenvector of the pencil corresponding to λ (see Lemma 2.1.2). Furthermore, an asymptotic convergence rate for the sequence $\{\rho^{(k)}\}$ is established in [25]; this result is recalled in the Theorem 2.2.1.

Theorem 2.2.1 (Theorem 3.4 of [25]). *Let $\lambda_1 < \lambda_2 \leq \cdots \leq \lambda_n$ be the eigenvalues of (A, B) and $(\rho^{(k+1)}, x^{(k+1)})$ the approximate eigenpair obtained by Algorithm 2.2 from $(\rho^{(k)}, x^{(k)})$. Let $\sigma_1 < \sigma_2 \leq \cdots \leq \sigma_n$ be the eigenvalues of $A - \rho^{(k)}B$ and u_1 be a unit eigenvector corresponding to σ_1. Assume that $\lambda_1 < \rho^{(k)} < \lambda_2$. Then*

$$\rho^{(k+1)} - \lambda_1 \leq (\rho^{(k)} - \lambda_1)\epsilon_m^2 + 2(\rho^{(k)} - \lambda_1)^{3/2}\epsilon_m \left(\frac{\|B\|}{\sigma_2}\right)^{1/2} + \delta_k \tag{2.12}$$

where

$$0 \leq \delta_k \equiv \rho^{(k)} - \lambda_1 + \frac{\sigma_1}{u_1^* B u_1} = \mathcal{O}((\rho^{(k)} - \lambda_1)^2)$$

and

$$\epsilon_m = \min_{p \in \mathcal{P}_m, p(\sigma_1)=1} \max_{i \neq 1} |p(\sigma_i)|$$

with $\mathcal{P}_m$ denoting the set of all polynomials of degree not greater than m.

The quantity ϵ_m^2 bounds the rate of convergence here and it is well-known that it may be bounded by

$$\epsilon_m \leq 2\left(\frac{1-\sqrt{\psi}}{1+\sqrt{\psi}}\right)^m, \quad \psi := \frac{\sigma_2 - \sigma_1}{\sigma_n - \sigma_1}$$

Thus, any scheme which increases the value of ψ will accelerate the convergence of Algorithm 2.2. That is, any transformation which sufficiently separates σ_1 from the rest of the spectrum (of $A - \rho^{(k)}B$) will correspondingly accelerate the convergence of the sequence $\{\rho^{(k)}\}$ to λ_1. We will refer to schemes aimed at transforming the problem to make the quantity ψ larger as preconditioning schemes, and these are the topic of Chapter 3.

2.3 Block Inverse Free Algorithms

Block methods taking the form of the generic eigensolver described in Section 2.1 include LOBPCG [34], a block variant of DACG [2],[19], and Jacobi-Davidson [17] among others. The recent report by Arbenz et al. gives an overview of such algorithms [2] with applications to structural dynamics. We seek here to develop a block variant of Algorithm 2.2.

2.3.1 Block variant I

In order to successfully compute p eigenvalues of the pencil (A, B) using an algorithm based on the generic process, it is clear from Lemma 2.1.1 and Lemma 2.1.2 that inclusion of Ritz vectors and the associated residuals in the search space give a sufficient condition for convergence. Each of the algorithms mentioned above include the residuals in the search space. Thus, when considering a method to compute the p lowest eigenvalues based on Algorithm 2.2, we first examine a method which uses the subspace $\mathcal{S}^{(k)} = \mathcal{K}$ with

$$\mathcal{K} := \sum_{i=1}^{p} K_m(A - \theta_i^{(k)} B, x_i^{(k)}). \tag{2.13}$$

That is, any vector $u \in \mathcal{K}$ is given by $u = \sum_{i=1}^{p} u_i$ where $u_i \in K_m(A - \theta_i^{(k)} B, x_i^{(k)})$. One may construct a basis for this subspace by first constructing p bases, say Z_i for $1 \leq i \leq p$ of the p Krylov subspaces $K_m(A - \theta_i^{(k)} B, x_i^{(k)}), 1 \leq i \leq p$, and then performing a global orthogonalization to obtain Z, an orthogonal basis of $\mathcal{K}$. We remark that although the basis formed by the columns of Z may be orthogonal with respect to an arbitrary inner product, the "correct" inner product to work with here is the B inner product, as the Ritz vectors are automatically B-orthogonal.The algorithm is detailed in Algorithm 2.3.

Algorithm 2.3 Block variant I: direct sum of Krylov spaces

```
Input: Symmetric $A$, s.p.d. $B$, $X^{(1)} \in \mathbb{R}^{n \times p}$ with $X^{(1)*} B X^{(1)} = I_p$, and $m \geq 1$.
 1: $\Theta^{(1)} = \mathrm{diag}(X^{(1)*} A X^{(1)})$
 2: for $k = 1, 2, \ldots$ do
 3:     for $i = 1, \ldots, p$ do
 4:         Construct a basis $\hat{Z}_i$ of $K_m(A - \theta_i^{(k)} B, x_i^{(k)})$
 5:     end for
 6:     Orthonormalize $\begin{pmatrix} \hat{Z}_1 & \cdots & \hat{Z}_p \end{pmatrix}$ to obtain $Z$
 7:     Form projection $A_m = Z^* A Z$, $B_m = Z^* B Z$
 8:     Compute $p$ smallest eigenpairs $(\theta_i, u_i), 1 \leq i \leq p$ of $(A_m, B_m)$
 9:     $\Theta^{(k+1)} = \mathrm{diag}(\theta_1, \ldots, \theta_p)$; $X^{(k+1)} = ZU$, $U = \begin{pmatrix} u_1 & \cdots & u_p \end{pmatrix}$
10: end for
```

Orthonormality of the matrix Z, though not mandated by the Rayleigh-Ritz process, is desirable as it contributes stability to the computation of the eigenpairs of the projected problem. Moreover, the two blocks may together be rank deficient. That is, although $\hat{Z}_1$ and $\hat{Z}_2$ are orthonormal matrices by themselves, the larger matrix $\begin{pmatrix}\hat{Z}_1 & \hat{Z}_2\end{pmatrix}$ may fail to have full column rank. The detection (and subsequent deletion) of linearly dependent collections of vectors from $\hat{Z} = \begin{pmatrix}\hat{Z}_1 & \cdots & \hat{Z}_p\end{pmatrix}$ will be a desirable byproduct of the global orthogonalization step.

The desire for global orthogonality here leads to difficulty in forming the projected pencil (Z^*AZ, Z^*BZ). This is a distinct weakness of this approach, despite the inherently parallel process expressed by the loop in lines three through five. In the case that $B = I$, however, the Krylov space $K_m(A - \theta_2 B, x_2) = K_m(A - \theta_1 B, x_2)$ for any choices of θ_1 and θ_2, and the algorithm reduces to the usual Block Lanczos algorithm. The need for performing the global orthogonalization therefore evaporates; the method takes care of the orthogonalities. By examining carefully the construction of the basis Z, we may arrive at a process which obviates the need for the global orthogonalization in line six of Algorithm 2.3 while still results in an orthogonal basis, even when $B \neq I$. Also, by storing some auxiliary vectors throughout the computation, construction of the projected problem becomes easier. Of course, we trade storage for floating point operations.

2.3.2 Revealing an Arnoldi-like structure

Using the Arnoldi process to generate bases of $K_m(A - \theta_i^{(k)} B, x_i^{(k)})$ in line four of Algorithm 2.3 gives the relations

$$(A - \theta_i B)\hat{Z}_i = \hat{Z}_i \hat{H}_i + w_i e_m^* \quad 1 \leq i \leq p \tag{2.14}$$

where $\hat{H}_i$ is upper Hessenberg and $w_i e_m^*$ is the rank one error term generated by the Arnoldi process such that $\hat{Z}_i^* w_i = 0$. We have suppressed the superscripts on θ_i for readability. Collecting all of the relations (2.14) we obtain

$$A\hat{Z} = \hat{Z}\hat{H} + \hat{W} + B\hat{Z}(\Theta \otimes I_{m+1}) \tag{2.15}$$

where

$$\hat{H} = \operatorname{diag}(\hat{H}_1, \cdots, \hat{H}_p) \quad \text{and} \quad \hat{W} = \begin{pmatrix} w_1 e_m^* & w_2 e_m^* & \cdots & w_p e_m^* \end{pmatrix}$$

and we recall that given two matrices $A = [a_{ij}]$ and B, the Kronecker product of A and B is

$$A \otimes B := \begin{pmatrix} a_{11}B & a_{12}B & \cdots & a_{1n}B \\ a_{21}B & a_{22}B & \cdots & a_{2n}B \\ \vdots & \vdots & & \vdots \\ a_{m1}B & a_{m2}B & \cdots & a_{mn}B \end{pmatrix}.$$

Equation (2.15) is similar to the usual block Krylov relation, however with one striking difference. The usual error term for a block Krylov process is an $n \times (mp)$ matrix whose first $(m-1)p$ columns are zero. Our error term, $\hat{W}$, consists of p blocks of dimension $n \times m$, each of which has the m^{th} column as the only nonzero column. To give the error term in relation (2.15) a structure similar to that of the usual block Arnoldi process, we multiply by an appropriately chosen permutation matrix. Indeed, let $\{e_1, \ldots, e_m\}$ be the canonical basis of $\mathbb{R}^m$ and let $\{\epsilon_1, \ldots, \epsilon_p\}$ be the canonical basis of $\mathbb{R}^p$. Define

$$P := \begin{pmatrix} e_1\epsilon_1^* & e_2\epsilon_1^* & \cdots & e_m\epsilon_1^* \\ e_1\epsilon_2^* & e_2\epsilon_2^* & \cdots & e_m\epsilon_2^* \\ \vdots & \vdots & & \vdots \\ e_1\epsilon_p^* & e_2\epsilon_p^* & \cdots & e_m\epsilon_p^* \end{pmatrix} \tag{2.16}$$

and note that P is a $p \times m$ block permutation matrix with blocks of dimension $m \times p$. As $\hat{W}$ is a $1 \times p$ block matrix with blocks of dimension $n \times m$, we may form the product $\hat{W}P$ to obtain

$$\begin{aligned} \hat{W}P &= w_1 e_m^* \begin{pmatrix} e_1\epsilon_1^* & e_2\epsilon_1^* & \cdots & e_m\epsilon_1^* \end{pmatrix} + \cdots + w_p e_m^* \begin{pmatrix} e_1\epsilon_p^* & e_2\epsilon_p^* & \cdots & e_m\epsilon_p^* \end{pmatrix} \\ &= \begin{pmatrix} 0 & 0 & \cdots & w_1\epsilon_1^* \end{pmatrix} + \cdots + \begin{pmatrix} 0 & 0 & \cdots & w_p\epsilon_p^* \end{pmatrix} = WE_m^* \end{aligned} \tag{2.17}$$

where $W = \begin{pmatrix} w_1 & \cdots & w_p \end{pmatrix}$ and E_m is the $m \times 1$ block matrix of $p \times p$ blocks whose m^{th} block is I_p, the identity of order p.

Post-multiplying (2.15) by P, noting that $I_{mp} = PP^*$, and applying (2.17) we obtain

$$A(\hat{Z}P) = (\hat{Z}P)(P^*\hat{H}P) + WE_m^* + B(\hat{Z}P)P^*(\Theta \otimes I_m)P. \tag{2.18}$$

Investigating further, we find that

$$P^*(\Theta \otimes I_m)P = \begin{pmatrix} \epsilon_1 e_1^* & \epsilon_2 e_1^* & \cdots & \epsilon_p e_1^* \\ \epsilon_1 e_2^* & \epsilon_2 e_2^* & \cdots & \epsilon_p e_2^* \\ \vdots & \vdots & & \vdots \\ \epsilon_1 e_m^* & \epsilon_2 e_m^* & \cdots & \epsilon_p e_m^* \end{pmatrix} \begin{pmatrix} \theta_1 e_1\epsilon_1^* & \theta_1 e_2\epsilon_1^* & \cdots & \theta_1 e_m\epsilon_1^* \\ \theta_2 e_1\epsilon_2^* & \theta_2 e_2\epsilon_2^* & \cdots & \theta_2 e_m\epsilon_2^* \\ \vdots & \vdots & & \vdots \\ \theta_p e_1\epsilon_p^* & \theta_p e_2\epsilon_p^* & \cdots & \theta_p e_m\epsilon_p^* \end{pmatrix}. \tag{2.19}$$

It is easy to see that each off-diagonal block of the product in (2.19) is zero and each diagonal block is simply

$$\theta_1\epsilon_1\epsilon_1^* + \theta_2\epsilon_2\epsilon_2^* + \cdots + \theta_p\epsilon_p\epsilon_p^* = \Theta. \tag{2.20}$$

Therefore $P^*(\Theta \otimes I_m)P = (I_m \otimes \Theta)$.

Considering the matrix $P^*\hat{H}P$, we note that the ij block of $P^*\hat{H}P$ is given by

$$\begin{pmatrix} \epsilon_1 e_i^* & \epsilon_2 e_i^* & \cdots & \epsilon_p e_i^* \end{pmatrix} \begin{pmatrix} \hat{H}_1 & & & \\ & \hat{H}_2 & & \\ & & \ddots & \\ & & & \hat{H}_p \end{pmatrix} \begin{pmatrix} e_j\epsilon_1^* \\ e_j\epsilon_2^* \\ \vdots \\ e_j\epsilon_p^* \end{pmatrix} = \sum_{k=1}^{p} (e_i^*\hat{H}_k e_j)\epsilon_k\epsilon_k^*. \tag{2.21}$$

The ij block of $P^*\hat{H}P$ is therefore a diagonal matrix whose entries are the ij entries of each matrix $\hat{H}_k, 1 \le k \le p$. Thus, $P^*\hat{H}P$ is block upper Hessenberg with a lower bandwidth of p. Revisiting (2.15), we now write

$$A\tilde{Z} = \tilde{Z}\tilde{H} + WE_m^* + B\tilde{Z}(I_m \otimes \Theta) \tag{2.22}$$

where $\tilde{Z} = \hat{Z}P = \begin{pmatrix} \tilde{Z}_1 & \cdots & \tilde{Z}_m \end{pmatrix}$ and $\tilde{H} = P^*\hat{H}P$ and is, again, block upper Hessenberg. The relation (2.22) is even more like that obtained from the Arnoldi process. Furthermore, we note that by construction, $\tilde{Z}_i^*\tilde{Z}_j$ has all zeros for its diagonal entries when $i \ne j$ and $\tilde{Z}_1^*B\tilde{Z}_1 = I$, but this is still far from the desired global orthogonality.

Following the proposal of Algorithm 2.3, we may choose to explicitly orthogonalize the columns of $\hat{Z}$, or rather, the columns of $\tilde{Z}$. Orthogonalizing the columns of $\tilde{Z}$ yields

$$\tilde{Z} = \begin{pmatrix} \tilde{Z}_1 & \tilde{Z}_2 & \cdots & \tilde{Z}_m \end{pmatrix} = \begin{pmatrix} Z_1 & Z_2 \cdots & Z_m \end{pmatrix} \begin{pmatrix} R_{11} & R_{12} & \cdots & R_{1m} \\ 0 & R_{22} & \cdots & R_{2m} \\ & \ddots & \ddots & \vdots \\ & & 0 & R_{mm} \end{pmatrix} = ZR \tag{2.23}$$

where R is an upper triangular matrix. We may then substitute $\tilde{Z} = ZR$ into (2.22) to obtain

$$AZR = ZR\tilde{H} + WE_m^* + BZR(I_m \otimes \Theta) \tag{2.24}$$

and therefore

$$AZ = Z(R\tilde{H}R^{-1}) + W(E_m^*R^{-1}) + BZ(R(I_m \otimes \Theta)R^{-1}). \tag{2.25}$$

We note that the matrix $R\tilde{H}R^{-1}$ maintains its block Hessenberg and its banded structure since R is upper triangular. Also, $E_m^*R^{-1} = R_{mm}^{-1}E_m^*$, so our error term becomes $WR_{mm}^{-1}E_m^*$. The matrix $R(I_m \otimes \Theta)R^{-1}$ now becomes an upper triangular matrix as opposed to a simple diagonal one. Thus, although we have obtained a block Krylov-like relation, it comes at the cost of a global orthogonalization step. Although in theory this global orthogonalization may not be strictly necessary, without it, the basis Z may become ill-conditioned and thereby cause the algorithm to perform poorly.

2.3.3 A Block Arnoldi-like Process and Block variant II

Investigating relation (2.22) on a block-by-block basis reveals that

$$A\tilde{Z}_j = \sum_{k=1}^{j+1} \tilde{Z}_k\tilde{H}_{kj} + B\tilde{Z}_j\Theta \tag{2.26}$$

for each $j : 1 \leq j \leq m-1$, and

$$A\tilde{Z}_m = \sum_{k=1}^{m} \tilde{Z}_k \tilde{H}_{kj} + W + B\tilde{Z}_m \Theta. \tag{2.27}$$

This is quite reminiscent of the Arnoldi recurrence (1.8), and motivates us to redefine the block entries in the block upper Hessenberg matrix to impose orthogonality on the blocks of $\tilde{Z}$. Indeed, (2.26) and (2.27) suggest that we build a basis for a block Krylov-like subspace by means of an Arnoldi-like process performed with respect to the linear operator $\mathrm{op} : \mathbb{R}^{n\times p} \to \mathbb{R}^{n\times p}$ defined by $\mathrm{op}(X) := AX - BX\Theta$. To this end, suppose that the inner product we wish to work with is defined by the symmetric positive definite matrix M. Given an arbitrary block Z_1 such that $Z_1^* M Z_1 = I$, we develop

$$W = AZ_1 - BZ_1\Theta - Z_1 H_{11} \tag{2.28}$$

with $H_{11} = Z_1^* M(AZ_1 - BZ_1\Theta)$. We then define Z_2 and H_{21} by $W = Z_2 H_{21}$, the QR factorization of W with respect to the M inner product. We may then continue to develop $Z_3, \ldots, Z_m$ in a similar Arnoldi-like manner. Indeed, applying Arnoldi to $\mathrm{op}(\cdot)$ to build a block variant of Algorithm 2.2 seems to be a natural block extension of the algorithm, from merely a notational standpoint. The algorithm is detailed in Algorithm 2.4. By construction, we obtain an orthonormal matrix Z and a block

Algorithm 2.4 Arnoldi-like process for the operator $\mathrm{op}(X) := AX - BX\Theta$

```
Input: $A, B \in \mathbb{R}^{n\times n}, \Theta \in \mathbb{R}^{p\times p}$, s.p.d. $M$, $Z_1 \in \mathbb{R}^{n\times p}$, with $Z_1^* M Z_1 = I_p$, and $m \geq 1$.
1: for $j = 1, \ldots, m$ do
2:    $W_j = AZ_j - BZ_j\Theta$
3:    for $i = 1, \ldots, j$ do
4:       $H_{ij} = Z_i^* M W_j$
5:       $W_j = W_j - Z_i H_{ij}$
6:    end for
7:    Compute $W_j = Z_{j+1} H_{j+1,j}$, the $QR$ factorization of $W$ with respect $\langle \cdot, \cdot \rangle_M$.
8: end for
```

upper Hessenberg H. Furthermore, we have the Arnoldi-like relation

$$AZ = ZH + W_m E_m^* + BZ(I_m \otimes \Theta) \tag{2.29}$$

where $Z^* M W_m = 0$. It is interesting to note that Algorithm 2.4 does not take advantage of any of the possible symmetries of A, B, or Θ. Indeed, one would hope that in our case, where A and B are both symmetric, and Θ is diagonal, we could then reduce the Arnoldi-like algorithm to a Lanczos-like algorithm, however, the presence of Θ does not allow this. If Θ is a multiple of the identity, then of course we have such a reduction, but in general this will not be the case. Also, if $B = I$, the algorithm reduces to the usual block Lanczos, and we may use a short-term recurrence.

Just as with the usual block Arnoldi/Lanczos processes, for some j, the matrix W_j generated by Algorithm 2.4 may not have full column rank, or may have *nearly* linearly dependent columns. Indeed, this situation should be revealed when attempting to compute the QR factorization of W_j, and the offending columns should be deleted altogether, or replaced. We note that for the usual block Lanczos, Golub and Underwood [22] recommend replacing the dependent columns while Cullum and Willoughby [14] advocate dropping the offenders. In either case, this foists some difficulty onto the generation of the blocks Z_{j+1} and $H_{j+1,j}$. For this reason, Ruhe [47] and Freund [18] suggest altering the block algorithm to make this situation somewhat easier to handle. This modification is known as band Lanczos, and it reduces a matrix to a banded form as opposed to a block upper Hessenberg form. Band Lanczos has the advantage of not requiring a QR factorization subprogram, however, we take, in some measure, a performance hit by moving to level 2 BLAS from level 3. That is, we no longer form blocks AZ_i, but rather we must form single vectors Az_i.

A band Arnoldi-like process corresponding to Algorithm 2.4 may be derived from a straightforward adaptation of the algorithm found in [18] and has only one little bit of extra bookkeeping, namely, that associated with which operator to use in generating a candidate Arnoldi vector. With a block size of p, the process demands storage of p candidate vectors. At each step j, a candidate vector is either added to the basis or dropped from consideration altogether. When the candidate is added to the basis, another candidate is added so that there are still p candidates. When the candidate is dropped, no new candidate is considered resulting in an effective reduction of block size. Perhaps the best way to describe the method is through an example.

Suppose we have a block size $p = 2$ and two initial vectors z_1 and z_2. These are our initial candidate vectors, and as candidate vectors, they should wear hats. The algorithm is stepped through in the following manner:

1. Determine whether or not to accept $\hat{z}_1$ as an Arnoldi vector. As it was one of our initial vectors, it is accepted as z_1, the first Arnoldi vector. We then orthogonalize the remaining candidate vectors against the newly accepted Arnoldi vector z_1. Next, as we accepted a new Arnoldi vector, we construct a new candidate vector $\hat{z}_3$ and store it where $\hat{z}_1$ was stored. The rule for forming $\hat{z}_3$ is

$$\hat{z}_3 = Az_1 - Bz_1\theta_1.$$

   We then orthogonalize the new candidate vector against all of the accepted Arnoldi vectors. In this case, the only vector to orthogonalize against is z_1.

2. We now accept $\hat{z}_2$ as the next Arnoldi vector z_2 as it was one of our initial vectors. The remaining candidate vectors (only $\hat{z}_3$ in this case) are orthogonalized against

z_2. A new candidate vector $\hat{z}_4$ is constructed as

$$\hat{z}_4 = Az_2 - Bz_2\theta_2$$

and stored where $\hat{z}_2$ was stored. The new candidate is now orthogonalized against all accepted Arnoldi vectors.

3. We now consider whether or not to accept the candidate $\hat{z}_3$, and let's suppose that this candidate is unacceptable as $\|\hat{z}_3\| < \tau$, where τ is a deflation tolerance. We discard this vector, and moreover, we never again use θ_1 to build a candidate vector. More accurately, we never again use $A - \theta_1 B$ as the operator used in constructing a candidate vector. p is now effectively one, and the process proceeds as the usual single vector Arnoldi for the operator $A - \theta_2 B$.

4. Suppose now, that the next candidate $\hat{z}_4$, is accepted as a new Arnoldi vector z_3. There are now no candidates to orthogonalize against, and a new candidate $\hat{z}_5$ is constructed as

$$\hat{z}_5 = Az_3 - Bz_3\theta_2$$

and subsequently orthogonalized against all accepted Arnoldi vectors.

5. The process continues as usual single vector Arnoldi until either we exhaust the allotted storage or we need another deflation.

In general with a block size p, p exact deflations implies that the eigenvalues of the projected problem are eigenvalues of the original problem. With a sufficiently tight deflation tolerance, this may be used as a stopping criterion for the algorithm. We also point out that the band Arnoldi variant described here implicitly assumes that Θ is a diagonal matrix, whereas no such assumption was made in the development of Algorithm 2.4. A detailed description of the band Arnoldi-like process is given in Algorithm 2.5.

With the block or band Arnoldi-like process, we may construct a block variant of Algorithm 2.2 which differs from block variant I only in the choice of subspace at each step. Indeed, we note that by examining the relation (2.29) on a vector-by-vector basis, it is easy to see that the subspace spanned by the columns of Z as constructed by Algorithm 2.4 differs from the subspace $\mathcal{K}$. These subspaces, though different, yield algorithms which have similar convergence properties, as we shall see in the examples that follow. This is not too surprising in light of the discussion motivating the block Arnoldi-like process. We note that the use of either the block or band Arnoldi-like process allows us to compute and store the products AZ and BZ throughout the development of the basis. Formation of the projection may be obtained at the cost of applying Z^* to each of these stored blocks for a total of $2(mp)^2(2n-1)$ floating point operations. For completeness, block variant II is detailed in Algorithm 2.6.

Algorithm 2.5 Band Arnoldi-like process

```
Input: $A, B \in \mathbb{R}^{n\times n}, \{\hat{z}_1, \dots, \hat{z}_p\}$, and diagonal $\Theta \in \mathbb{R}^{p\times p}$.
 1: Set $d = 0$. /* $d$ counts the number of deflations. */
 2: Set $k = 1$. /* $k$ indicates which $\theta$ to use. */
 3: for $j = 1, 2, \dots$ do
 4:   if ($\hat{z}_j$ should be included in the basis) then
 5:     Normalize $\hat{z}_j$ to obtain the Arnoldi vector $z_{j-d}$.
 6:     Orthogonalize the $p-1$ remaining candidates against $z_{j-d}$.
 7:   else
 8:     Relabel $\theta_i = \theta_{i+1}$ for $i : k \le i \le p-1$.
 9:     Set $p = p-1, d = d+1$.
10:     if ($p == 0$) then
11:       break
12:     else
13:       continue
14:     end if
15:   end if
16:   Construct $\hat{z}_{j+p} = Az_{j-d} - \theta_k Bz_{j-d}$
17:   Orthogonalize $\hat{z}_{j+1}$ against all accepted Arnoldi vectors.
18:   if ($k < p$) then
19:     $k = k+1$
20:   else
21:     $k = 1$
22:   end if
23: end for
```

Algorithm 2.6 Block variant II: block or band Arnoldi variant

```
Input: Symmetric $A$, s.p.d. $B$, $X^{(1)} \in \mathbb{R}^{n\times p}$ with $X^{(1)*}BX^{(1)} = I_p$, and $m \ge 1$.
 1: $\Theta^{(1)} = \mathrm{diag}(X^{(1)*}AX^{(1)})$
 2: for $k = 1, 2, \dots$ do
 3:   Apply block Arnoldi (Algorithm 2.4) or band Arnoldi (Algorithm 2.5) to de-
      velop a basis $Z$.
 4:   Complete the projection $A_m = Z^*AZ$, $B_m = Z^*BZ$
 5:   Compute $p$ smallest eigenpairs $(\theta_i, u_i), 1 \le i \le p$ of $(A_m, B_m)$
 6:   $\Theta^{(k+1)} = \mathrm{diag}(\theta_1, \dots, \theta_p)$; $X^{(k+1)} = ZU$
 7: end for
```

2.4 Further considerations

To construct a robust black-box eigensolver, we need to be able to declare when Ritz pairs are accepted as eigenpairs and purge them as necessary. Also, we should include a strategy for choosing a suitable block size when that information may not be available, as the distribution of eigenvalues will affect the rate of convergence of the algorithms. This section provides some details about extra steps taken in the development of the block algorithm.

2.4.1 Deflation of converged Eigenvectors

Suppose that we have a collection of j Ritz vectors that have converged to the eigenvectors corresponding to the j algebraically smallest eigenvalues $\lambda_1 \leq \cdots \leq \lambda_j$ of the pencil (A, B). Suppose further that we have still not obtained all of the desired eigenpairs. Iterating the outer loop in either of the block variants with the converged Ritz pairs will lead to extra computation. To avoid superfluous computation, one usually removes the converged eigenvectors via a process called deflation, and there are at least two ways to go about this; deflation schemes are discussed in [14] [46], [50], and [62].

Explicit deflation removes the j Ritz vectors from the initial block and makes use of the observation that given any $j \times j$ diagonal matrix Σ, the pencil

$$(A + BX_j\Sigma X_j^*B, B) \tag{2.30}$$

has eigenvalues $\lambda_1+\sigma_1, \lambda_2+\sigma_2, \ldots, \lambda_j+\sigma_j, \lambda_{j+1}, \ldots, \lambda_n$. Here, X_j denotes the matrix whose columns are the j eigenvectors corresponding to the eigenvalues $x_1, \ldots, x_j$. By all accounts, this idea is attributed to Hotelling, and is easy to verify by noting that the matrix X of eigenvectors of (A, B) simultaneously diagonalizes A and B. Thus if the eigenvectors are normalized such that $X^*BX = I$, then we have

$$X^*(A + BX_j\Sigma X_j^*B)X = \Lambda + \begin{pmatrix} \sigma_1 & & & & & \\ & \ddots & & & & \\ & & \sigma_j & & & \\ & & & 0 & & \\ & & & & \ddots & \\ & & & & & 0 \end{pmatrix} \tag{2.31}$$

showing that the eigenvalues are as desired. Indeed, in an implementation of Algorithm 2.2, Money and Ye [40] use this technique to successively compute eigenpairs from the left end of the spectrum. They take special care, however, to choose $\sigma_1, \ldots, \sigma_j$ such that $\lambda_{p+1} < \lambda_i + \sigma_i < \lambda_n, 1 \leq i \leq j$ so as to preserve the convergence properties of the algorithm. That is, such choices of σ_i will not necessarily degrade the convergence of each Ritz pair, as the interval which determines the rate of convergence is interfered with minimally, if at all. The block variant of the algorithm may be applied to the pencil (2.30) with a block size of $p - j$ in order to resolve the remaining $p - j$ desired eigenpairs.

Implicit deflation, on the other hand, includes the already converged Ritz vectors in successive bases, however, their "descendents" are not included. In fact, the band Arnoldi process described by Algorithm 2.5 has this deflation built in. The algorithm thus recomputes the Ritz pairs, and Cullum and Willoughby [14] point out that this has the desirable side effect of possibly refining the eigenvector approximations. This is in contrast to the approach taken in explicit deflation where Ritz vectors are locked when convergence is declared. Such refinement comes at the cost of maintaining a few extra Ritz vectors, which in an explicit deflation scheme, would have been pushed out to make more room for Arnoldi vectors.

As for determining convergence of the Ritz pairs, we use the residual norms as motivated by Theorem 1.2.9. We note that due to the restart, at the second inner iteration, we have a measure of the residual vector for free. That is, when using the block Arnoldi-like process in block variant II, the block H_{21} is just the residual. Thus, we may get the previous residual norms by taking the norms of the columns of H_{21}. We note that an implicit deflation procedure detects these converged Ritz pairs and deflates them automatically.

2.4.2 Adding the previous Ritz vector

Knyazev's LOBPCG algorithm [34] is essentially the steepest descent algorithm with the previous iterate added. In fact, Knyazev adds the vector $p^{(k)} = x^{(k)} - x^{(k-1)}$ to the steepest descent subspace, provided that it is significant. Simply adding this vector gives a significant increase in the rate of convergence. In view of this, Money and Ye [40] also add the vector $p^{(k)}$ to the subspace $K_m(A - \rho^{(k)} B, x^{(k)})$, and their experiments show a similar increase in convergence. We will add the block of difference vectors to our subspaces in block variants I and II. We point out that as of yet, there is no theory defining just how addition of this vector affects the rate of convergence of schemes in the form of our generic eigensolver, although many have observed the benefit [34], [35], [40], [44]. Moreover, it has been noted that the addition of any more previous iterates, say the past two or three for example, does not give a corresponding acceleration.

2.4.3 Adaptively choosing the block size

The block methods described above are easily modified to include a scheme for the adaptive choice of block size as in [3] and [63]. Indeed, all that is required is a test for clustering of Ritz values and then, if necessary, increase of the block size. Due to the inner-outer iteration structure of the algorithms, changing the block size does not complicate construction of the basis in any way; vectors must simply be added to the initial block. The newly added vectors should be B orthonormal to the current vectors, and the corresponding eigenvalue approximations should be chosen as the Rayleigh quotients of the new vectors.

As in the adaptive block Lanczos procedure in [63], we are only interested in possible clusters which exceed our current block size, since presence of such clusters degrade convergence of our algorithms. In order to detect such a cluster, we use a heuristic like the one appearing in Ye's paper. That is, supposing that we wish to compute the algebraically smallest eigenvalue λ_1, we would begin with a block size of at least two, and we would examine the quantity

$$\frac{\theta_2 - \theta_1}{\theta_3 - \theta_2}. \tag{2.32}$$

If the ratio (2.32) is smaller than some user-defined tolerance η that indicates that the eigenvalues approximated by those Ritz values may be clustered, and increasing the block size is justified. More generally, if the w algebraically smallest eigenvalues of (A, B) are desired, and the current block size is $p > w$, then if

$$\frac{\theta_p - \theta_w}{\theta_{p+1} - \theta_p} \leq \eta \tag{2.33}$$

is satisfied, we increase the block size by one and continue.

Should the algorithm experience misconvergence, that is, if the extreme Ritz values begin to converge to some eigenvalues but suddenly "step over" those eigenvalues on their way to converging to the truly extreme eigenvalues then the heuristic (2.33) may falsely indicate a larger cluster size than is really there when η is sufficiently large. Therefore, a limit on the block size should be imposed to regulate storage requirements. Of course, setting the block size to be the size of the largest known cluster is ideal, however, this may not always possible.

2.5 Some Implementation Details

For the purposes of constructing examples, the algorithms described above have been implemented as MATLAB programs. Portions of these codes will ultimately be woven into the piece of software from [40] and made available for public distribution. We discuss a few pertinent issues pertaining to the implementation of these programs and omit a detailed listing of the programs.

The block algorithm is realized as one driver, with subroutines corresponding to each of the three algorithms for basis construction described above. There is also a subroutine implementing the addition of the previous Ritz vectors as described in Section 2.4.2 and this subroutine is meant to be called from the routine which constructs the basis. A pseudocode listing of the implemented algorithm is given as Algorithm 2.7.

Algorithm 2.7 The Implemented Block Algorithm

```
Input: Symmetric $A$, s.p.d. $B$, initial vectors $X \in \mathbb{R}^{n\times p}$ and the parameters outlined
    in Table 2.1.
 1: Orthonormalize $X$ with respect to the $B$-inner product to obtain $X^{(1)}$.
 2: Compute initial eigenvalue approximations $\Theta^{(1)} = X^{(1)^*}AX^{(1)}$.
 3: for $j = 1 : \texttt{maxits}$ do
 4:   Build the basis $Z$ and form the projected problem $(A_m, B_m)$. The basis build
      routine should add the previous Ritz vector, if requested.
 5:   Examine the constructed residuals; these are a byproduct of the basis build
      routine. If their norms satisfy the tolerance, break;
 6:   Compute $p$ smallest eigenpairs of $(A_m, B_m)$ and form $\Theta^{(k+1)}$ and $X^{(k+1)}$.
 7:   if use adaptive block sizing then
 8:     Compute the ratio $\phi^{(k+1)} = \frac{\theta_p^{(k+1)} - \theta_w^{(k+1)}}{\theta_{p+1}^{(k+1)} - \theta_p^{(k+1)}}$
 9:     if $\phi^{(k+1)} < \eta$ then
10:       $p = p + 1$
11:       Add a column to $X^{(k+1)}$ and $B$-orthonormalize it against the other
          columns.
12:       Compute $\theta_p^{(k+1)} = x_p^{(k+1)}$.
13:     end if
14:   end if
15: end for
```

The basis development algorithms are implemented with an eye to minimizing storage and matrix-vector multiplies (matvecs). We remark, however, that the number of matrix-vector multiplies returned by the driver routine accounts for *all* applications of A or B to a *single* vector. Thus, if A is applied to an $n \times p$ block, this is counted as p matvecs. Furthermore, calls to modified Gram-Schmidt to orthogonalize a collection of p vectors with respect to the inner product defined by the matrix B requires as many as $2p-1$ matvecs with B, depending on whether or not full reorthogonalization is used. These matvecs are not counted in the total counts reported in the experiments. Rather, the number of calls to the modified Gram-Schmidt subprogram is reported.

Regarding storage, the complete basis Z is stored, as well as the matrix BZ. Here we trade storage for diminishing the number of necessary matvecs, however, with large values of m, this option may become intractable. Also, only the upper triangles of the projected matrices A_m and B_m are maintained.

Parameter	Description	Default Value
w	The number of desired eigenpairs.	1
p	The initial block size.	2
m	The number of inner iterations.	2
maxits	The maximum number of outer iterations.	300
ϵ	The residual norm tolerance.	1e$-$12
τ	The deflation tolerance.	1e$-$08
δ	The difference tolerance.	1e$-$08
adapt	Flag indicating whether or not to use adaptive block sizing.	true
η	The clustering tolerance (only used if adapt is true).	1e$-$01
max_p	The maximum block size permitted (only used if adapt is true).	12
add_previous	Flag indicating whether or not to add the previous Ritz vector to the basis.	true

Table 2.1: Parameters for BLOCKEIGIFP

This method is far from parameter-free, however, for the most part, sensible default choices may be made, especially if we allow for adaptive choice of block size. The necessary parameters are detailed in Table 2.1. Notably, we accept four tolerances among our parameters, many of which may be defaulted. The residual tolerance ϵ is used as a stopping criterion. The deflation tolerance τ is used to determine whether or not to deflate a candidate vector. The clustering tolerance η is used to determine whether or not to increase the block size, and the difference tolerance δ is used to determine whether or not the previous Ritz vector is significant enough to add to the basis. One remark about the stopping criterion is in order here. We note that the residual $Ax_i^{(k)} - \theta_i^{(k)} Bx_i^{(k)}$ is explicitly computed during the construction of the basis in step $k+1$, and thus there is no need for computation of this quantity at the end of step k. The usual deflation tolerance τ is not applicable to this vector, and in our codes, we use ϵ to determine deflation, as deflation of this vector is tantamount to declaring that the Ritz pair $(\theta_i^{(k)}, x_i^{(k)})$ has converged.

2.6 Numerical Examples I

Here we consider some examples illustrating the performance of the block algorithms described in Section 2.3 above. In this section, the algorithms are applied without preconditioning. We discuss preconditioning techniques for the algorithms and present examples illustrating the effects of various techniques in Chapter 3.

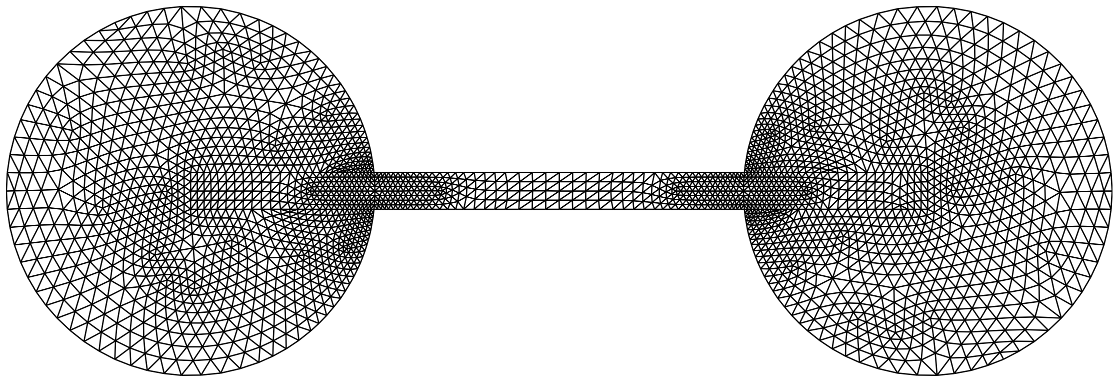


Figure 2.1: Mesh for barbell shaped domain.

2.6.1 Example: block versus single vector

The first example we consider is the Laplacian eigenvalue problem

$$\begin{aligned} -\Delta u(x) &= \lambda u(x) \quad x \in \Omega \\ u(x) &= 0 \quad x \in \partial\Omega \end{aligned} \tag{2.34}$$

with Ω being the barbell shaped domain depicted in Figure 2.1. The mesh contains 2713 nodes, of which 2441 are interior nodes. The discretized problem $Kx = \lambda Mx$ is obtained by a finite element model obtained using MATLAB's PDE toolbox. Here K and M are symmetric positive definite matrices of dimension 2441×2441; the Dirichlet boundary conditions are eliminated in the construction of these matrices. The pencil (K, M) has eigenvalues occurring in small clusters of two due to the binodal structure of the domain. In fact the two algebraically smallest eigenvalues match to five significant digits. Here we illustrate the difficulties the single vector algorithm experiences in the presence of these clusters, and the remedies afforded by the block variants. Indeed, running BlockVarII(1,4) fails to converge after 600 outer iterations. Also, increasing m here fails to give convergence; it merely accelerates the rate at which we get to stagnation. This is precisely the principle of Theorem 2.2.1; the degree of the polynomial is simply too small to cope with small gap between the lowest two eigenvalues of the iteration matrix. The residual norm histories of BlockVarII(1, m) for $m = 2, 4, 8, 16$ appears in Figure 2.2. Convergence is obtained for BlockVarII(1,90) however, this is at the extreme cost of 180 matrix-vector multiplies, and storage of up to 90 vectors which must be orthogonalized against per outer iteration! Increasing the block size to $p = 2$ gives convergence for all choices of m, with convergence accelerated as m increases. The residual norm histories for the lowest eigenvalue for BlockVarII(2,m), $m = 2, 4, 8, 16$ are presented in Figure 2.3.

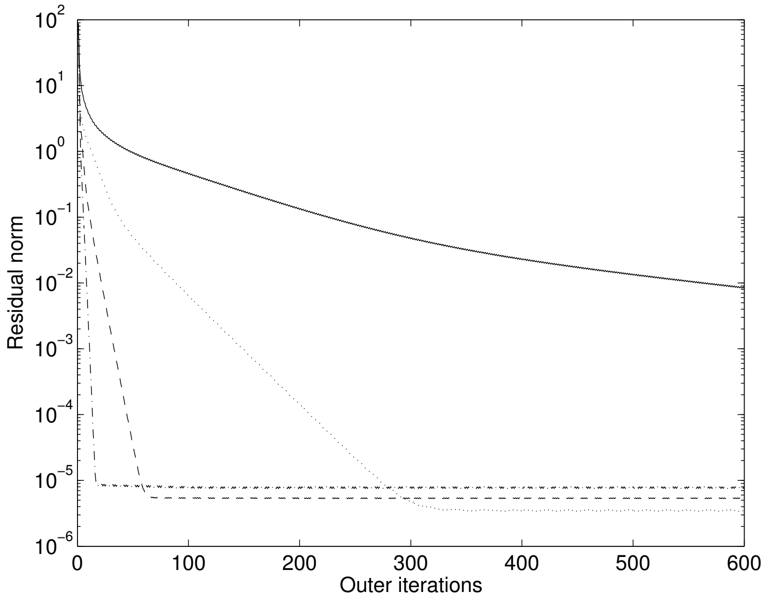


Figure 2.2: Convergence history of BlockVarII(1,m), m = 2 (solid) 4 (dotted), 8(dashed), 16(dash-dot).

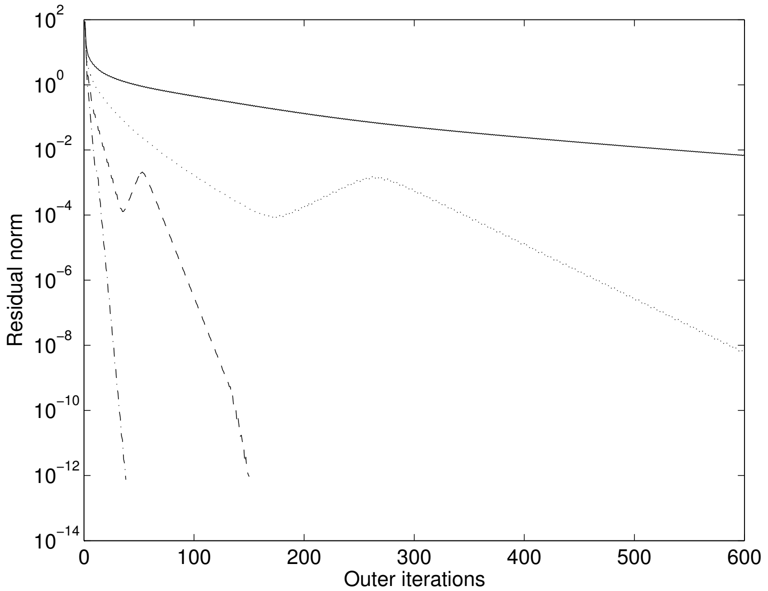


Figure 2.3: Convergence history of BlockVarII(2,m), m = 2 (solid) 4 (dotted), 8(dashed), 16(dash-dot).

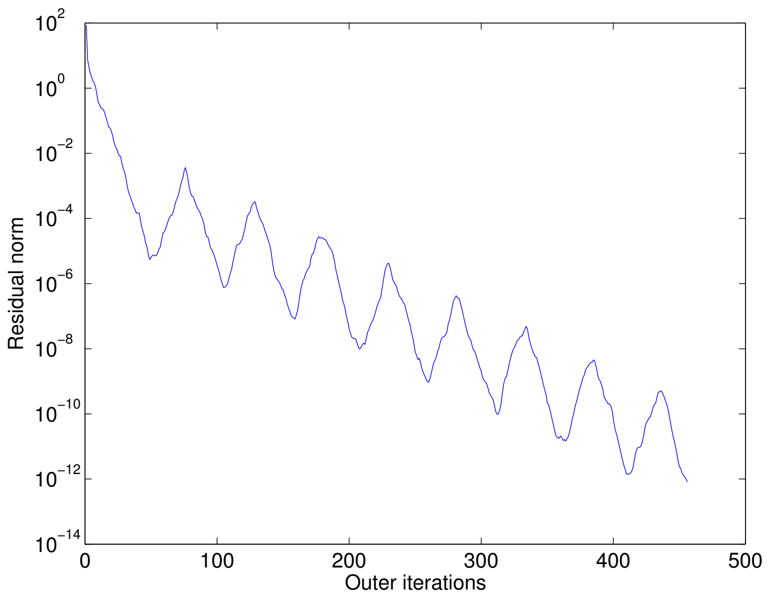


Figure 2.4: Convergence history of BlockVarIIp(1,4).

2.6.2 Example: adding the previous Ritz vector

In both cases ($p = 1$ and $p = 2$), addition of the previous Ritz vector accelerates convergence. For the single vector case, convergence is obtained, even for the modest choice $m = 4$. Convergence here, however, is irregular, owing to the presence of the cluster. Again, we point out that there is currently no known theory elucidating this behavior. The convergence history of BlockVarIIp(1,4) is presented in Figure 2.4. For the case $p = 2$, 3297 matrix-vector multiplies are required to obtain convergence, as opposed to the 5006 matrix-vector multiplies required for convergence when $p = 1$. We remark that for this price, we have two converged eigenpairs instead of just one making the cost 1649 matrix-vector multiplies per eigenpair. That is, roughly one-third of the work is required for the price of storing five extra vectors (four for the extra block size, and one for adding the previous Ritz vector). The convergence history of the two lowest eigenpairs obtained from BlockVarIIp(2,4) is presented in Figure 2.5. We note further that BlockVarII(2,4) in the first example required 9602 matrix-vector products to obtain the two lowest eigenpairs with residuals on the order of $6e - 09$. This is almost three times as many as the number of matvecs required of BlockVarIIp(2,4) and only offers roughly two-thirds of the accuracy. The negligible cost of five extra vectors is well worth the work saved.

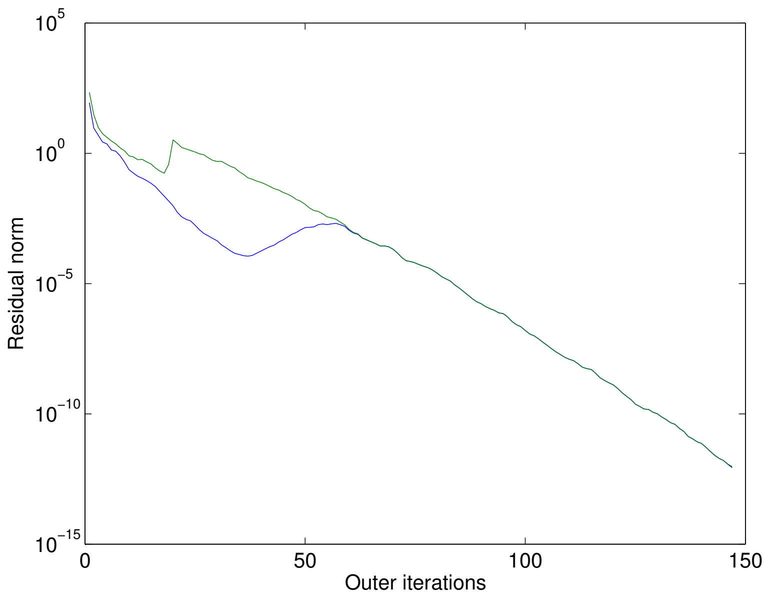


Figure 2.5: Convergence history of BlockVarIIp(2,4).

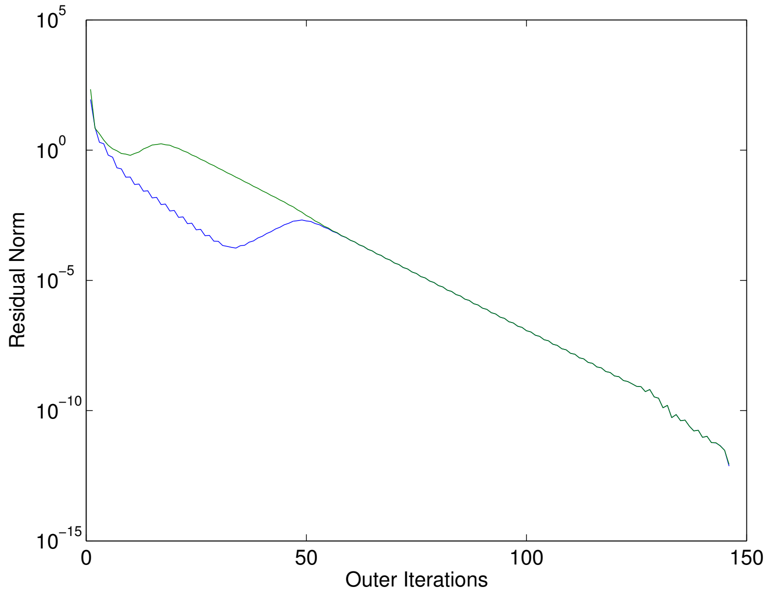
2.6.3 Example: comparing basis construction methods

The basis used in BlockVarI is different from the basis generated by the block Arnoldi-like process, however, the convergence characteristics of the two algorithms are remarkably similar. This is not too surprising in view of the discussion in Section 2.3. Figure 2.6 shows the convergence histories for BlockVarI(2,8) and BlockVarII(2,8) applied to the Laplacian eigenvalue problem of Section 2.6.1.

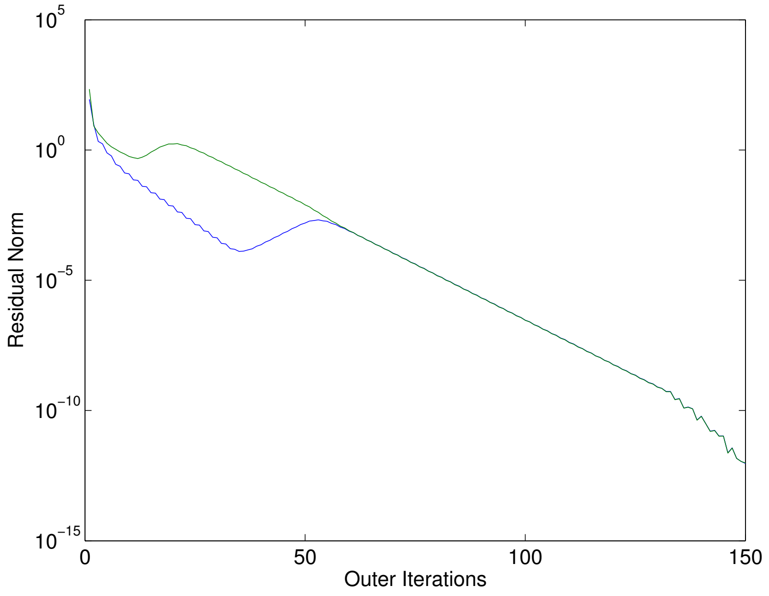
2.6.4 Example: choosing the number of inner iterations

The choice of the number of inner iterations m, certainly has an effect on the quality of the iteration. Indeed m can be any integer larger than or equal to two, and the question at hand is what is the optimal choice of the parameter? Is it always the case that larger values of m gives better convergence? In terms of outer iterations, the answer is yes. Larger values of m require fewer outer iterations to complete, but at some point, the number of matrix vector multiplications and the amount of storage required per outer iteration will become so high that it becomes infeasible to pay those costs.

We present here some data obtained by running BlockVarIIp(1, m) on a family of operators obtained from the usual uniform five-point finite difference discretization



(a) BlockVarI



(b) BlockVarII

Figure 2.6: Convergence histories of two lowest eigenvalues computed by BlockVarI(2,8) and BlockVarII(2,8).

Problem Size (k^2)	625	900	1225	1600	2025	2500	3025
$\hat{m}$	13	29	12	13	11	20	25
$\hat{m}/k$	0.520	0.967	0.343	0.325	0.244	0.400	0.500

Table 2.2: Empirically optimal number of inner iterations $\hat{m}$ for finite difference discretized Laplacian of order k^2.

of the boundary value problem

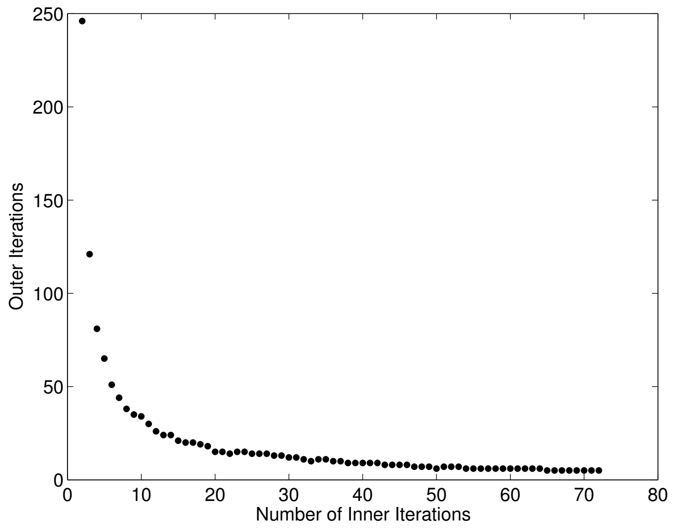
$$\begin{aligned} -\Delta u(x,y) + \rho(x,y)u(x,y) &= \lambda u(x,y) \quad (x,y) \in \Omega \\ u(x,y) &= 0 \quad (x,y) \in \partial\Omega. \end{aligned} \tag{2.35}$$

where $\Omega = [0,1] \times [0,1]$ with step sizes $\Delta x = \Delta y = \frac{1}{k+1}$. We take $\rho(x,y) = 0$ and form the discretized operator A, which has dimension k^2. We then compute the smallest eigenvalue of A by executing BlockVarII$(1,m)$ for $k = 25, 30, \ldots, 55$ and $m = 2, 3, \ldots, 72$. Figure 2.7 shows the number of outer iterations required for convergence of the smallest Ritz value when $k = 50$ plotted against the value of the parameter m. Also presented is a plot of the total number of inner iterations required for convergence for the same problem. Table 2.2 shows the inner iteration value, denoted $\hat{m}$ which gave the fewest total inner iterations. Also presented is the ratio $\frac{\hat{m}}{k} = \frac{\hat{m}}{\sqrt{n}}$, and this data is used to justify the selection of $m \approx \frac{1}{2}\sqrt{n}$ for the more difficult problems seen later. We stress, however, that generally, the optimal value m is linked to the relative difficulty of the problem; for easier problems, smaller values of m give excellent performance, while for more difficult problems, larger values of m are required. This example indicates that the optimal number may be linked somehow to the condition number of the matrix as for finite difference discretizations of elliptic partial differential equations grows as $O(k^2)$. This connection is tenuous at best, but deserves more investigation.

In their package Eigifp, Money and Ye [40] begin iterations with $m = 3$ and then increase if the convergence rate they are tracking is not suitable. We include no such strategy in our codes, but acknowledge the benefit of such a strategy, as for more difficult problems a small number of inner iterations will not suffice.

2.6.5 Example: adaptive block sizing

Motivated by the work in [63], we have discussed a strategy for adaptively choosing the block size. Here, we examine the effect of adaptive block sizing on the convergence of the block algorithm. We again consider the Laplacian eigenvalue problem (2.34), however, now the domain under consideration is the four node domain depicted in Figure 2.8. This mesh is less refined, containing only 1553 nodes of which 1217 are interior nodes. The four node structure of this domain gives the discretized problem



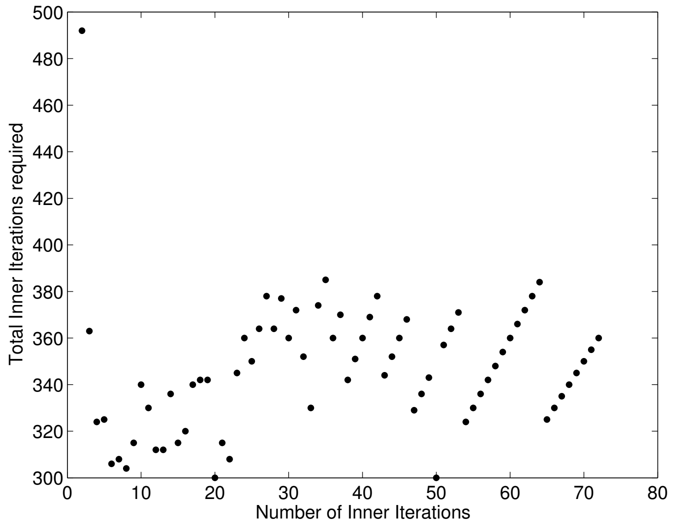


Figure 2.7: Outer and total inner iterations versus m.

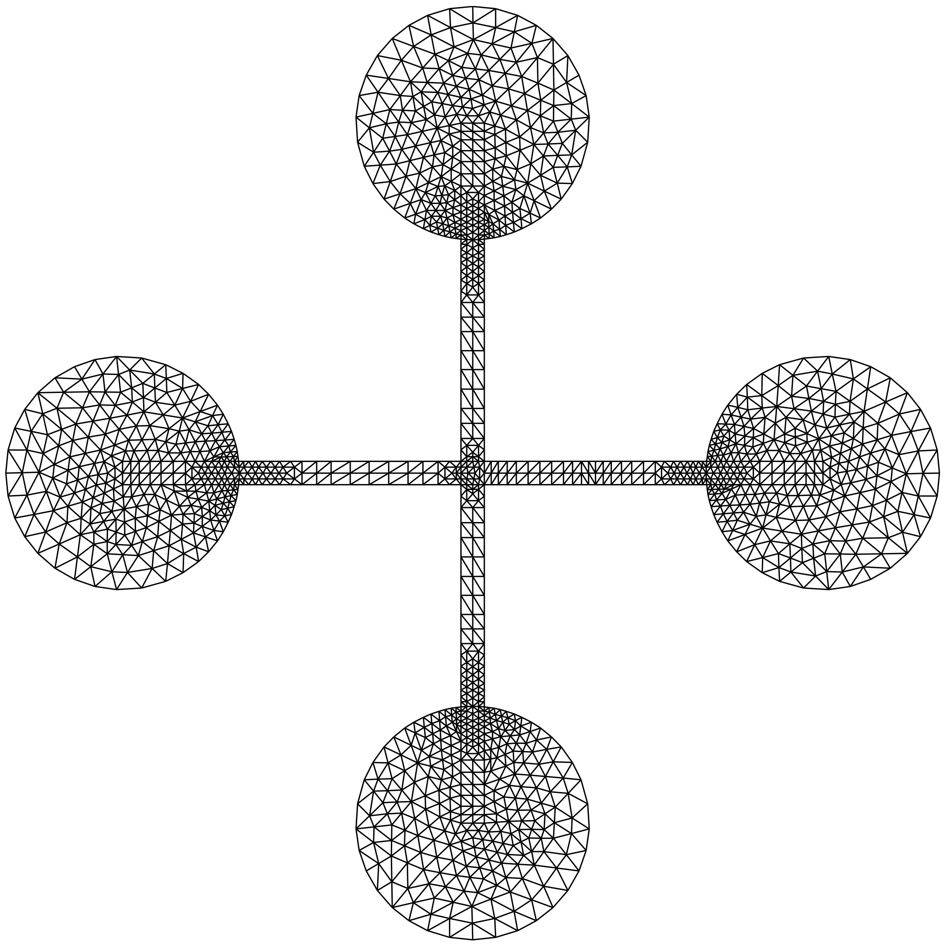


Figure 2.8: Mesh for four node domain.

eigenvalues in clusters of four. As before, the mesh and the resulting finite element formulation are obtained from the MATLAB PDE toolbox. The convergence history of adaptBlockVarIIp(2, 4) is presented in Figure 2.9. The initial block size in this case is two, and the final block size is five, corresponding to the clustering of the four algebraically smallest eigenvalues. To obtain these approximations, 2966 matrix vector multiplications were required. On the other hand, with a fixed initial block size of four and no adaptive block sizing, the algorithm required only 2867 matvecs to converge. Indeed, as pointed out in [63], the algorithms will perform better in general if a suitable block size is known a priori. Again, such knowledge may not be at hand, and an adaptive scheme such as this may be necessary in detecting multiplicities.

2.6.6 Example: the need for accelerated convergence

The problems considered above are representative of many engineering problems, however, they are well-conditioned and usually serve only as model problems. Here we consider problems from the popular Harwell-Boeing collection [16] obtained from the

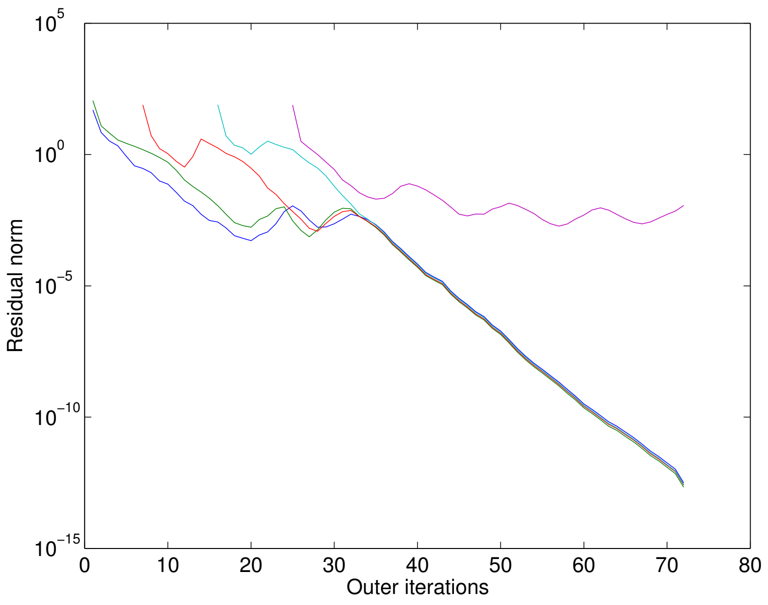


Figure 2.9: Convergence history of adaptBlockVarIIp(2,4).

Matrix Market [39]. In particular, we consider the problems from the BCSSTRUC collection. This collection is intended to be representative of modeling techniques in the dynamic analysis of structures. We demonstrate how poorly the algorithm performs on these more realistic problems in an effort to motivate the need for preconditioning, which will be discussed at length in Chapter 3. We note that for some of these problems (BCSST01, BCSST03, BCSST04, BCSST13), the mass matrix is only semidefinite and not positive definite as the algorithm theoretically requires. Since the stiffness matrices associated with these problems are all positive definite, the algorithm has no trouble, as is demonstrated below. Generally, we may expect no troubles in the presence of a semidefinite mass matrix provided that the mass matrix and the stiffness matrix do not have a common null space.

The examples considered here were executed using Block Variant II with the band Arnoldi-like process as the basis construction method. Adaptive block sizing was used, and the previous Ritz vectors were added to the subspace in an effort to accelerate convergence. For all problems, we used $m = \text{ceil}\left(\frac{1}{2}\sqrt{n}\right)$ as the number of inner iterations, as motivated by the discussion in Section 2.6.4 above.

The algorithm terminated either when the initial residual norm was reduced by ten orders of magnitude, or when a maximum of 1000 outer iterations was reached. Table 2.3 details the number of matrix-vector products (matvecs) required of the algorithm and the final residual norm. The asterisk $(*)$ indicates that the maximum number of outer iterations was reached.

Problem	size	m	matvecs	final residual
BCSST01	48	4	59033	1.518e+03*
BCSST02	66	5	1761	6.395e−07
BCSST03	112	6	41533	1.785e−01
BCSST04	132	6	30527	2.409e−03
BCSST05	153	7	3873	9.765e−05
BCSST06	420	11	115863	1.516e+01*
BCSST07	420	11	166481	1.376e−02
BCSST08	1074	17	78029	3.177e−03
BCSST09	1083	17	2076	8.899e−02
BCSST10	1086	17	190166	8.317e−02*
BCSST11	1473	20	286749	2.419e+00*
BCSST12	1473	20	179429	1.465e+03*
BCSST13	2003	23	396390	2.569e+03*
BCSST19	817	15	188746	1.470e+06*
BCSST20	485	12	233634	2.908e+06*
BCSST21	3600	30	103376	3.563e−01
BCSST22	138	6	8519	1.623e−03
BCSST23	3134	28	235638	7.954e+06*
BCSST24	3562	30	402239	4.728e+05*

Table 2.3: Convergence characteristics of adaptBlockVarIIp(2, m) applied to some BCS matrices from the Harwell-Boeing collection.

Chapter 3

Preconditioning Inverse Free Algorithms for (A, B)

Iterative methods such as Krylov methods frequently suffer from slow convergence. In many cases, it is desirable to accelerate this convergence by applying a preconditioner. Preconditioning a Krylov subspace method usually means that instead of operating with $\mathcal{K}_m(A, x)$, we work with the Krylov space $\mathcal{K}_m(M^{-1}A, x)$, where M is the preconditioner. Usually M is chosen so that $M^{-1}A$ approximates the identity in some sense, for in this case, Krylov methods will converge very quickly. M should also be as inexpensive as possible to construct, and M^{-1} should be as inexpensive as possible to apply to a vector. As the only requirement is the action of M^{-1} on a vector, preconditioners need not be explicit matrices; they may be operators defined by methods such as multilevel methods or Krylov methods themselves. In this work, we investigate the viability of incomplete factorization preconditioners in eigenvalue computations. For more information regarding preconditioning techniques, we refer the interested reader to Benzi's survey [5] and Saad's book [52]. We begin by reviewing how incomplete factorization preconditioners are constructed, and move into a discussion about preconditioning the algorithms of Chapter 2.

3.1 Incomplete Factorizations—Background

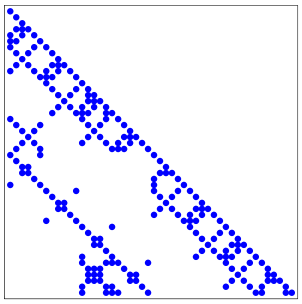
For large sparse matrices, computing a complete factorization such as an LU or QR factorization is usually intractable, since the factors may experience fill-in as the factorization progresses. That is, the factors of a sparse matrix may have a significant increase in the number of nonzeros as compared to the number of nonzeros in the original matrix. For this reason, iterative methods become attractive if not necessary, and their convergence is accelerated by using *incomplete* factors as preconditioners.

Two important ideas when considering the quality of preconditioners are the ideas of accuracy and stability. The accuracy of a preconditioner refers to the difference between the preconditioner and the matrix. Suppose a matrix M is to be used as a preconditioner for a matrix A; that is, we'll perform iterations with the matrix $M^{-1}A$. Accuracy of the preconditioner is measured by the quantity $\|A - M\|_F$. Stability, on the other hand, refers to how well the preconditioned matrix approximates the identity and is measured by the quantity $\|I - M^{-1}A\|_F$. Saad points out [52] that the important quantity to examine is usually not the error (accuracy), but rather the preconditioned error (stability). The desire to produce more stable incomplete factors led to the development of sparse approximate inverse preconditioners, which will be considered in Section 3.2.

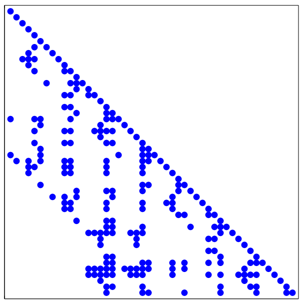
There are many flavors of incomplete factorizations corresponding first to the type of factorization sought, and second, to different dropping strategies. We will briefly review two basic types of incomplete factorizations—incomplete LU (ILU) factorizations, and incomplete LQ (ILQ) factorizations. First, however, we note that there are essentially two dropping strategies—static pattern and dynamic pattern. A static pattern dropping strategy is one in which a pattern for the nonzero structure of the incomplete factorization is determined a priori. A good example of a static pattern dropping strategy is a zero-fill strategy. That is, only positions which contained nonzeros in the matrix to be factored are permitted to be nonzero in the factors. On the other hand, a dynamic pattern dropping strategy develops a pattern for the nonzero structure during the computation of the factor. Usually, the dynamic pattern is determined by the relative sizes of the elements being computed; the smaller elements in a given row or column are dropped after they are computed. Figure 3.1 shows the sparsity pattern of a zero-fill incomplete Cholesky factor and an incomplete Cholesky factor obtained with threshold dropping. The factorized matrix is the symmetric positive definite matrix BCSSTK01 from the Harwell-Boeing sparse matrix collection [16], and obtained from the Matrix Market [39]. Both of the factors have exactly the same number of nonzeros, but rather different nonzero structures, and both perform rather well when used as preconditioners for systems involving the BCSSTK01 matrix.

3.1.1 ILU factorizations

With regards to specific factorizations, ILU factorizations are generally favored as they are quite easy to compute even in the case of a sparse matrix. In implementations involving sparse matrices, special data structures are typically used which store only the nonzeros. Two very popular such structures are compressed sparse row (CSR) and compressed sparse column (CSC) structures. The remainder of our discussion will be based on the assumption that CSC structures are used. For matrices stored in CSC format, one may use a tailored form of Gaussian elimination known as the *jki*-variant which requires the update of only single columns at a time with only requiring



(a) Zero-fill factor



(b) Threshold dropping factor

Figure 3.1: Sparsity patterns of two incomplete Cholesky factors

the access of the previous columns. The use of a tailored Gaussian elimination makes the computation of ILU factorizations very efficient.

The most popular static pattern ILU factorizations are based on a concept known as level of fill and are denoted by ILU(p) where p is an integer indicating the level of fill. We do not make use of these factorizations here, with the exception of the ILU(0), the zero-fill ILU, and we point the interested reader to [12],[49],[52]. Instead, we favor the dynamic pattern ILU factorizations developed via numerical dropping. Such factorizations are known as threshold ILU factorizations and are denoted by ILUT(τ) where τ is the drop tolerance. Here, each column of the ILU is computed and then subjected to a dropping rule. In computing a column w, it is typical to drop the elements which are smaller than $\tau\|w\|$. Notably, it is impossible to control fill for an arbitrary tolerance τ and therefore, another parameter, a fill-control parameter p, may be included. This factorization is called ILUT(τ, p). Once the column is computed, only the p entries with the largest magnitude are kept in the L factor, and similarly, the p largest entries are kept in the U factor, including the diagonal entry which is always kept. Unfortunately, ILUT may fail due to encountering a zero pivot, or exponential pivot growth. Also, ILUT may produce two factors which though nonsingular, are entirely unstable. To remedy this, one may implement a partial pivoting strategy and obtain an algorithm known as ILUTP, where the P stands for pivoting. Again, we point the interested reader to details found in [51]. Finally we note that ILUT fails to produce a symmetric matrix and as a result Chow and Saad exploit yet another variant of Gaussian elimination to develop the ILUS factorization [13].

When dealing with symmetric positive definite matrices, it is desirable to compute an incomplete Cholesky (IC) factor, and there has been much work devoted to this particular subject. It is well-known [5], [52] that IC factorizations for a matrix exist when the matrix is an H-matrix with positive diagonal entries (a generalization of the concept of M-matrices), otherwise, however, much work may be required to prevent breakdowns of the incomplete factorization process. A popular method is the method of Ajiz-Jennings which compensates for the dropping of entries l_{ij} by adding $|l_{ij}|$ to the diagonal entries a_{ii} and a_{jj}. For details on how this is done exactly, see [1]. We mention this method here as we will consider it in our numerical examples, thanks to an implementation obtained from Miroslav Tuma [55].

3.1.2 ILQ Factorizations

We may also produce incomplete triangular-orthogonal factorizations, or ILQ factorizations. Preconditioners based on ILQ factorizations are most frequently used in conjunction with the Conjugate Gradient method applied to the normal equations for Linear Least squares problems. We briefly review some of the work done in this area as we will consider normal equations like operators in Chapter 4.

A straightforward ILQ is obtained by performing Gram-Schmidt orthogonalization incompletely. In computing ILQ factorizations with Gram-Schmidt, we note that the columns of the orthogonal factor must be kept in order to orthogonalize successive columns against them. In general, the columns of Q are kept after some dropping rule has been applied. The application of this dropping rule almost always forces Q to lose orthogonality, and in fact, the Q factor may even become singular. In view of this, for the normal equations, Q-free methods are favored. One such method is called compact incomplete modified Gram-Schmidt (CIMGS) [59], [60], and exploits the fact that the complete Cholesky factor of AA^* is exactly the triangular factor of the LQ factorization of A. During the execution of CIMGS, the rows of the normal equations operator are formed as needed, and the entire operator is never explicitly formed. We will not consider preconditioners obtained from CIMGS in this work.

Rather, we consider ILQ factorizations obtained from execution of incomplete Givens orthogonalization IGO [4], [45]. The advantage of IGO is that it always produces a unitary Q. For our purposes, Q will not even be necessary, however, implicit maintenance of the relation $Q^*Q = I$ will be desirable. As with other incomplete factorization techniques, the IGO process is subject to a dropping strategy, either static or dynamic. In our implementation, we use threshold dropping (a variant called TIGO in [4], [45]) to determine if an element should be rotated (pre-filtration), and then threshold dropping is applied to prune the resulting column (post-filtration). Although much attention has not gone to using Householder reflectors in constructing ILQ factorizations, they may be used as well. Preconditioners based on Householder ILQ factorizations will be considered in some detail in Chapter 4.

3.2 Sparse Approximate Inverses—Background

Another class of preconditioners are the sparse approximate inverse preconditioners which are motivated by the desire to construct a preconditioner M for a matrix A which approximates A^{-1} directly. The foundations of this desire lie in producing a more stable preconditioner, by eliminating the need to apply M^{-1}. Many methods have been considered to develop such preconditioners, although it seems that such preconditioners are not nearly as popular as ILU type preconditioners. We review one type of sparse approximate inverse preconditioner for symmetric positive definite matrices, as it is the only such preconditioner we will consider. The method we consider has with it an associated incomplete factorization which we will also consider. For details and references concerning other sparse approximate inverse preconditioners, see [5] and [52].

The SAINV [6] and RIF [7] preconditioners arise from the observation that when A is symmetric positive definite, we may orthogonalize a collection of vectors with respect to the inner product defined by A to obtain

$$Z^*AZ = D \tag{3.1}$$

where D is a diagonal matrix. We may rewrite (3.1) as $A = Z^{-*}DZ^{-1}$ and invert to obtain

$$A^{-1} = ZD^{-1}Z^*. \tag{3.2}$$

Performing the A-orthogonalization incompletely via modified Gram-Schmidt leads to the SAINV preconditioner. In a typical implementation, Z is initialized to be the identity and incomplete modified Gram-Schmidt with threshold dropping is used to incompletely A-orthogonalize the columns of Z. When no dropping is performed, this process computes a QR factorization of the identity such that the columns of Z are A-orthogonal. Thus we have formed

$$I = ZL^* \tag{3.3}$$

where L is unit lower triangular (since normalization is not required). It follows, then, that $Z = L^{-*}$ and so we have

$$A = LDL^* \quad \text{and} \quad A^{-1} = L^{-*}D^{-1}L^{-1}.$$

The RIF preconditioner takes $LD^{\frac{1}{2}}$ as the lower triangular incomplete factor. Post-filtration is applied to the elements of L to maintain a sparser factor. We note that post-filtration is not required in construction of the SAINV preconditioner, as there is no need to store the multipliers once they have been used. As all that is required in the construction of SAINV and RIF preconditioners is computation of A inner products we may construct preconditioners of these types for the normal equations without ever having to form A^*A explicitly. This is the subject of [8] and will be considered for our problems in Chapter 4.

3.3 Preconditioning Inverse free algorithms

In [25], Golub and Ye note that when ρ satisfies $\lambda_1 < \rho < \lambda_2$ then an LDL^* factorization of $A - \rho B$ may be found and scaled such that

$$L^{-1}(A - \rho B)L^{-*} = D = \text{diag}(-1, 1, 1, \ldots, 1). \tag{3.4}$$

We may then use the L factor as a preconditioner by noting that the pencil

$$(L^{-1}AL^{-*}, L^{-1}BL^{-*})$$

has exactly the same eigenvalues as the pencil (A, B). Supposing that ρ as above is a Ritz value obtained from Algorithm 2.2, the iteration matrix

$$L^{-1}AL^{-*} - \rho L^{-1}BL^{-*} = L^{-1}(A - \rho B)L^{-*} = D$$

has exactly two eigenvalues, namely $\sigma_1 = -1$ and $\sigma_2 = \cdots = \sigma_n = 1$. Algorithm 2.2 therefore enjoys asymptotically quadratic convergence even when $m = 1$. Of course in practice, one would not compute a complete LDL^* factorization at each outer iteration, however, one may compute an incomplete factorization of $A - \theta_k B$ at each outer iteration. This is still a costly operation, so Golub and Ye propose instead to compute an incomplete factorization of $A - \hat{\lambda}B$ where $\hat{\lambda}$ is some approximation to the smallest eigenvalue λ_1. This factor may then be used as a preconditioner to give improved convergence.

It is also pointed out in [25] that applying Algorithm 2.2 to the pencil $(\hat{A}, \hat{B}) \equiv (L^{-1}AL^{-*}, L^{-1}BL^{-*})$ does not require the explicit formation of $\hat{A}$ and $\hat{B}$, however, we may apply L^{-1} and L^{-*} in such a way as to implicitly construct the desired basis. We repeat the discussion found in [25] and show that the formulation of B-orthonormal preconditioned Arnoldi found there applies to the block Arnoldi-like process (Algorithm 2.4) and therefore to the band Arnoldi-like process (Algorithm 2.5).

The block Arnoldi-like recurrence for $(\hat{A}, \hat{B})$ for the construction of a basis $\hat{Z}$ orthogonal with respect to the $\hat{B}$ inner product is given by

$$\hat{Z}_{j+1}H_{j,j+1} = \hat{A}\hat{Z}_j - \hat{B}\hat{Z}_j\Theta - \sum_{i=1}^{j} \hat{Z}_i H_{ij} \tag{3.5}$$

where

$$H_{ij} = \hat{Z}_i^* \hat{B}(\hat{A}\hat{Z}_j - \hat{B}\hat{Z}_j\Theta). \tag{3.6}$$

Substituting $\hat{A} := L^{-1}AL^{-*}$ and $\hat{B} := L^{-1}BL^{-*}$ into (3.6) we obtain

$$H_{ij} = \hat{Z}_i^* L^{-1}BL^{-*}L^{-1}(AL^{-*}\hat{Z}_j - BL^{-*}\hat{Z}_j\Theta). \tag{3.7}$$

Defining $Z_j := L^{-*}\hat{Z}_j$, we see that

$$H_{ij} = Z_i^* BL^{-*}L^{-1}(AZ_j - BZ_j\Theta), \tag{3.8}$$

and we have the recurrence

$$Z_{j+1}H_{j,j+1} = L^{-*}L^{-1}(AZ_j - BZ_j\Theta) - \sum_{i=1}^{j} Z_i H_{ij} \tag{3.9}$$

defining the preconditioned block Arnoldi-like process. We note that as (3.5) constructs a basis $\hat{Z}$ which is orthogonal with respect to the inner product defined by $\hat{B}$, the relation (3.9) constructs a basis Z which is orthogonal with respect to the inner product defined by B. To incorporate the preconditioned block Arnoldi-like process, we form the projected problem (Z^*AZ, Z^*BZ) as usual. For the sake of completeness, we include a statement of the preconditioned block Arnoldi-like process for $\mathrm{op}(X) := AX - BX\Theta$.

Algorithm 3.1 Preconditioned Arnoldi-like process for $\mathrm{op}(X) := AX - BX\Theta$

```
Input: $A, B \in \mathbb{R}^{n\times n}, \Theta \in \mathbb{R}^{p\times p}$, s.p.d. $M$, $Z_1 \in \mathbb{R}^{n\times p}$, with $Z_1^* M Z_1 = I_p$, $m \geq 1$,
    and $L \in \mathbb{R}^{n\times n}$
1: for $j = 1, \ldots, m$ do
2:   $W_j = L^{-*}L^{-1}(AZ_j - BZ_j\Theta)$
3:   for $i = 1, \ldots, j$ do
4:     $H_{ij} = Z_i^* M W_j$
5:     $W_j = W_j - Z_i H_{ij}$
6:   end for
7:   Compute $W_j = Z_{j+1}H_{j+1,j}$, the $QR$ factorization of $W$ with respect $\langle\cdot,\cdot\rangle_M$.
8: end for
```

Since all that is required to precondition the block Arnoldi-like process is application of the operator $L^{-*}L^{-1}$, which presumably approximates $A - \theta B$ for some θ, we may consider using various types of preconditioners, not necessarily incomplete LDL^* factorizations. We examine three different preconditioning strategies below and discuss their strengths and weaknesses.

First, we recall that from Theorem 2.2.1, we know that the rate of convergence of the sequence $\left\{\theta_1^{(k)}\right\}$ generated by Algorithm 2.2 may be bounded by ϵ_m^2 where

$$\epsilon_m \leq 2\left(\frac{1-\sqrt{\psi}}{1+\sqrt{\psi}}\right)^m, \quad \psi := \frac{\sigma_2 - \sigma_1}{\sigma_n - \sigma_1} \tag{3.10}$$

where $\sigma_1 \le \sigma_2 \le \cdots \le \sigma_n$ are the eigenvalues of the iteration matrix $A - \theta_1^{(k)} B$. We call ψ the relative spectral gap. As the bound on ϵ_m shows, increasing the relative spectral gap results in better convergence behavior for the algorithm. As we include the action of a preconditioner in the algorithm, the iteration matrix we will consider is $L^{-1}(A - \theta_1^{(k)} B)L^{-*}$, and we wish to investigate relative spectral gap ψ of this matrix, whose eigenvalues we will denote by σ_i, with $1 \le i \le n$. The following lemma will be exploited in the study of certain preconditioners.

Lemma 3.3.1. *Let $L^{-1}(A - \theta_1^{(k)} B)L^{-*} = F + G$ where F and G are symmetric. Let the eigenvalues of F be denoted by $\delta_1 \le \cdots \le \delta_n$ and let* $\operatorname{spread}(G) := \lambda_n(G) - \lambda_1(G)$ *be the spread of the eigenvalues of the matrix G. Then, the spectral gap of the iteration matrix is bounded as*

$$\frac{\delta_2 - \delta_1 - \operatorname{spread}(G)}{\delta_n - \delta_1 + \operatorname{spread}(G)} \le \frac{\sigma_2 - \sigma_1}{\sigma_n - \sigma_1} \le 1. \tag{3.11}$$

Proof. By the Weyl monotonicity theorem, for each i such that $1 \le i \le n$ we have that $\lambda_1(G) \le \sigma_i - \delta_i \le \lambda_n(G)$. Exploiting this, we see that

$$\delta_2 - \delta_1 - (\lambda_n(G) - \lambda_1(G)) \le \sigma_2 - \sigma_1 \tag{3.12}$$

and

$$\delta_n - \delta_1 + (\lambda_n(G) - \lambda_1(G)) \ge \sigma_n - \sigma_1 \tag{3.13}$$

The result immediately follows. $\square$

We note that spread(G) may be very crudely bounded by

$$\operatorname{spread}(G) \le 2\|G\|_2 \le 2\|G\|_F. \tag{3.14}$$

and so we may write (3.11) as

$$\frac{\delta_2 - \delta_1 - 2\|G\|_F}{\delta_n - \delta_1 + 2\|G\|_F} \le \frac{\sigma_2 - \sigma_1}{\sigma_n - \sigma_1} \le 1. \tag{3.15}$$

3.3.1 Factoring the iteration matrix

The first preconditioner we consider is that obtained from an incomplete LDL^* factorization of the iteration matrix $A - \theta_1^{(k)} B$ for some $\theta_1^{(k)}$. This preconditioning scheme is considered by Money and Ye [40] and proves to be very effective, however, knowledge of $\theta_1^{(k)}$ is absolutely necessary. Moreover, $\theta_1^{(k)} \ge \lambda_1$ and this may cause difficulty in attempting to form stable incomplete factorizations of $A - \theta_1^{(k)} B$, as this iteration matrix is indefinite. When $\lambda_1 < \theta_1^{(k)} < \lambda_2$, however, the iteration matrix is only slightly indefinite (it has only one negative eigenvalue), and the quality of the preconditioners usually does not suffer. For our study, we assume that

$$A - \theta_1^{(k)} B = LDL^* + E \tag{3.16}$$

is an incomplete LDL^* factorization of $A - \theta_1^{(k)} B$ such that $D = \text{diag}(\pm 1)$. Then the preconditioned iteration matrix has the form

$$L^{-1}(A - \theta_1^{(k)} B)L^{-*} = D + L^{-1}EL^{-*} \tag{3.17}$$

Now the matrix D is diagonal with eigenvalues ± 1. In the case that D has -1 as a simple eigenvalue, we may apply Lemma 3.3.1 to obtain

$$\frac{2 - \text{spread}(L^{-1}EL^{-*})}{2 + \text{spread}(L^{-1}EL^{-*})} \leq \frac{\sigma_2 - \sigma_1}{\sigma_n - \sigma_1} \leq 1. \tag{3.18}$$

Again, $\text{spread}(L^{-1}EL^{-*})$ may be bounded by $2\|L^{-1}EL^{-*}\|_F$, and so the bound (3.18) reduces to

$$\frac{1 - \|L^{-1}EL^{-*}\|_F}{1 + \|L^{-1}EL^{-*}\|_F} \leq \frac{\sigma_2 - \sigma_1}{\sigma_n - \sigma_1} \leq 1. \tag{3.19}$$

Thus, the stability of the incomplete factorization is key in this case.

We remark that should D have -1 as an eigenvalue of multiplicity two, the best lower bound we may obtain for the relative spectral gap of the preconditioned iteration matrix is $\frac{\sigma_2 - \sigma_1}{\sigma_n - \sigma_1} \geq 0$, which gives no useful information. This does not, however, deem the preconditioner useless, rather, we simply have no a priori expectations of it. Similarly, when the D has only one distinct eigenvalue, the best lower bound we obtain is -1, which also fails to provide any meaningful information.

3.3.2 A fixed preconditioner

Another kind of preconditioning scheme which we may consider is that of building an incomplete factorization preconditioner of $A - \lambda_0 B$ where λ_0 is a lower bound for the spectrum of (A, B). In this case, the matrix $A - \lambda_0 B$ is symmetric positive definite and there are many more methods for the construction of incomplete factorizations for symmetric positive definite matrices than for indefinite ones. Indeed, let us suppose that

$$A - \lambda_0 B = LL^* + E \tag{3.20}$$

is an incomplete factorization of $A - \lambda_0 B$. The preconditioned iteration matrix then takes the form

$$\begin{aligned} L^{-1}(A - \theta_1^{(k)} B)L^{-*} &= L^{-1}(A - \lambda_0 B - (\theta_1^{(k)} - \lambda_0)B)L^{-*} \\ &= I + L^{-1}EL^{-*} - (\theta_1^{(k)} - \lambda_0)L^{-1}BL^{-*}. \end{aligned} \tag{3.21}$$

That is,

$$L^{-1}(A - \theta_1^{(k)} B)L^{-*} = \Big(I - (\theta_1^{(k)} - \lambda_0)L^{-1}BL^{-*}\Big) + L^{-1}EL^{-*}. \tag{3.22}$$

To apply Lemma 3.3.1 to bound the relative spectral gap, we compute the eigenvalues $\delta_1 \leq \cdots \leq \delta_n$ of the matrix $\left(I - (\theta_1^{(k)} - \lambda_0)L^{-1}BL^{-*}\right)$. To do this, denote the eigenvalues of the pencil (LL^*, B) by $0 < \eta_1 \leq \cdots \leq \eta_n$, and note that $(\theta^{(k)} - \lambda_0) \geq 0$. Therefore, it follows that

$$\delta_i = 1 - \frac{\theta_1^{(k)} - \lambda_0}{\eta_i} \tag{3.23}$$

for i such that $1 \leq i \leq n$. Therefore, for any $i \neq 1$ we have

$$\delta_i - \delta_1 = \frac{\theta_1^{(k)} - \lambda_0}{\eta_1 \eta_i}(\eta_i - \eta_1) \tag{3.24}$$

and applying Lemma 3.3.1, we see that

$$\frac{\eta_n}{\eta_2}\left(\frac{\eta_2 - \eta_1 - \frac{\eta_1\eta_2}{\theta_1^{(k)} - \lambda_0}\,\mathrm{spread}(L^{-1}EL^{-*})}{\eta_n - \eta_1 + \frac{\eta_1\eta_n}{\theta_1^{(k)} - \lambda_0}\,\mathrm{spread}(L^{-1}EL^{-*})}\right) \leq \frac{\sigma_2 - \sigma_1}{\sigma_n - \sigma_1}. \tag{3.25}$$

Again, we see that to guarantee good convergence characteristics, the quantity

$$\mathrm{spread}(L^{-1}EL^{-*}) \leq 2\|L^{-1}EL^{-*}\|_F = 2\|L^{-1}(A - \lambda_0 B)L^{-*} - I\|_F \tag{3.26}$$

should be made as small as possible. Producing stable incomplete factors will help to minimize spread$(L^{-1}EL^{-*})$. In the presence of "perfect" preconditioning, that is, $A - \lambda_0 B = LL^* + E$ with $E = 0$, the lower bound on the relative spectral gap is

$$\frac{\eta_n}{\eta_2}\left(\frac{\eta_2 - \eta_1}{\eta_n - \eta_1}\right) \tag{3.27}$$

where $\eta_i, 1 \leq i \leq n$ are the eigenvalues of (LL^*, B). As $E = 0$ in this case, $(LL^*, B) \equiv (A - \lambda_0 B, B)$ and so $\eta_i = \lambda_i - \lambda_0$ for $1 \leq i \leq n$. The bound (3.27) becomes

$$\frac{\lambda_n - \lambda_0}{\lambda_2 - \lambda_0}\left(\frac{\lambda_2 - \lambda_1}{\lambda_n - \lambda_1}\right) \tag{3.28}$$

thereby showing that the quality of the lower bound λ_0 directly influences convergence. That is, good approximations $\lambda_0 \approx \lambda_1$ may give better convergence behavior of the preconditioned algorithm.

In general, however, it is not desirable to compute a complete factor $A - \lambda_0 B = LL^*$ and so $\eta_i \neq \lambda_i - \lambda_0$. Again, we may appeal to the Weyl monotonicity theory to see that

$$\frac{\lambda_2 - \lambda_1 - \mathrm{spread}(E, B)}{\lambda_n - \lambda_1 + \mathrm{spread}(E, B)} \leq \frac{\eta_2 - \eta_1}{\eta_n - \eta_1} \leq \frac{\lambda_2 - \lambda_1 + \mathrm{spread}(E, B)}{\lambda_n - \lambda_1 - \mathrm{spread}(E, B)}. \tag{3.29}$$

The quantity spread(E, B) denotes the difference between the algebraically largest and algebraically smallest eigenvalues of the pencil (E, B) and indicates, in some measure, the *accuracy* of the incomplete factorization $A - \lambda_0 B = LL^* + E$. We note that stability of the incomplete factorizations is much more important to the quality of these types of preconditioners.

3.3.3 Factoring B

A third preconditioning style to consider is the construction of an incomplete factorization of the matrix B. The reason for considering such a preconditioner will be made clear when we examine the interior eigenvalue problem in Chapter 4. Suppose that we have constructed

$$B = LL^* + E. \tag{3.30}$$

Using L as the preconditioner as described above, our preconditioned iteration matrix becomes

$$L^{-1}(A - \theta_1^{(k)} B)L^{-*} = L^{-1}AL^{-*} - \theta_1^{(k)} I - \theta_1^{(k)} L^{-1}EL^{-*}. \tag{3.31}$$

We denote the eigenvalues of the pencil (A, LL^*) by $\eta_1 \leq \cdots \leq \eta_n$ and we note that the eigenvalues $\delta_1 \leq \cdots \leq \delta_n$ of the matrix $L^{-1}AL^{-*} - \theta_1^{(k)} I$ are given by

$$\delta_i = \eta_i - \theta_1^{(k)}. \tag{3.32}$$

As before, we apply Lemma 3.3.1 to obtain a lower bound on the relative spectral gap. We have

$$\frac{\eta_2 - \eta_1 - |\theta^{(k)}| \operatorname{spread}(L^{-1}EL^{-*})}{\eta_n - \eta_1 + |\theta^{(k)}| \operatorname{spread}(L^{-1}EL^{-*})} \leq \frac{\sigma_2 - \sigma_1}{\sigma_n - \sigma_1} \leq 1. \tag{3.33}$$

In the case of perfect preconditioning, the bound reduces to

$$\frac{\lambda_2 - \lambda_1}{\lambda_n - \lambda_1} \leq \frac{\sigma_2 - \sigma_1}{\sigma_n - \sigma_1} \leq 1. \tag{3.34}$$

That is, the distribution of the eigenvalues of (A, B) give a lower bound on the effectiveness of the preconditioned iteration. This is not at all surprising, as preconditioning the inverse free algorithm with a complete factor L such that $B = LL^*$ corresponds to applying restarted Lanczos to the pencil $(L^{-1}AL^{-*}, I)$. Thus we may consider this type of preconditioning to be in the class of algorithms for (A, B) which require an implicit inexact inversion of B at each step. We will discuss reasons for considering this approach in more detail in Chapter 4.

3.3.4 A small example

To illustrate the effects that each of these preconditioning schemes has on the convergence of the algorithms of Chapter 2, consider the pencil (A, B) where

$$A = \begin{pmatrix} 1 & & & \\ & 2 & & \\ & & \ddots & \\ & & & 1000 \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} 1000 & & & \\ & 999 & & \\ & & \ddots & \\ & & & 1 \end{pmatrix}. \tag{3.35}$$

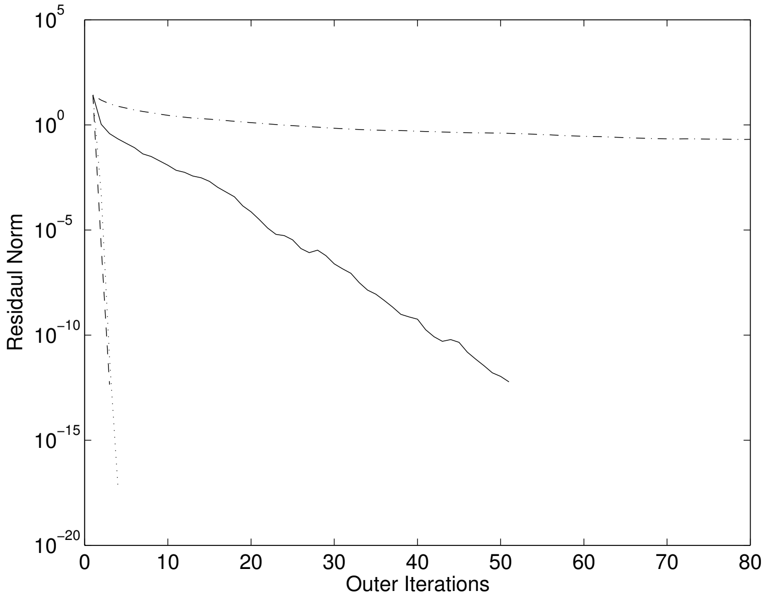


Figure 3.2: Convergence histories for the diagonal problem with various preconditioners

Since the matrices are diagonal, we demonstrate the effects of perfect preconditioning. The convergence histories of the preconditioned iterations appear in Figure 3.2. The eigenvalues of (A, B) are given by $\lambda_i = \frac{i}{1001-i}$, and for any $\theta^{(k)}$, the eigenvalues of the iteration matrix are $\sigma_i = i\left(1+\theta^{(k)}\right) - 1001\theta^{(k)}$ giving a relative spectral gap of the (unpreconditioned) iteration matrix as

$$\frac{\sigma_2 - \sigma_1}{\sigma_{1000} - \sigma_1} = \frac{1}{999} \approx 1\text{e} - 03. \tag{3.36}$$

The solid line in Figure 3.2 represents the convergence history of the unpreconditioned algorithm.

Using the L factor from a complete LDL^* factorization of $A - \theta^{(k)}B$ will give, in theory, quadratic convergence, once the approximation $\theta^{(k)}$ lies between the two algebraically smallest eigenvalues. We use the L factor from the complete LDL^* factorization of $A - \frac{1}{2}(\lambda_1 + \lambda_2)B$ to illustrate the kind of convergence one may expect. The convergence history of the algorithm using this preconditioner is the dotted line in Figure 3.2.

Noting that A is positive definite, we have $\lambda_0 = 0$ as a lower bound for the spectrum of (A, B) and we may construct a preconditioner as discussed in Section 3.3.2. As described above, use of the perfect preconditioner gives a lower bound on the relative spectral gap of the preconditioned iteration matrix as

$$\frac{\lambda_{1000} - \lambda_0}{\lambda_2 - \lambda_0}\frac{\lambda_2 - \lambda_1}{\lambda_{1000} - \lambda_1}. \tag{3.37}$$

In this case, the bound becomes

$$\frac{1000-0}{\frac{2}{999}-0}\frac{\frac{2}{999}-\frac{1}{1000}}{1000-\frac{1}{1000}}=\frac{1000}{\frac{2}{999}}\frac{1001}{999}\frac{1}{999999}\approx\frac{1}{2}, \tag{3.38}$$

a quantity much larger than the relative spectral gap of the unpreconditioned iteration matrix. Using this preconditioner, we may expect much better convergence than experienced by the unpreconditioned iteration. The convergence history of the algorithm utilizing this preconditioner appears as the dashed line in Figure 3.2.

On the other hand, using a Cholesky factor of B as a preconditioner gives a lower bound on the relative spectral gap as the relative spectral gap of the eigenvalues of (A, B). In this case, the bound is

$$\frac{\lambda_2-\lambda_1}{\lambda_{1000}-\lambda_1}=\frac{\frac{2}{999}-\frac{1}{1000}}{1000-\frac{1}{1000}}\approx 1\text{e}-06, \tag{3.39}$$

a quantity much smaller than the bound on the relative spectral gap of the unpreconditioned iteration matrix. Using this preconditioner, we can expect much worse convergence behavior than that experienced by the unpreconditioned iteration. In fact, this is the case as shown by the dash-dotted line in Figure 3.2.

We note that there is almost no difference between the convergence rates of the algorithm using the preconditioner from the LDL^* factorization of the iteration matrix and algorithm using the Cholesky factor of A as the preconditioner. However, as our study above shows, if λ_0, the lower bound of our spectrum, is not a decent approximation of λ_1, with respect to the size of $\lambda_2-\lambda_1$, then the preconditioner obtained from the factor of $A-\lambda_0 B$ is of lower quality. Figure 3.3 shows the convergence histories of the algorithm for (A, I) preconditioned by factors of $A-\lambda_0$ for $\lambda_0 = 0, -10, -100, -1000$. Notice how the quality of the preconditioner diminishes as the difference $\lambda_1-\lambda_0$ increases.

3.4 Numerical Examples II

We revisit many of the examples from Section 2.6 and demonstrate the effectiveness of the preconditioning techniques described in this chapter. In running these examples, we used and benefited from some codes written by Miroslav Tuma and Michele Benzi. In particular, we have obtained Fortran 90 implementations of RIF, SAINV, RIFNR, and ICAJ along with wrappers so that they may be called from Matlab.

3.4.1 Example: Harwell-Boeing problems revisited

We begin by revisiting the problems considered in Section 2.6.6 which motivated the need for preconditioning. All of the problems considered have a positive definite

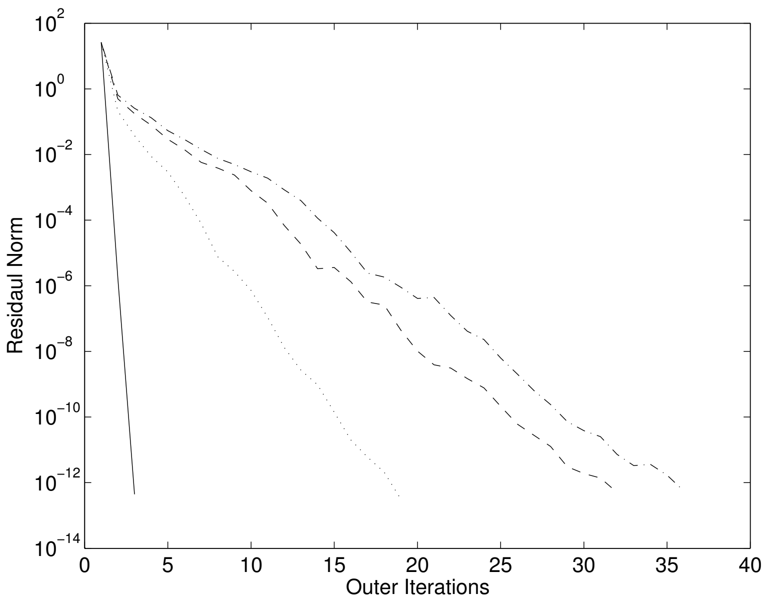


Figure 3.3: Convergence histories using preconditioners generated with variable λ_0. $\lambda_0 = 0$(solid), $\lambda_0 = -10$ (dotted), $\lambda_0 = -100$(dashed), $\lambda_0 = -1000$ (dash-dotted)

stiffness matrix K, and a positive semidefinite mass matrix M and therefore $\lambda_0 = 0$ is a lower bound for the spectrum of (K, M). For each matrix, we used $m = \text{ceil}(\frac{1}{4}\sqrt{n})$ as the number of inner iterations, half the number of inner iterations used in the examples in Section 2.6.6. For each problem we used the RIF of K with drop tolerance $1\text{e}-01$ and post-filtration tolerance $1\text{e}-01$. Convergence was declared when the residual norm fell below 1e−06, and a maximum of 1000 outer iterations was allowed. Table 3.1 reports the results of these test runs. Columns three and four are simply columns four and five of Table 2.3, and are included for direct comparison. Again, the asterisk($*$) indicates that the maximum number of outer iterations was reached before the residual norm fell to the desired level. Note that only four problems were "unable to be solved" when preconditioning was employed. We note that for each of these problems, BCSST19, BCSST20, BCSST23, and BCSST24, the stiffness matrices are very poorly conditioned, with two norm condition numbers all on the order of 1e+12. Also, the mass matrix from BCSST24 has condition number 1.8e+13. With this in mind it is unreasonable to expect to get residual norms on the order of $1\text{e}-06$ as we requested. For each of these problems, the algorithm resolved the eigenpairs to the order of the final residual fairly quickly and then stagnated. For example, for BCSST19, the residual norm at outer iteration number 480 was 9.731e−05, and the algorithm stagnated there until outer iteration 1000 was reached. It is worth remarking that the algorithm found four Ritz pairs with residual norms in the neighborhood of $1\text{e}-04$ after 1000 outer iterations were completed. The other three

		No Preconditioner		RIF Preconditioner		
Problem	size	matvecs	final residual	matvecs	preapps	final residual
BCSST01	48	59033	1.518e+03*	240	70	5.072e−07
BCSST02	66	1761	6.395e−07	373	126	8.243e−08
BCSST03	112	41533	1.785e−01	2474	825	8.983e−07
BCSST04	132	30527	2.409e−03	435	147	2.919e−07
BCSST05	153	3873	9.765e−05	491	180	1.966e−07
BCSST06	420	115863	1.516e+01*	597	240	6.947e−07
BCSST07	420	166481	1.376e−02	627	252	2.517e−07
BCSST08	1074	78029	3.177e−03	249	108	9.463e−08
BCSST09	1083	2076	8.899e−02	712	306	7.746e−07
BCSST10	1086	190166	8.317e−02*	2723	1170	8.786e−07
BCSST11	1473	286749	2.419e+00*	16855	7330	8.581e−07
BCSST12	1473	179429	1.465e+03*	30257	13160	9.620e−07
BCSST13	2003	396390	2.569e+03*	4101	1824	7.328e−07
BCSST19	817	188746	1.470e+06*	67518	28432	5.913e−05*
BCSST20	485	233634	2.908e+06*	72770	29112	1.511e−03*
BCSST21	3600	103376	3.563e−01	789	360	6.419e−07
BCSST22	138	8519	1.623e−03	527	177	2.138e−07
BCSST23	3134	235638	7.954e+06*	121760	54992	4.511e−04*
BCSST24	3562	402239	4.728e+05*	127108	57780	1.360e−05*

Table 3.1: Convergence characteristics of adaptBlockVarIIp(2, m) applied to some BCS matrices from the Harwell-Boeing collection with and without preconditioning.

Problem	nnz(K)	nnz(L)	$\|K - LL^*\|_F/\|K\|_F$	$\|I - L^{-1}KL^{-*}\|_F$
BCSST01	400	597	6.381e−03	7.188e−01
BCSST02	4356	864	7.073e−02	2.257e+00
BCSST03	640	380	6.815e−03	3.371e+00
BCSST04	3648	3239	1.323e−02	2.286e+00
BCSST05	2423	2470	5.088e−02	3.290e+00
BCSST06	7860	9298	1.508e−02	4.821e+00
BCSST07	7860	9298	1.508e−02	4.821e+00
BCSST08	12960	28043	9.200e−03	2.498e+00
BCSST09	18437	18378	5.214e−02	7.657e+00
BCSST10	22070	18402	3.075e−02	9.810e+00
BCSST11	34241	48234	1.618e−02	1.651e+01
BCSST12	34241	48234	1.618e−02	1.651e+01
BCSST13	83883	168344	3.028e−02	1.228e+01
BCSST19	6853	11401	1.623e−02	9.661e+00
BCSST20	3135	2325	3.531e−03	7.012e+00
BCSST21	26600	47184	3.100e−02	1.139e+01
BCSST22	696	520	1.004e−02	2.745e+00
BCSST23	45178	117262	2.102e−02	1.180e+01
BCSST24	159910	276445	2.419e−02	2.079e+01

Table 3.2: Preconditioner statistics for Harwell-Boeing matrices.

problems expressed similar behavior.

Table 3.2 details some statistics regarding the preconditioners. We include measures of relative accuracy ($\|K - LL^*\|_F/\|K\|_F$), stability ($\|I - L^{-1}KL^{-*}\|_F$), and the number of nonzeros in the incomplete factor. Note that all of the preconditioners are relatively accurate, but stability varies a bit more. In particular, note that the problems which have higher stability estimates seem to require more matvecs to converge. Examples include BCSST11, BCSST12, and BCSST13. Though we can draw no hard and fast conclusions, these data support our discussion regarding the primacy of stability.

3.4.2 Example: Preconditioning and clustered eigenvalues

As our second example, we take the example from Section 2.6.1 and we demonstrate the effectiveness of the preconditioning scheme described in Section 3.3.1. The two algebraically smallest eigenvalues of the finite element model of the Laplacian on the barbell shaped domain (Figure 2.1) agree to the first five digits and are slightly greater than 23. The next algebraically smallest eigenvalues are slightly greater than 58. Suppose that we are endowed with this information a priori, making the preconditioning strategy in Section 3.3.1 attractive. To use said strategy, we form an

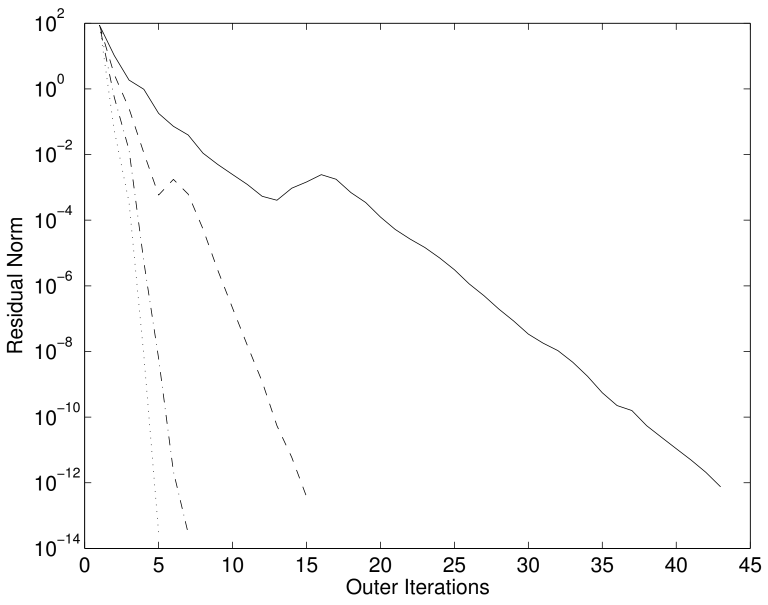


Figure 3.4: Convergence histories of the smallest Ritz value computed by BlockVarIIp(2,4) with $K - 24M \approx LDL^*$ preconditioners with various drop tolerances τ. $\tau = 1\mathrm{e}-01$ (solid), $\tau = 1\mathrm{e}-02$ (dashed), $\tau = 1\mathrm{e}-03$ (dash-dotted), $\tau = 1\mathrm{e}-04$ (dotted).

incomplete LDL^* factorization of $K - \theta M$, where θ lies between 23 and 58, the bounds of our eigenvalues. We choose, for the sake of this example $\theta = 24$. Here the incomplete factorization is computed via MATLAB's `luinc` function applied without pivoting and with diagonal replacement. That is, zero entries on the diagonal of the U factor get replaced with the local drop tolerance in an attempt to avoid constructing a singular factor. Figure 3.4 shows the convergence histories of the smallest Ritz value for preconditioners described above with different drop tolerances τ. The solid line is the history for the preconditioned iteration where $\tau = 1\mathrm{e}-01$, the dashed is for $\tau = 1\mathrm{e}-02$, the dash-dotted is for $\tau = 1\mathrm{e}-03$, and the dotted is for $\tau = 1\mathrm{e}-04$. The experiments were run using BlockVarIIp(2,4). Note that even though the lower bound on the relative spectral gap in this case should be zero, since generation of the complete LDL^* factorization of $K - 24M$ has $D = \mathrm{diag}(-1, -1, 1, 1, \ldots, 1)$, we still get excellent convergence when we use block size two.

For comparison, Figure 3.5 shows the convergence history for the smallest Ritz value preconditioned by the incomplete LDL^* factorization of $K - 24M$ (solid line) and the convergence history for the smallest Ritz value preconditioned by the incomplete LDL^* factorization of K (dotted line). Both factorizations were computed with a drop tolerance of $1\mathrm{e}-01$ and again, BlockVarIIp(2,4) was used. Note how similar the two convergence curves are. The advantage to using a preconditioner based on K

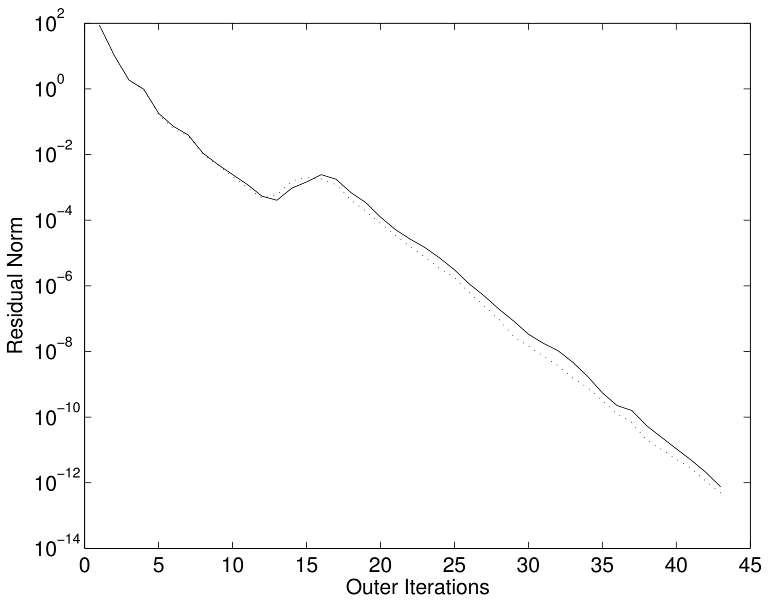


Figure 3.5: Convergence histories of the smallest Ritz value computed by Block-VarIIp(2,4) with incomplete LDL^* factorization preconditioners with two different drop tolerances. $K - 24M \approx LDL^*, \tau = 1\text{e}-01$ (solid), $K \approx LDL^*, \tau = 1\text{e}-01$ (dotted).

alone is that we do not have to compute an approximation to the smallest eigenvalue before developing a suitably good preconditioner to use. This example shows that preconditioners based on a lower bound of the spectrum may be equally competitive.

When used as a black-box, the EIGIFP package [40] includes a strategy which switches preconditioning on once it seems that a reasonable approximation to λ_1 has been determined. We ran this example through the EIGIFP software and allowed the program to switch to preconditioning once a suitable shift was found. Indeed, the package switched to performing preconditioned iterations when the Ritz value θ was around 24, and the preconditioner developed was based on the incomplete LDL^* factorization of $K - \theta M$ with drop tolerance 1e–01. The algorithm was executed with a fixed number of inner iterations, 4, and the convergence history of the smallest Ritz value is presented in Figure 3.6. In this example, preconditioning is switched on at outer iteration 14. There is a notable increase in the rate of convergence around outer iteration 35 or so, and then the sawtooth behavior begins. This sawtooth behavior is caused by the tight clustering of λ_1 and λ_2, the fact that the method implemented in EIGIFP is the single-vector algorithm, Algorithm 2.2, and the inclusion of the previous Ritz vector as described in Section 2.4.2. Although the convergence is markedly faster, than that seen in Figure 2.4, however, the sawtooth behavior seems to imply that preconditioners may not be able to rescue us from the need for a block algorithm.

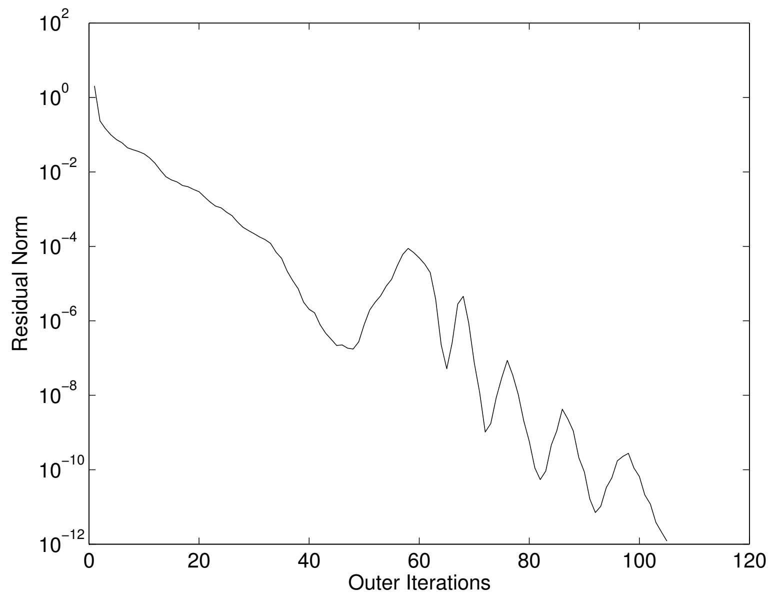


Figure 3.6: Convergence history of EIGIFP with automatic preconditioning.

Indeed, applying BlockVarII(1,4) with the $K - 24M \approx LDL^*$ preconditioner constructed with drop tolerances $\tau = 1\mathrm{e}{-}01, 1\mathrm{e}{-}02, 1\mathrm{e}{-}03$ fails to give convergence after 300 outer iterations. Using $\tau = 1\mathrm{e}{-}04$ as the drop tolerance, however, gives excellent convergence. See Figure 3.7. The preconditioner developed with drop tolerance $\tau = 1\mathrm{e}{-}04$ is an order of magnitude more accurate than the preconditioner constructed with drop tolerance $\tau = 1\mathrm{e}{-}03$, however, both result in $D = I$, instead of $D = \mathrm{diag}(-1, -1, 1, 1, \ldots, 1)$ as they should. Notably, when $\tau = 1\mathrm{e}{-}05$ is used as the drop tolerance, the correct D is realized.

3.4.3 Example: Comparing Preconditioners and Methods

Here we consider using four different kinds of preconditioners in computing the smallest eigenvalue of the BCSST13 generalized fluid flow problem from the Harwell-Boeing collection [16]. This problem has a non-diagonal semidefinite mass matrix associated with it and the positive definite stiffness matrix is rather poorly conditioned with a two norm condition estimate of 4.6e+10. As the stiffness matrix is positive definite, we have $\lambda_0 = 0$ as a lower bound for the spectrum of (K, M), and as such, we will construct incomplete factorizations of $K \approx LL^*$ and use those factors as the preconditioners. The four types of preconditioners we construct are RIF, SAINV, ICT (incomplete Cholesky with threshold dropping), and ICAJ (incomplete Cholesky with Ajiz-Jennings dropping rules). We consider performance of the preconditioners with respect to drop tolerances, and detail measures of accuracy and stability of

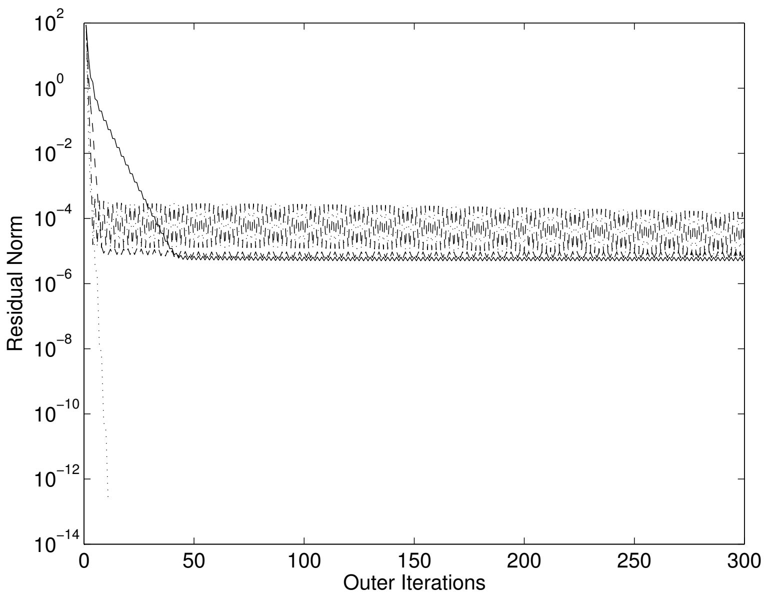


Figure 3.7: Convergence histories of the smallest Ritz value computed by Block-VarII(1,4) with $K - 24M \approx LDL^*$ preconditioners with various drop tolerances τ. $\tau = 1\text{e}-01$ (solid), $\tau = 1\text{e}-02$ (dashed), $\tau = 1\text{e}-03$ (dash-dotted), $\tau = 1\text{e}-04$ (dotted).

the factorizations. We remark that it is not appropriate to measure accuracy of the SAINV preconditioner, as it is a sparse approximate inverse. In this example, we take $m = 12$ and we terminate when the residual norm falls below 1e−07, or when 250 outer iterations are reached.

Table 3.3 details the number of matrix vector multiplies and the number of preconditioner applications required for the algorithm to reach termination. The final residual is also reported. Pertinent preconditioner statistics, such as accuracy, stability, and the number of nonzeros in the factors are reported in Table 3.4.

Some comments about the data in the tables are in order. We note that RIF actually requires two tolerances, a drop tolerance and a post-filtration tolerance. For the experiments here, the tolerances are taken to be the same.

For this problem, it appears that the best preconditioner is the RIF preconditioner. In all cases, the algorithm preconditioned with the RIF preconditioner requires the fewest matvecs to converge. In all fairness, however, it is also one of the densest preconditioners constructed. Recalling that K here has only about 83000 entries, we see that the RIF preconditioner constructed with a drop tolerance of 1e−01 has roughly twice as many entries as K. The RIF preconditioner also enjoys the smallest stability estimate, with the exception of the ICAJ preconditioner, which has an extremely small stability estimate in all cases. Why then, does RIF outperform the

Method	τ	matvecs	preapps	final residual
RIF	1e−01	1621	721	8.357e−07
	1e−02	757	337	6.346e−07
	1e−03	244	109	2.661e−07
	1e−04	136	61	3.260e−09
SAINV	1e−01	4051	1801	7.453e−07
	1e−02	1432	637	8.180e−07
	1e−03	460	205	2.903e−07
	1e−04	244	109	2.584e−07
ICT	1e−01	6748	3000	2.132e+05
	1e−02	6748	3000	6.288e+01
	1e−03	946	421	2.760e−07
	1e−04	352	157	6.041e−07
ICAJ	1e−01	6589	2929	7.880e−07
	1e−02	2647	1177	7.993e−07
	1e−03	1594	709	8.198e−07
	1e−04	568	253	8.029e−07

Table 3.3: Convergence characteristics for BlockVarIIp(1,12) applied to BCSST13 with various preconditioners.

Method	τ	nnz(L)	$\|K - LL^*\|_F / \|K\|_F$	$\|I - L^{-1}KL^{-*}\|_F$
RIF	1e−01	168344	3.028e−02	1.228e+01
	1e−02	318629	2.352e−03	7.320e+00
	1e−03	399817	1.380e−04	3.764e+00
	1e−04	421398	1.288e−05	1.827e+00
SAINV	1e−01	60656	—	2.426e+01
	1e−02	319735	—	1.583e+01
	1e−03	699084	—	8.529e+00
	1e−04	1093534	—	4.627e+00
ICT	1e−01	16409	2.465e−01	3.444e+14
	1e−02	44860	1.330e−01	3.933e+08
	1e−03	83104	2.137e−02	2.116e+01
	1e−04	143025	3.492e−03	8.840e+00
ICAJ	1e−01	9192	7.294e−01	2.340e−05
	1e−02	31617	9.379e−02	3.640e−05
	1e−03	79900	1.570e−02	3.299e−05
	1e−04	165650	3.064e−03	1.649e−05

Table 3.4: Preconditioner statistics of various incomplete factorizations of K from BCSST13.

ICAJ preconditioner in terms of matvecs across the board? We conjecture that this is due to the construction of the ICAJ preconditioner in that if L is the ICAJ factor, we have $LL^* = K + C$ where C is a positive semidefinite matrix [1],[5]. Computing the Cholesky factor of $K + C$ may be not so far from computing the Cholesky factor of $K - \lambda_0 M$ where $\lambda_0 < 0$ is a looser bound on the spectrum of (K, M). As we saw in Section 3.3.4, in the presence of perfect preconditioning, an factor of K will outperform a factor of $K - \lambda_0 M$, when $\lambda_0 < 0$. In this case, then, it may be that a very stable factorization of $K - \lambda_0 M$, $\lambda_0 < 0$ may not be as effective as a less stable factorization of K in terms of matrix-vector multiplications.

We now give a limited comparison between BlockVarIIp(1,m) and JDQZ, a Jacobi-Davidson code obtained from Gerard Sleijpen's web page, implementing the algorithm found in [17]. For the purposes of comparison, we endowed both methods with the same initial vector and the same preconditioner. Giving the same preconditioner to Jacobi-Davidson makes sense in that the preconditioner there is used in solution of the correction equation, and as such, should be an approximation to $K - \sigma M$ where σ is the target for eigenvalue location. The target we provide JDQZ is $\sigma = 0$, and we set $jmin$ to be the same as m, and $jmax = m + 5$, the default value in the code. We also set the test space to be the search space.

In comparing BlockVarIIp with JDQZ, we applied each algorithm to the problem using the RIF preconditioner with drop tolerance $\tau = 1\text{e}{-02}$. Figure 3.8 shows the convergence history of the smallest Ritz value obtained by BlockVarIIp(1,12) with RIF as the solid line and the convergence history of the smallest Ritz value computed by JDQZ with an RIF preconditioner as the dotted line. Some comments are certainly in order here. For JDQZ, each outer iteration corresponds to using preconditioned GMRES to solve a correction equation. Here, we have forced GMRES to take 12 steps, in an effort to have a legitimate baseline. JDQZ as executed here, required 1796 matvecs and 1932 applications of the preconditioner, where BlockVarIIp(1,12) required only 757 matvecs and 337 preconditioner applications. We also stress that we count each application of K *and* M as separate matvecs, where it appears that the count in JDQZ combines the actions of K and M.

3.4.4 Example: An Algebraic Multigrid Preconditioner

As our last example, we include an algebraic multigrid (AMG) preconditioner. The code implementing AMG used here is found in the AMGToolBox [58] and was obtained from Menno Verbeek and Jane Cullum. The package implements the Ruge-Stüben AMG algorithm to construct coarse levels and apply V-cycles. We apply the solver as a preconditioner by calling the setup routine to construct an AMG solver for $K - \lambda_0 M$, where we are interested in the algebraically smallest eigenvalues of (K, M) and λ_0 is a lower bound for the eigenvalues of interest. We then precondition

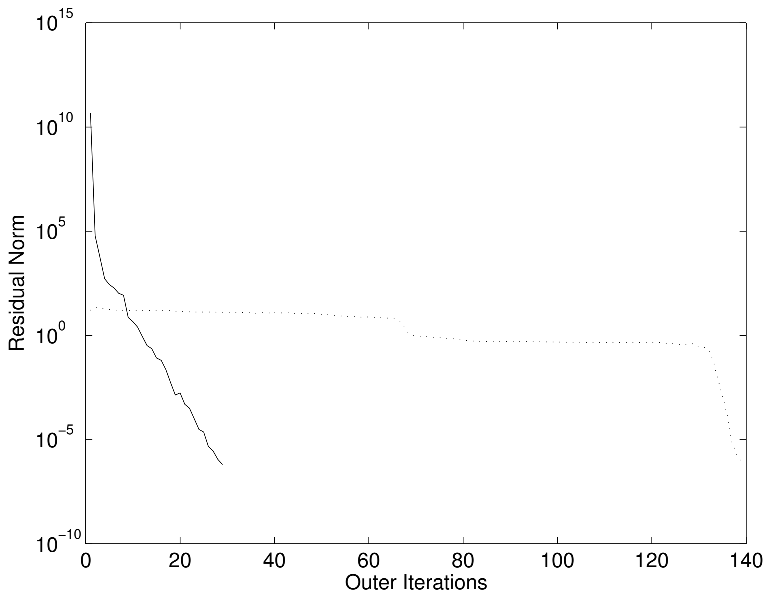


Figure 3.8: Convergence histories of the smallest Ritz value computed by BlockVarIIp(1,12) (solid), and JDQZ (dotted). Both iterations utilized the RIF preconditioner.

the block Arnoldi-like process by applying a fixed number of V-cycles to the newly computed residual block. This is in place of the application of $L^{-*}L^{-1}$ in line two of Algorithm 3.1. As an example, we turn again to the finite element model of the Laplacian on the barbell shaped domain. We seek the two algebraically smallest eigenpairs and declare convergence once we have achieved a residual norm smaller than 1e−12. Again, four inner iterations are used. Figure 3.9 presents the convergence histories of the smallest Ritz value obtained from BlockVarIIp(2,4) preconditioned by AMG with one V-cycle. The AMG preconditioned iteration required only 134 matvecs to converge, compared to the 3192 matvecs required by BlockVarIIp(2,4) without preconditioning. This did come at the cost of a setup phase, however, this cost is negligible compared to the speed-up experienced by the algorithm. We stress that this example serves only to illustrate the viability of preconditioning the algorithms discussed in Chapter 2 with a multigrid method. We also note that the discretized Laplacian is precisely the problem for which multigrid methods were developed, and it is not too surprising that we get this kind of behavior using the AMGToolBox solver as a black box as we did. We do remark that for other problems, finding a good coarsening algorithm may be more difficult and not yield the kind of convergence we see here. See [2], [36] for more discussions concerning application of multigrid as a preconditioner for eigensolvers.

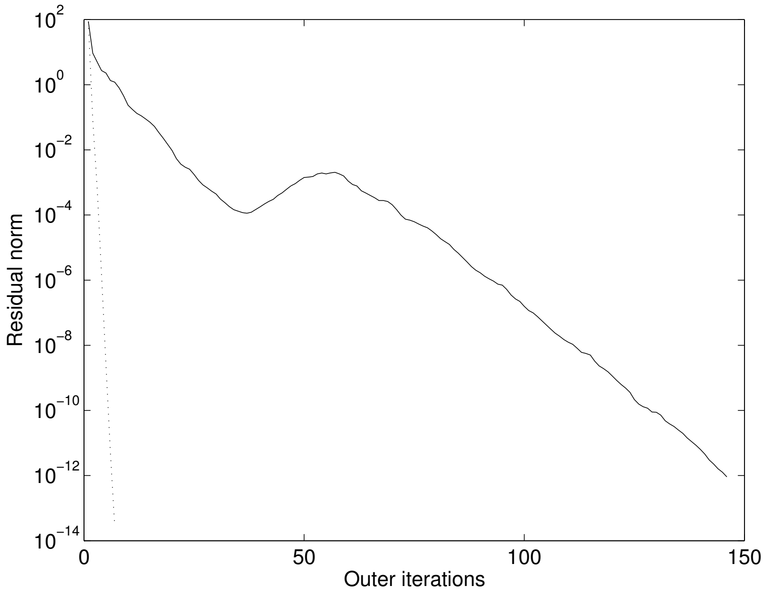


Figure 3.9: Convergence of the smallest Ritz value computed by BlockVarIIp(2,4) with no preconditioner (solid) and preconditioned by AMG with one V-cycle (dotted).

Chapter 4

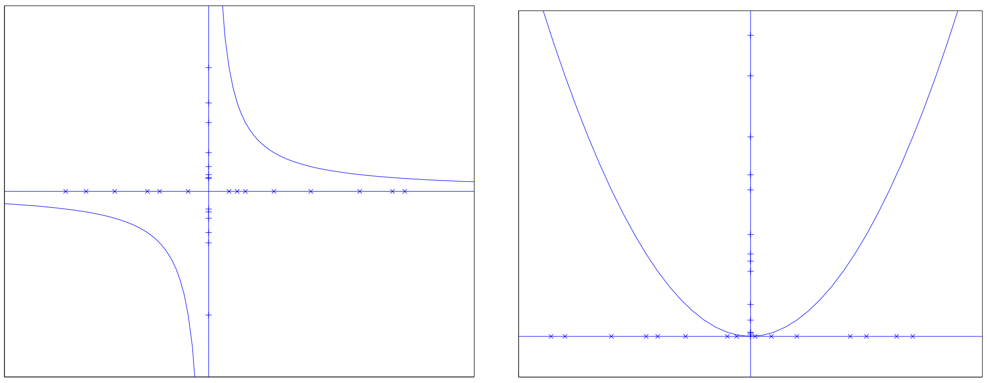
The Interior Eigenvalue Problem

The methods described in Chapter 2 are suitable for computing the extreme eigenvalues; i.e. those at either end of the spectrum. In some applications, however, it is necessary to compute a few eigenvalues in the interior of the spectrum.

In general, computation of interior eigenvalues is very difficult, as good approximations to them from small subspaces are hard to come by. From the Courant-Fischer principle, it is clear that the algebraically smallest eigenvalue is the global minimum of the Rayleigh quotient, and the algebraically largest eigenvalue is the global maximum. Interior eigenvalues, on the other hand are minima or maxima over some lower dimensional subspace subject to a number of constraints. For this reason, they are typically more difficult to locate, unless they are made to be extreme eigenvalues by a suitable transformation. This is the heart of the shift-and-invert Lanczos method, which is currently the only truly viable method for finding interior eigenvalues.

There exist other approaches based on harmonic Ritz approximations developed for the symmetric eigenvalue problem in [41] and investigated by many others. The algorithm does not require applications of an inverse anywhere, but convergence can be very slow. The JDQZ method also includes a harmonic projection, however the same troubles are experienced there. Convergence is usually slow, if it happens at all. For a discussion concerning the convergence of harmonic Ritz pairs, see [30].

This chapter presents two approaches which transform the interior eigenvalue problem to an extreme eigenvalue problem to which we may apply the algorithms of Chapter 2. We remark that due to the transformation we consider the algorithm will no longer be inverse free, but will rely only on the ability to apply B^{-1}. As B is positive definite, applying B^{-1} is frequently a much simpler task than applying $(A - \mu B)^{-1}$ when μ lies in the interior of the spectrum.



(a) Spectrum of (C, D) on the vertical axis; spectrum of (A, B) on the horizontal axis

(b) Spectrum of (D, B) on the vertical axis; spectrum of (A, B) on the horizontal axis

Figure 4.1: Spectra of the Transformed Problems

4.1 Recasting the problem

We consider the problem of locating the eigenvalues of (A, B) nearest to some target $\mu \in \mathbb{R}$. Here, it is assumed that μ will be supplied to the algorithm by the user. Consider the two operators

$$C := A - \mu B \quad \text{and} \quad D := (A - \mu B)B^{-1}(A - \mu B) \equiv CB^{-1}C \tag{4.1}$$

and the pencils (C, D) and (D, B). We note that since B is positive definite, it follows that D is as well, and the algorithms of Chapter 2 may be applied to compute the extreme eigenvalues of the pencils (C, D) and (D, B). If (λ, x) is an eigenpair of (A, B), it is easy to see that $(\hat{\lambda}, x)$ is an eigenpair of (C, D) where

$$\hat{\lambda} = \frac{1}{\lambda - \mu}. \tag{4.2}$$

Thus the eigenvalues of (A, B) immediately to the left of μ correspond to the algebraically smallest eigenvalues of (C, D). Indeed, if μ lies in the interior of the spectrum, the matrix C will be indefinite, and the algebraically smallest eigenvalues of (C, D) will be negative. On the other hand, the eigenvalues of (A, B) immediately larger than μ will be the algebraically largest eigenvalues of (C, D) and will be positive. Similarly, an eigenpair (λ, x) of (A, B), corresponds to an eigenpair $(\tilde{\lambda}, x)$ of (D, B) via the relation

$$\tilde{\lambda} = (\lambda - \mu)^2. \tag{4.3}$$

Here the eigenvalues of (A, B) nearest to μ correspond to the algebraically smallest eigenvalues of (D, B), regardless of which side of μ they appear on. Due to the positive definiteness of D, all of the eigenvalues of (D, B) are in fact positive. In view of the discussion in Chapter 3, this may be more favorable for preconditioning. We note that the use of the pencil (D, B) in computation of interior modes is similar to the folded spectrum approach of Wang and Zunger found in, for example, [11] and [61]. Visualizations of sample distributions of eigenvalues of (C, D) and (D, B) are found in Figure 4.1.

Both of these formulations have their advantages and disadvantages, so which is the one to choose? In view of the discussion in Section 3.3, constructing effective preconditioners for (D, B) will be, in general, easier than constructing preconditioners for (C, D). That is, of course, unless a good lower bound for the spectrum (C, D) is known. If this method is being used in a black-box fashion, this is highly unlikely. Even if a good lower bound λ_0 for the spectrum of (C, D) is known, it may be difficult to form incomplete factorizations of $C - \lambda_0 D$ without explicitly forming D.

If μ is close to some eigenvalues of (A, B), then the matrix D may be highly ill-conditioned, making it very difficult to construct accurate incomplete factorizations. We point out again that it is stability which is of prime importance here, and accuracy does not matter nearly as much for the quality of convergence. On the other hand, when μ is close to some eigenvalues of (A, B), the eigenvalues of (C, D) show excellent separation, inviting us to consider the preconditioning scheme discussed in Section 3.3.3. This may not be as attractive as it seems, however, since the eigenvalues on either side of μ will get well separated to *opposite* ends of the spectrum resulting in an extremely large spread $\lambda_n(C, D) - \lambda_1(C, D)$, and therefore a small relative spectral gap, even though the quantity $\lambda_2(C, D) - \lambda_1(C, D)$ may be large. For these reasons, we will favor the use of the pencil (D, B) and this will be considered throughout the remainder of this chapter.

4.2 Computing B^{-1}-orthogonal QR factorizations

In constructing factorizations of D, we wish to avoid the explicit formation of $D = (A - \mu B)B^{-1}(A - \mu B)$, and we seek to exploit the normal equations structure of the matrix. Indeed, if $B = I$, then $D = (A - \mu B)(A - \mu B)$ and we may compute an LQ factorization of $A - \mu B$ and thereby implicitly obtain the Cholesky factor of D. On the other hand, if $B \neq I$ and we compute $A - \mu B = QR$ where Q is a B^{-1}-orthonormal matrix, that is, $Q^* B^{-1} Q = I$, then

$$D = (A - \mu B)B^{-1}(A - \mu B) = R^* Q^* B^{-1} Q R = R^* R \tag{4.4}$$

and we have implicitly computed a Cholesky factor of D.

One approach to constructing B-orthogonal QR factorizations of a matrix is to use the Gram-Schmidt process with-respect to the B inner product. This process is straightforward to implement and will produce the factorization we seek at the cost of storing the entire matrix Q, which for our purposes will be unnecessary. Moreover, the inner product we would like to work in is the B^{-1} inner product, requiring many applications of B^{-1}. Also, as discussed in [4], incomplete Gram-Schmidt frequently produces poor-quality preconditioners. We will instead build B^{-1}-orthogonal factorizations via B^{-1}-unitary Householder-like matrices. We recall briefly that B^{-1}-unitary matrices are those which preserve the B^{-1} inner product. Thus Q is B^{-1}-unitary if $Q^*B^{-1}Q = B^{-1}$. As we will see, the connection between B-unitary and B^{-1}-unitary matrices is very strong, and thus we first discuss the B-unitary tools required.

4.2.1 B-unitary matrices

Some elementary facts concerning B-unitary matrices follow, which are included for completeness. We will specifically exploit property 3 to build B^{-1}-unitary QR factorizations, without having to rely on the ability to apply B^{-1} in its entirety. The proof of the proposition is elementary and is therefore omitted.

Proposition 4.2.1. *Suppose Q and V are B-unitary. Then*

1. *QV is B-unitary.*
2. *Q is nonsingular, Q^{-1} is B-unitary, and $Q^{-1} = B^{-1}Q^*B$*
3. *Q^* is B^{-1}-unitary.*

That is, the set of B-unitary matrices form a group with respect to matrix multiplication.

Motivated by the Householder reflectors with respect to the usual inner product, we make the following definition and note some immediate results.

Definition 4.2.2. *Suppose B is a symmetric positive-definite matrix. Let $Q = I - 2P$ where $P := u(u^*Bu)^{-1}u^*B$, for some vector $u \in \mathbb{R}^n$. We call Q a Householder-like transformation.*

Proposition 4.2.3. *Let Q be a Householder-like transformation.*

1. *Q is B-unitary.*
2. *Given x, y two vectors such that $\|x\|_B = \|y\|_B$, then choosing $u = x - y$ defines Q so that $Qx = y$.*
3. *Given x, y two vectors such that $\|x\|_{B^{-1}} = \|y\|_{B^{-1}}$, then choosing $u = B^{-1}(x-y)$ defines Q so that $Q^*x = y$.*

Proof. Note first that $P = u(u^*Bu)^{-1}u^*B$ is a projector onto $\operatorname{span}\{u\}$. Furthermore

$$BP = Bu(u^*Bu)^{-1}u^*B = P^*B$$

indicating that P is orthogonal with respect the B-inner product. Simple calculations following from the properties of the projector P reveal that Q is indeed B-unitary.

$$\begin{aligned} Q^*BQ &= (I-2P)^*B(I-2P) \\ &= B - 2P^*B - 2BP + 4P^*BP \\ &= B - 4BP + 4BP \\ &= B \end{aligned}$$

Now suppose that x and y are two vectors such that $\|x\|_B = \|y\|_B$. With $u = x-y$, we see that

$$Qx = x - 2u(u^*Bu)^{-1}u^*Bx = x - \frac{2u^*Bx}{u^*Bu}u.$$

Now

$$u^*Bu = (x-y)^*B(x-y) = x^*Bx - 2x^*By + y^*By = 2\left(x^*Bx - y^*Bx\right) = 2u^*Bx$$

and therefore $Qx = x - (x-y) = y$.

Similarly, if x and y are two vectors such that $\|x\|_{B^{-1}} = \|y\|_{B^{-1}}$, then

$$Q^*x = x - 2Bu(u^*Bu)^{-1}u^*x = x - \frac{2u^*x}{u^*Bu}Bu.$$

Now

$$\begin{aligned} u^*Bu &= (B^{-1}(x-y))^*BB^{-1}(x-y) \\ &= x^*B^{-1}x - 2x^*B^{-1}y + y^*B^{-1}y \\ &= 2\left(x^*B^{-1}x - y^*B^{-1}x\right) = 2u^*x \end{aligned}$$

and so $Q^*x = x - B(B^{-1}(x-y)) = x - (x-y) = y$. □

Definition 4.2.2 and Proposition 4.2.3 are both very specific instances of definitions and results found in [37] and [38] which develop Householder-like reflectors, among other transformations, with respect to not only inner products, but rather more general scalar products. It bears mentioning that we are concerned here with the choice of y in order to preserve the upper-triangular structure of R that we covet—a topic which is not explicitly addressed in either [37] or [38].

4.2.2 Constructing QR factorizations via Householder-like reflectors

We recall that the construction of a QR factorization of a matrix A with respect to the usual Euclidean inner product, requires the choice of $u = x - y$ at each step j of the algorithm in order to introduce zeros in the $j+1$ through n positions of column j in the matrix R, where R is initialized to be A. At step j of the construction of the QR factorization, $x = \begin{pmatrix} 0 & \cdots & 0 & r_{j,j} & r_{j+1,j} & \cdots & r_{n,j} \end{pmatrix}^*$ and y is chosen to be $\pm\|x\|e_j$. We refer to these choices of x and y as the usual choices for x and y. With the usual choices of x and y we have $u = \begin{pmatrix} 0 & \cdots & 0 & u_j & \cdots & u_n \end{pmatrix}^*$ with associated Householder reflector

$$\begin{aligned} I - 2u(u^*u)^{-1}u^* &= \begin{pmatrix} I_{j-1} & \\ & I_{n-j+1} \end{pmatrix} - 2(u^*u)^{-1} \begin{pmatrix} 0 & 0 \\ 0 & \hat{u}\hat{u}^* \end{pmatrix} \\ &= \begin{pmatrix} I_{j-1} & \\ & I_{n-j+1} - 2\hat{u}(\hat{u}^*\hat{u})^{-1}\hat{u}^* \end{pmatrix} \end{aligned}$$

where $\hat{u} = \begin{pmatrix} u_j & \cdots & u_n \end{pmatrix}^*$. It is plain to see that application of this Householder matrix preserves the upper-triangular structure already formed in the previous $j-1$ columns of R.

When using B-unitary Householder-like transformations as defined by definition 4.2.2, however, we do not necessarily get this same structure preservation by choosing x and y in the usual manner. As Proposition 4.2.3 shows, $u = x - y$ will define a B-unitary Householder-like transformation Q such that $Qx = y$, and

$$u = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ u_j \\ \vdots \\ u_n \end{pmatrix} = \begin{pmatrix} 0 \\ \hat{u} \end{pmatrix}$$

when we choose x and y in the usual manner. At the j^{th} step of the algorithm, this transformation has the structure

$$\begin{aligned} Q = I - 2u(u^*Bu)^{-1}u^*B &= \begin{pmatrix} I_{j-1} & \\ & I_{n-j+1} \end{pmatrix} - \beta \begin{pmatrix} 0 \\ \hat{u} \end{pmatrix} \begin{pmatrix} z_1^* & z_2^* \end{pmatrix} \\ &= \begin{pmatrix} I_{j-1} & 0 \\ -\beta\hat{u}z_1^* & I_{n-j+1} - \beta\hat{u}z_2^* \end{pmatrix} \end{aligned}$$

where $\beta = 2(u^*Bu)^{-1}$ and $z = \begin{pmatrix} z_1^* & z_2^* \end{pmatrix}^* = Bu$. It should be clear that premultiplication by such a Q will destroy the structure of the leading upper-triangular part of R. A first remedy to this situation lies in a more shrewd choice of y which, though expensive, guarantees a transformation which will not destroy the structure of the first $j-1$ columns of R. On the other hand, application of Q^*, a B^{-1}-unitary operator, will not destroy the upper triangular structure of the first $j-1$ columns of R, but the usual choice for y may not suffice, as we must guarantee that $u \in \operatorname{span}\{e_j, \dots, e_n\}$ to maintain the structure.

Although the typical choice of y is $y = \pm\|x\|e_j$, this may not allow for the preservation of the upper-triangular structure of R as discussed above. Instead, at the j^{th} step of a QR factorization algorithm, we may choose $y \in \operatorname{span}\{e_1, e_2, \dots, e_j\}$, subject to the constraint $\|y\| = \|x\|$. In the following, we discuss how to choose y to develop a B-unitary or B^{-1}-unitary QR factorization of a nonsingular matrix. This discussion will not be exactly pertinent to our goal of constructing a B^{-1}-orthogonal QR factorization, however, we include it for completeness, and believe it to be of interest since we have not found a method like this in the literature.

4.2.3 Towards B-unitary QR factorizations

Suppose first we wish to construct $A = QR$ with Q being B-unitary for a nonsingular A. The correct choice of u is $u = x - y$ as noted above, however, we wish that $Bu = z \in \operatorname{span}\{e_j, \dots, e_n\}$ while $y \in \operatorname{span}\{e_1, \dots, e_j\}$, with $Bu = Bx - By$. Choosing u in this way will preserve the upper-triangular structure in the first $j-1$ columns while reducing the j^{th} column (x) to be an element of $\operatorname{span}\{e_1, \dots, e_j\}$, namely y. Now, let $y = \begin{pmatrix} \hat{y}^* & 0^* \end{pmatrix}^*$ where $\hat{y} \in \mathbb{R}^j$ and let

$$B_{j-1} = \begin{pmatrix} b_{1,1} & \cdots & b_{1,j-1} \\ \vdots & & \vdots \\ b_{j-1,1} & \cdots & b_{j-1,j-1} \end{pmatrix} \qquad \hat{B}_{j-1} = \begin{pmatrix} b_{1,1} & \cdots & b_{1,j-1} & b_{1,j} \\ \vdots & & \vdots & \vdots \\ b_{j-1,1} & \cdots & b_{j-1,j-1} & b_{j-1,j} \end{pmatrix}$$

Finally, let $d = Bx$ and denote the first $j-1$ entries of d by the vector $\hat{d}$. Now to construct u as desired, we require that

$$\hat{B}_{j-1}\hat{y} = \hat{d}$$

To this end, let $\hat{y} = \hat{v} + \alpha\hat{w}$ where

$$\hat{B}_{j-1}\hat{v} = \hat{d} \qquad \hat{B}_{j-1}\hat{w} = 0$$

and α is chosen so that $\|y\|_B = \|x\|_B$ as required by Proposition 4.2.3. Since B is positive definite, the leading principal submatrices, such as B_{j-1} are nonsingular, and therefore the choice of $\hat{y}$, and therefore y, is unique up to choice of α. That is, we let

$$\hat{v} = \begin{pmatrix} \tilde{v} \\ 0 \end{pmatrix}, \quad \tilde{v} = B_{j-1}^{-1}\hat{d} \in \mathbb{R}^{j-1}$$

and

$$\hat{w} = \begin{pmatrix} \tilde{w} \\ -1 \end{pmatrix}, \quad \tilde{w} = B_{j-1}^{-1}\hat{b}_j \in \mathbb{R}^{j-1}, \quad \hat{b}_j = \begin{pmatrix} b_{1,j} \\ \vdots \\ b_{j-1,j} \end{pmatrix}.$$

Now with $\hat{v}$ and $\hat{w}$ defined above, we define

$$y = v + \alpha w = \begin{pmatrix} \hat{v} \\ 0 \end{pmatrix} + \alpha \begin{pmatrix} \hat{w} \\ 0 \end{pmatrix}$$

Note that

$$\begin{aligned} y^* By &= (v + \alpha w)^* B (v + \alpha w) \\ &= v^* Bv + 2\alpha w^* Bv + \alpha^2 w^* Bw \end{aligned}$$

To ensure that $\|y\|_B = \|x\|_B$, we choose α to be a root of the quadratic

$$(w^* Bw)\alpha^2 + (2w^* Bv)\alpha + (v^* Bv - x^* Bx) \tag{4.5}$$

This quadratic is never degenerate, as $w \neq 0$ by construction. Moreover, when A is nonsingular with real entries, then this quadratic has real roots. We state this as the following lemma.

Lemma 4.2.4. *Under the assumption that A is nonsingular, the quadratic (4.5) has two real roots, one positive and one negative.*

Proof. The result follows if the quantity $(w^* Bw)(v^* Bv - x^* Bx) < 0$. Since B is positive definite, all we must show is that $(v^* Bv - x^* Bx) < 0$. Now let B be partitioned as

$$B = \begin{pmatrix} B_{j-1} & B_{12} \\ B_{12}^* & B_{n-j+1} \end{pmatrix}$$

Partitioning $x = \begin{pmatrix} \hat{x}^* & \tilde{x}^* \end{pmatrix}^*$ so that $\hat{d} = B_{j-1}\hat{x} + B_{12}\tilde{x}$ we get that

$$v = \begin{pmatrix} \tilde{v} \\ 0 \end{pmatrix} = \begin{pmatrix} B_{j-1}^{-1}\hat{d} \\ 0 \end{pmatrix} = \begin{pmatrix} \hat{x} + B_{j-1}^{-1}B_{12}\tilde{x} \\ 0 \end{pmatrix}$$

With this formula, we note that

$$\begin{aligned} v^* Bv &= \begin{pmatrix} \hat{x} + B_{j-1}^{-1}B_{12}\tilde{x} \\ 0 \end{pmatrix}^* \begin{pmatrix} B_{j-1} & B_{12} \\ B_{12}^* & B_{n-j+1} \end{pmatrix} \begin{pmatrix} \hat{x} + B_{j-1}^{-1}B_{12}\tilde{x} \\ 0 \end{pmatrix} \\ &= \hat{x}^* B_{j-1}\hat{x} + \tilde{x}^* B_{12}^*\hat{x} + \tilde{x}^* B_{12}\hat{x} + \tilde{x}^* B_{12}^* B_{j-1}^{-1} B_{12}\tilde{x} \\ &= \begin{pmatrix} \hat{x} \\ \tilde{x} \end{pmatrix}^* \begin{pmatrix} B_{j-1} & B_{12} \\ B_{12}^* & B_{12}^* B_{j-1}^{-1} B_{12} \end{pmatrix} \begin{pmatrix} \hat{x} \\ \tilde{x} \end{pmatrix} \end{aligned}$$

Therefore,

$$\begin{aligned} v^*Bv - x^*Bx &= \begin{pmatrix} \hat{x} \\ \tilde{x} \end{pmatrix}^* \begin{pmatrix} 0 & 0 \\ 0 & B_{12}^* B_{j-1}^{-1} B_{12} - B_{n-j+1} \end{pmatrix} \begin{pmatrix} \hat{x} \\ \tilde{x} \end{pmatrix} \\ &= \tilde{x}^*(B_{12}^* B_{j-1}^{-1} B_{12} - B_{n-j+1})\tilde{x} \end{aligned}$$

Noting that

$$\begin{pmatrix} B_{j-1} & 0 \\ 0 & B_{n-j+1} - B_{12}^* B_{j-1}^{-1} B_{12} \end{pmatrix} = \\ \begin{pmatrix} I & 0 \\ -B_{12}^* B_{j-1}^{-1} & I \end{pmatrix} \begin{pmatrix} B_{j-1} & B_{12} \\ B_{12}^* & B_{n-j+1} \end{pmatrix} \begin{pmatrix} I & -B_{j-1}^{-1} B_{12} \\ 0 & I \end{pmatrix}$$

it follows that $B_{n-j+1} - B_{12}^* B_{j-1}^{-1} B_{12}$ is positive definite and thus

$$v^*Bv - x^*Bx = -\tilde{x}^*(B_{n-j+1} - B_{12}^* B_{j-1}^{-1} B_{12})\tilde{x} < 0$$

provided that $\tilde{x} \neq 0$. Our assumption of nonsingularity of A guarantees for all $j > 1$ that $\tilde{x} \neq 0$. $\square$

Development of an algorithm for the construction of a B-unitary QR factorization of a matrix A now follows immediately by iteratively applying the process described above to reduce the j^{th} column of A to be a vector which is an element of span$\{e_1, \ldots, e_j\}$. As the goal of our work is not the construction of B-unitary QR factorizations, we omit a detailed listing of the algorithm here.

4.2.4 Towards B^{-1}-unitary QR factorizations

Now when attempting to form $A = QR$ with Q being B^{-1}-unitary, when reducing the j^{th} column of A, we wish to form $u = B^{-1}(x - y) \in \text{span}\{e_j, \ldots, e_n\}$. Furthermore, we require that $y \in \text{span}\{e_1, \ldots, e_j\}$ and so the conditions we impose are

$$y = x - z, \quad y \in \text{span}\{e_1, \ldots, e_j\}, \quad z = Bu \in \text{span}\{b_j, \ldots, b_n\}, \quad \|y\|_{B^{-1}} = \|x\|_{B^{-1}}.$$

These conditions stem from the desire to maintain the upper-triangular structure generated by reduction of the previous columns, while reducing the current column to be an element of span$\{e_1, \ldots, e_j\}$. Denoting $\hat{x} = \begin{pmatrix} x_{j+1} & \cdots & x_n \end{pmatrix}^*$, $\hat{u} = \begin{pmatrix} u_j & \cdots & u_n \end{pmatrix}^*$, and

$$B_{n-j} = \begin{pmatrix} b_{j+1,j+1} & \cdots & b_{j+1,n} \\ \vdots & & \vdots \\ b_{n,j+1} & \cdots & b_{n,n} \end{pmatrix} \qquad \hat{B}_{n-j} = \begin{pmatrix} b_{j+1,j} & b_{j+1,j+1} & \cdots & b_{j+1,n} \\ \vdots & \vdots & & \vdots \\ b_{n,j} & b_{n,j+1} & \cdots & b_{n,n} \end{pmatrix}$$

our conditions necessitate

$$\hat{B}_{n-j}\hat{u} = \hat{x}$$

To this end, we let $\hat{u} = \hat{v} + \alpha\hat{w}$ where

$$\hat{B}_{n-j}\hat{v} = \hat{x} \qquad \hat{B}_{n-j}\hat{w} = 0$$

and α is chosen so that $\|y\|_{B^{-1}} = \|x\|_{B^{-1}}$ as required by Proposition 4.2.3. Again, since B is positive definite, the principal submatrices, such as B_{n-j} are nonsingular, and therefore the choice of $\hat{u}$, and therefore u, is unique up to choice of α. That is, we let

$$\hat{v} = \begin{pmatrix} 0 \\ \tilde{v} \end{pmatrix}, \quad \tilde{v} = B_{n-j}^{-1}\hat{x} \in \mathbb{R}^{n-j}$$

and

$$\hat{w} = \begin{pmatrix} -1 \\ \tilde{w} \end{pmatrix}, \quad \tilde{w} = B_{n-j}^{-1}\hat{b}_j \in \mathbb{R}^{n-j}, \quad \hat{b}_j = \begin{pmatrix} b_{j+1,j} \\ \vdots \\ b_{n,j} \end{pmatrix}$$

Now with $\hat{v}$ and $\hat{w}$ defined above, we define

$$u = v + \alpha w = \begin{pmatrix} 0 \\ \hat{v} \end{pmatrix} + \alpha \begin{pmatrix} 0 \\ \hat{w} \end{pmatrix}$$

Note that

$$\begin{aligned} y^*B^{-1}y &= y^*B^{-1}(x - Bu) \\ &= y^*(B^{-1}x - u) \\ &= (x - Bu)^*(B^{-1}x - u) \\ &= x^*B^{-1}x - x^*u - u^*x + u^*Bu \end{aligned}$$

and therefore $\|y\|_{B^{-1}} = \|x\|_{B^{-1}}$ if and only if

$$2u^*x = u^*Bu \tag{4.6}$$

With $u = v + \alpha w$, we have that

$$\begin{gathered} u^*Bu = v^*Bv + 2\alpha w^*Bv + \alpha^2 w^*Bw \\ 2u^*x = 2v^*x + 2\alpha w^*x \end{gathered}$$

Equating these two as (4.6) requires, we obtain α by finding roots of the quadratic

$$(w^*Bw)\alpha^2 + 2w^*(Bv - x)\alpha + (v^*Bv - 2v^*x) \tag{4.7}$$

Lemma 4.2.5. *Under the assumption that A is nonsingular, the quadratic (4.7) has two real roots, at least one of which is nonzero.*

Proof. Begin by assuming that $v \neq 0$. In this case, we have two real roots, one positive and one negative if $(w^*Bw)(v^*Bv - 2v^*x) < 0$. Since B is positive definite, we simply require that $v^*Bv - 2v^*x < 0$. Partitioning B appropriately, we have that

$$\begin{aligned} v^*Bv &= \begin{pmatrix} 0 \\ \tilde{v} \end{pmatrix}^* \begin{pmatrix} B_j & B_{12} \\ B_{12}^* & B_{n-j} \end{pmatrix} \begin{pmatrix} 0 \\ \tilde{v} \end{pmatrix} \\ &= \tilde{v}^* B_{n-j} \tilde{v} \\ &= \tilde{v}^* B_{n-j} (B_{n-j}^{-1} \hat{x}) \\ &= \tilde{v}^* \hat{x} = v^* x \end{aligned}$$

Thus

$$v^*Bv - 2v^*x = v^*Bv - 2v^*Bv = -v^*Bv < 0.$$

On the other hand, if $v = 0$, then our quadratic degenerates to

$$(w^*Bw)\alpha^2 - 2w^*x\alpha$$

which has one zero root and the root

$$\alpha = \frac{2w^*x}{w^*Bw} \tag{4.8}$$

which is nonzero provided that $w^*x \neq 0$. Now, $v = 0$ if and only if $\hat{x} = 0$. Therefore, $w^*x = -x_j + \tilde{w}^*\hat{x} = -x_j$. By our assumption of nonsingularity, if $\hat{x} = 0$, then x_j must be nonzero, and therefore $\alpha \neq 0$. □

As before, we omit a detailed listing of the B^{-1}-unitary QR factorization, and point out that it will follow immediately if each column of A is reduced using the structure preserving choices developed above.

4.2.5 B^{-1}-orthonormal QR factorizations

Here we discuss the construction B^{-1}-orthonormal QR factorizations, the goal of our work. Should we successfully construct $C = QR$ such that $Q^*B^{-1}Q = I$, we will implicitly construct a Cholesky factor R of the matrix $D = C^*B^{-1}C$, since

$$C^*B^{-1}C = R^*Q^*B^{-1}QR = R^*R \tag{4.9}$$

Equivalently, should we carry out this construction in an incomplete manner, we could use the R in place of an incomplete Cholesky preconditioner without ever having to explicitly form $C^*B^{-1}C$. We begin with a proposition which gives insight to the structure of B^{-1}-orthonormal matrices.

Proposition 4.2.6. *Suppose that Q is B^{-1}-orthonormal and L is the Cholesky factor of B, that is $B = LL^*$. Then $Q = UL$ where U is a unique B^{-1}-unitary matrix.*

Proof. Suppose Q is B^{-1} orthonormal. Then, by definition,

$$Q^* B^{-1} Q = I$$

Then if L is the Cholesky factor of B, we have that

$$L^{-T} Q^* B^{-1} Q L^{-1} = L^{-T} L^{-1} = B^{-1}$$

That is, $U = QL^{-1}$ is a B^{-1}-unitary matrix. Therefore, we may express Q as $Q = UL$. Uniqueness of U follows from uniqueness and nonsingularity of the Cholesky factor L. □

Proposition 4.2.7. *Suppose Q is B^{-1}-orthonormal.*

1. *Q is nonsingular, and Q^{-T} is B-orthonormal.*

2. *If V is orthogonal (i.e. $V^*V = I$), then QV is B^{-1}-orthonormal.*

Proof. Now since Q is B^{-1}-orthonormal, it follows that

$$(\det(Q))^2 = \det(B) \neq 0$$

implying that Q is nonsingular. This may also be seen by through Proposition 4.2.6. That Q^{-T} is B-orthonormal follows from inverting the formula

$$Q^* B^{-1} Q = I$$

to obtain

$$I = Q^{-1} B Q^{-T} = (Q^{-T})^* B Q^{-T}$$

Now if V is orthogonal, then

$$(QV)^* B^{-1} (QV) = V^* Q^* B^{-1} Q V = V^* V = I$$

showing that the product QV is B^{-1}-orthonormal. □

In view of Proposition 4.2.6, we see that constructing a B^{-1}-orthonormal matrix may be achieved by simply constructing a B^{-1}-unitary matrix and post multiplying by the Cholesky factor of B. Since there is nothing practical about computing our B^{-1}-unitary QR factorization, we instead interlace the applications of our Cholesky factor. This allows us to build a B^{-1}-orthonormal QR factorization, with relative ease. As we saw, in constructing the B^{-1}-unitary QR we had to apply the inverse of an $(n-j) \times (n-j)$ matrix twice per step. In constructing the B^{-1}-orthonormal factorization, we will require only one such application.

The basic process used in building the B^{-1}-orthonormal QR factorization of a given matrix C is this

$$\text{Form } \hat{Q} \text{ such that } \hat{Q}(Ce_1) = \alpha e_1 \text{ and } \hat{Q}B\hat{Q}^* = \begin{pmatrix} 1 & 0 \\ 0 & B_1 \end{pmatrix}$$

We may continue the process as we would in the usual Householder process, this time using B_1 as the matrix defining the inner product. That B_1 defines an inner product is clear by the relation (4.2.5) and the positive definiteness of B. We are free of the difficulties of choosing u in the B^{-1}-unitary case as described above, since every time we wish to construct such a u, it is like the first step of a B^{-1}-unitary process where we have no constraints on u. This freedom comes at the price of changing our inner product at each step.

Throughout the remainder of our discussion, suppose that $B = L\Delta L^*$ is an LDL^* factorization of B, with $L = L_1 L_2 \cdots L_{n-1}$ and L_i is unit lower triangular and $\Delta = \operatorname{diag}(\delta_1, \ldots, \delta_n)$. Furthermore, let $\Delta_1 = \operatorname{diag}(\delta_1, 1, 1, \ldots, 1)$. Now if we let

$$\hat{Q} = \Delta^{-\frac{1}{2}} L_1^{-1} Q \tag{4.10}$$

where Q is B^{-1}-unitary, then

$$\hat{Q}B\hat{Q}^* = \Delta^{-\frac{1}{2}} L_1^{-1} QBQ^* L_1^{-*} \Delta^{-\frac{1}{2}} = \Delta^{-\frac{1}{2}} L_1^{-1} B L_1^{-*} \Delta^{-\frac{1}{2}} = \begin{pmatrix} 1 & 0 \\ 0 & B_1 \end{pmatrix}. \tag{4.11}$$

Thus, we must develop a B^{-1}-unitary matrix Q such that

$$\Delta^{-\frac{1}{2}} L_1^{-1} Qx = \alpha e_1, \tag{4.12}$$

or

$$Qx = \alpha L_1 \Delta^{\frac{1}{2}} e_1 = \alpha \sqrt{b_{11}} L_1 e_1. \tag{4.13}$$

Following Proposition 4.2.3, we compute

$$u = B^{-1}(x - y) \tag{4.14}$$

where $y = \alpha\sqrt{b_{11}} L_1 e_1$ and α is chosen so that $\|y\|_{B^{-1}} = \|x\|_{B^{-1}}$. We begin by computing α. Note that

$$\|y\|_{B^{-1}}^2 = \alpha^2 b_{11} \|L_1 e_1\|_{B^{-1}}^2$$

and

$$\|L_1 e_1\|_{B^{-1}}^2 = e_1^* L_1^* B^{-1} L_1 e_1 = e_1^* (L_1^{-1} B L_1^{-*})^{-1} e_1 = \frac{1}{b_{11}}$$

since

$$L_1^{-1} B L_1^{-*} = \begin{pmatrix} b_{11} & 0 \\ 0 & B_1 \end{pmatrix}.$$

It follows that

$$\alpha = \pm\|x\|_{B^{-1}}.$$

Now to resolve u, we must compute $B^{-1}L_1e_1$. To do this, we note again that $B = L_1 \begin{pmatrix} b_{11} & 0 \\ 0 & B_1 \end{pmatrix} L_1^*$. Therefore

$$B^{-1}L_1e_1 = L_1^{-*} \begin{pmatrix} \frac{1}{b_{11}} & 0 \\ 0 & B_1^{-1} \end{pmatrix} L_1^{-1}L_1e_1 = L_1^{-*}\frac{1}{b_{11}}e_1 = \frac{1}{b_{11}}e_1$$

since L_1 is unit lower triangular. Now then, $B^{-1}y$ is given by

$$B^{-1}y = \pm\|x\|_{B^{-1}}\sqrt{b_{11}}\left(\frac{1}{b_{11}}e_1\right) = \pm\frac{\|x\|_{B^{-1}}}{\sqrt{b_{11}}}e_1 \tag{4.15}$$

and so u is given by

$$u = B^{-1}x \pm \frac{\|x\|_{B^{-1}}}{\sqrt{b_{11}}}e_1. \tag{4.16}$$

Note that in the case that $B = I$ this corresponds to the usual choice of u. As in the usual process, we may choose the sign of coefficient so as to avoid cancellation. In this case, we should choose the sign to be the opposite of the sign of the first entry of the vector $B^{-1}x$.

Once the Householder-like reflector Q defined by the vector u matrix is applied, we must apply L_1^{-1} followed by $\Delta_1^{-\frac{1}{2}}$. In this manner, one step of the B^{-1}-orthonormal QR factorization process is completed. Again, to continue, we would require an LDL^* factorization of the matrix B_1. Fortunately, this is already available; simply strip the leading row and column off of L from the LDL^* factorization of B to obtain the L factor for B_1, and similarly for Δ_1.

The complete process is detailed in Algorithm 4.1. Throughout the development of the algorithm, let

$$L_j = \begin{pmatrix} 1 & & & & \\ l_{j+1,j} & 1 & & & \\ l_{j+2,j} & 0 & \ddots & & \\ \vdots & \vdots & \ddots & 1 & \\ l_{n,j} & 0 & & 0 & 1 \end{pmatrix} \in \mathbb{R}^{(n-j+1)\times(n-j+1)},$$

$$\hat{L}_j = \begin{pmatrix} 1 & & & & \\ l_{j+1,j} & 1 & & & \\ l_{j+2,j} & l_{j+2,j+1} & \ddots & & \\ \vdots & \vdots & \ddots & 1 & \\ l_{n,j} & l_{n,j+2} & & l_{n,n-1} & 1 \end{pmatrix} \in \mathbb{R}^{(n-j+1)\times(n-j+1)},$$

$$\Delta_j = \operatorname{diag}(\delta_j, 1, \ldots, 1) \in \mathbb{R}^{(n-j+1)\times(n-j+1)}$$

$$\text{and} \quad \hat{\Delta}_j = \operatorname{diag}(\delta_j, \delta_{j+1}, \ldots, \delta_n) \in \mathbb{R}^{(n-j+1)\times(n-j+1)}.$$

Similarly, let

$$C_j = \begin{pmatrix} c_{j,j} & c_{j,j+1} & \cdots & c_{j,n} \\ c_{j+1,j} & c_{j+1,j+1} & \cdots & c_{j+1,n} \\ \vdots & \vdots & & \vdots \\ c_{n,j} & c_{j,j+1} & \cdots & c_{n,n} \end{pmatrix}$$

denote the $(n-j+1) \times (n-j+1)$ lower principal submatrix of C.

Algorithm 4.1 QR factorization with respect to B^{-1} inner product

```
Input: $C$, symmetric positive definite $B = L\Delta L^*$.
 1: for $j = 1, 2, \ldots, n-1$ do
 2:     $x = \begin{pmatrix} c_{j,j} & \cdots & c_{n,j} \end{pmatrix}$
 3:     $u = \hat{L}_j^{-*}\hat{\Delta}_j^{-1}\hat{L}_j^{-1}x$.
 4:     $\beta = \sqrt{x^*u}$
 5:     $\gamma = \frac{\beta}{\sqrt{\delta_j}}\,\mathrm{sign}(-u_1)$
 6:     $u_1 = u_1 - \gamma$
 7:     $\sigma = \frac{1}{\beta^2 - \gamma x_1}$
 8:     $z = \Delta_j^{-\frac{1}{2}}L_j^{-1}x - \beta e_1$
 9:     $C_j = \Delta_j^{-\frac{1}{2}}L_j^{-1}C_j - \sigma z u^* C_j$
10: end for
```

Some comments regarding Algorithm 4.1 are in order. The coefficient σ corresponds to $\frac{2}{u^*Bu}$. To see this, note that

$$u^*Bu = \left(B^{-1}x - \gamma e_1\right)^* (x - \gamma B e_1) = \|x\|_{B^{-1}}^2 - 2\gamma e_1^* x + \gamma^2 b_{11}.$$

Now, $\gamma^2 = \frac{\beta^2}{b_{11}}$ and so $\gamma^2 b_{11} = \beta^2 = \|x\|_{B^{-1}}^2$. It follows that

$$u^*Bu = 2\|x\|_{B^{-1}}^2 - 2\gamma e_1^* x,$$

and the expression for σ follows. The expression for z on line eight is a simplification of $z = \Delta_j^{-\frac{1}{2}}L_j^{-1}Bu$

4.2.6 Performing the process incompletely

Of course, for the large, sparse problems we consider, it is infeasible to use the complete factor obtained from Algorithm 4.1. Thus we must find some way to apply it incompletely, while maintaining sparsity and orthogonality with respect to some inner product.

A major problem of Algorithm 4.1, with respect to developing sparse factors, is that Cholesky factors of sparse matrices are frequently not sparse. Therefore, instead of using the complete Cholesky factor of B, we may consider using an incomplete

Cholesky factor, or an RIF. Moreover, application of the inverse of a sparse triangular matrix to a sparse vector is more expensive than necessary when we only wish to apply the inverses approximately. One method of approximate application of the inverse is to appeal to a truncated Neumann series for the matrix, as is done in ILUS [13]. This has the advantage of allowing computation in sparse-sparse mode. That is, sparse vectors are being multiplied by sparse matrices, and presumably resulting in sparse vectors.

Another strategy would be to use a factored sparse approximate inverse of B, in place of the Cholesky factors. With this approach, we would always work in sparse-sparse mode thereby develop sparser factors.

The version we implement is detailed in Algorithm 4.2 and makes use of an incomplete Cholesky factorization for B. That is, $B \approx \tilde{B} = L\Delta L^*$ is input as the factorization for B. The vector u is then computed and subjected to a numerical dropping rule. The relative threshold dropping rule we apply appears in line 4. The vector x is recomputed so as to maintain the $\tilde{B}$-orthogonality of the transformation being applied. The algorithm proceeds as in Algorithm 4.1 until the last step in the for loop, where numerical dropping is applied to the most recently computed column of R to introduce more sparsity. This step is referred to as post-filtration. This algorithm will be denoted by IQR(τ, η), where τ is the drop tolerance applied to u, and η is the post-filtration tolerance. In our examples, however, we will frequently use $\eta = \tau$. We also drop the entries in the strict lower triangle to enforce reduction to an upper triangular form. We note that a strategy such as that found in [20] may be implemented to make the reflectors competitive with Givens rotations for sparse matrices.

4.3 Other preconditioning techniques for (D, B)

In addition to the incomplete B^{-1}-orthogonal QR factorization, we present a few techniques to construct incomplete factorization type preconditioners for D using existing techniques.

4.3.1 ILU and ILQ based preconditioners

A first approach to constructing an incomplete factorization for D is to construct an ILU factorization of $A - \mu B$ and exploit symmetry to precondition. That is, if the factors are complete, and $B = TT^*$ is a Cholesky factor of B, we have

$$D = (A - \mu B)B^{-1}(A - \mu B) = (U^* L^* T^{-*})(T^{-1} LU). \tag{4.17}$$

The preconditioner may then be used in block Arnoldi-like method by applying

$$(T^{-1}LU)^{-1}(U^* L^* T^{-*})^{-1} = U^{-1}L^{-1}BL^{-*}U^{-*}. \tag{4.18}$$

Algorithm 4.2 IQR(τ, η) factorization with respect to B^{-1} inner product

```
Input: $C$, symmetric positive definite $B \approx L\Delta L^*$, drop tolerance $\tau$, post-filtration
    tolerance $\eta$.
 1: for $j = 1, 2, \ldots, n-1$ do
 2:    $x = \begin{pmatrix} c_{j,j} & \cdots & c_{n,j} \end{pmatrix}$
 3:    $u = \hat{L}_j^{-*}\hat{\Delta}_j^{-1}\hat{L}_j^{-1}x$.
 4:    $droptol = \tau\sqrt{x^*u}$
 5:    if $|u_i| < droptol$ then
 6:       Drop $u_i$.
 7:    end if
 8:    Recompute $x = \hat{L}_j\hat{\Delta}_j\hat{L}_j^*u$
 9:    $\beta = \sqrt{x^*u}$
10:    $\gamma = \frac{\beta}{\sqrt{\delta_j}}\operatorname{sign}(-u_1)$
11:    $u_1 = u_1 - \gamma$
12:    $\sigma = \frac{1}{\beta^2 - \gamma x_1}$
13:    $z = \Delta_j^{-\frac{1}{2}}L_j^{-1}x - \beta e_1$
14:    $C_j = \Delta_j^{-\frac{1}{2}}L_j^{-1}C_j - \sigma z u^* C_j$
15:    if $|c_{ij}| < \beta * \eta$, for $1 \leq i < j$ then
16:       Drop $c_{ij}$.
17:    end if
18:    Drop $c_{ij}$ if $i > j$ to enforce upper triangularity.
19: end for
```

In general, we may expect μ to be well inside the spectrum, thereby making the matrix $A-\mu B$ highly indefinite. In this case, special care must be taken to construct ILU factorizations which exist and are sufficiently stable. Pivoting may be required to keep stability, thereby destroying symmetry. To preserve symmetry, we may have to turn to the ILUS factorization summarized in Section 3.1 and detailed in [13]. In general, constructing stable ILU factors for highly indefinite operators is difficult at best, and we refer the interested reader to [5], [12], [13], [52] and the references therein. In our case, the stability problem is exacerbated by the necessity of performing two triangular solves with each factor!

Due to the normal equations structure of the operator D, it is attractive to consider factorizations of the form $A-\mu B \approx LQ$, particularly when $B = I$. Should B not be the identity, we may compute an ILQ factorization of $(A-\mu B)Z$ where $B^{-1} \approx ZZ^*$. Here, ZZ^* may be taken to be a sparse approximate inverse of B. As application of D necessitates the action of B^{-1} to a vector as exactly as possible, we may have a complete (or nearly so) Cholesky factor of B on hand. Using this in place of Z may still not be desirable, as we wish to maintain sparsity of the factors. In the case that we have a complete LQ factorization of $(A-\mu B)Z$ where $ZZ^* = B^{-1}$, we see that

$$D = (A-\mu B)B^{-1}(A-\mu B) = LQQ^*L^* = LL^* \tag{4.19}$$

and so we have implicitly produced a Cholesky factor of D without the need to explicitly form D. A preconditioner based on this kind of ILQ requires two triangular solves: one with L followed by one with L^*. As discussed in Section 3.1.2, there are many methods for constructing incomplete LQ factorizations of a given matrix. The method we favor here is TIGO [4] [45], and this is the method we execute in the examples in Section 4.4. Like the ILU factorization, however, there are stability problems with ILQ factors as well, especially when the matrix to be factored is poorly conditioned.

4.3.2 RIF based preconditioners

In a strategy similar to that using ILQ factorizations above, we may construct preconditioners based on the RIF process for the normal equations as described in [8]. An advantage to using RIF is that due to the link with SAINV, the factors frequently demonstrate much more stable behavior than other incomplete factorization preconditioners, especially when applied to ill-conditioned operators. As before, we take a sparse approximate inverse for B, say $B \approx ZZ^*$ and we construct the RIF of $(A-\mu B)ZZ^*(A-\mu B)$ by collecting the multipliers formed in performing incomplete orthogonalization with respect to the $(A-\mu B)ZZ^*(A-\mu B)$ inner product. Note that constructing the RIF in this way does not require explicit formation of

$(A - \mu B)ZZ^*(A - \mu B)$. Upon completion, we will have implicitly formed a Cholesky factor of $(A - \mu B)ZZ^*(A - \mu B)$, which we may use to precondition D.

4.4 Numerical Examples III

Here we provide examples illustrating the effectiveness of the preconditioning schemes for the interior formulation.

4.4.1 Example: The Laplacian on a barbell-shaped domain

We return to the problem of Section 2.6.1, only this time we seek the eigenvalues nearest the target $\mu = 10000$. This is indeed a rather contrived example, we simply use it to illustrate the effectiveness of various preconditioning techniques, for the formulation (D, M), where M is the mass matrix, taking the role of B, and $D = CM^{-1}C$ with $C = K - 10000M$.

The first preconditioner we consider is the factor obtained from the incomplete M^{-1} QR factorization of C. Due to the structure of the matrix and the inefficiencies of our codes implementing the incomplete version of Algorithm 4.1, we first apply a reverse Cuthill-McKee ordering to C and M before computing the incomplete Householder QR decomposition with respect to the M^{-1} inner product. We note that in our experiences, the convergence behavior seen with and without application of pre-orderings is very similar for these types of problems; pre-ordering simply affords more sparsity and should, in general be applied.

The convergence histories for the first Ritz value obtained from adaptBlock-VarIIp(2,24) with IQR preconditioners with different drop tolerances are displayed in Figure 4.2. The pertinent preconditioner statistics are reported in Table 4.1. In all cases, the post-filtration tolerance is taken to be the same as the drop tolerance. Also, in all cases, the algorithm ran 250 outer iterations before reducing the residual to the desired level of 1e−07. The algorithm preconditioned by IQR(1e−03, 1e−03) (dashed) came close, however, and in view of these examples, it seems that should a good dropping strategy be applied to Algorithm 4.1, it may serve as an excellent preconditioner, not only for problems such as this, but also for Schur complement systems arising in the solution of saddle point problems.

In view of the entries in Table 4.1, it is abundantly clear that our post-filtration strategy is implemented over aggressively, and should probably be relaxed to allow for higher quality preconditioners. We also note that none of the factorizations are particularly stable, yet the factor developed with drop tolerance $\tau = $ 1e−03, performs moderately well. It is not stable enough, however, to be truly effective, especially in view of the examples that follow.

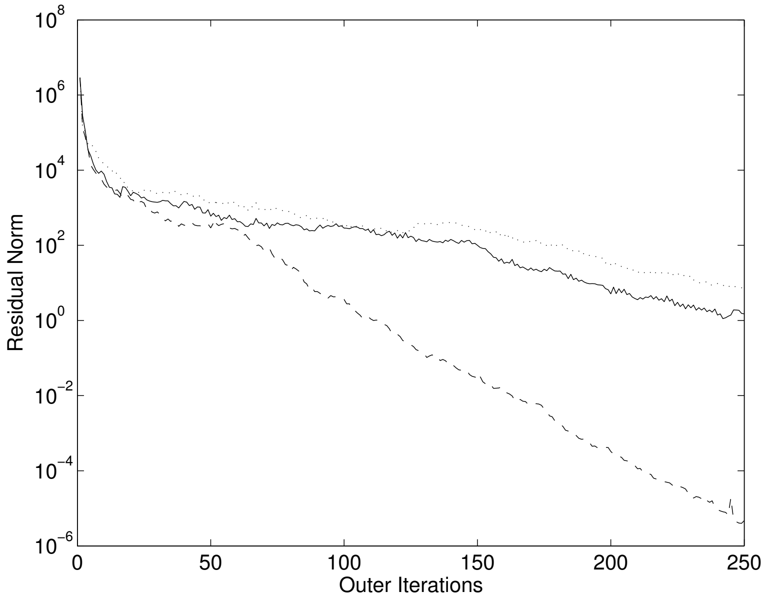


Figure 4.2: Convergence histories of smallest Ritz value of (D, M) computed by adaptBlockVarIIp(2,24) preconditioned by IQR(τ, τ) with τ = 1e−01 (solid), τ = 1e−02 (dotted), and τ = 1e−03 (dashed).

τ	nnz(R)	$\|D - R^*R\|_F/\|D\|_F$	$\|I - R^{-*}DR^{-1}\|_F$
1e−01	2541	8.512e−01	4.486e+05
1e−02	3289	8.274e−01	1.717e+09
1e−03	27779	2.217e−01	7.178e+11

Table 4.1: Preconditioner statistics for IQR with various drop tolerances τ.

SAINV	RIFNR	matvecs	preapps	final residual
	1e−01	11725	5520	9.319e−08
1e−01	1e−02	4483	2212	7.238e−08
	1e−03	1425	672	7.554e−08
	1e−01	12796	6024	7.002e−08
1e−02	1e−02	5095	2400	5.446e−08
	1e−03	1425	672	7.554e−08
	1e−01	12490	5880	7.303e−08
1e−03	1e−02	4840	2280	8.475e−08
	1e−03	1527	720	1.587e−08

Table 4.2: Convergence characteristics of adaptBlockVarIIp(2,24) applied to the pencil (D, M) with various preconditioners.

SAINV	RIFNR	nnz(L)	$\|D - LL^*\|_F/\|D\|_F$	$\|I - L^{-1}DL^{-*}\|_F$
	1e−01	100037	3.663e−01	2.612e+01
1e−01	1e−02	368251	3.678e−01	2.333e+01
	1e−03	687361	3.697e−01	2.186e+01
	1e−01	100037	3.663e−01	2.612e+01
1e−02	1e−02	368251	3.678e−01	2.333e+01
	1e−03	687361	3.697e−01	2.186e+01
	1e−01	101070	3.534e−01	2.563e+01
1e−03	1e−02	360828	3.580e−01	2.269e+01
	1e−03	681953	3.600e−01	2.109e+01

Table 4.3: Preconditioner statistics for SAINV/RIFNR preconditioners with various drop tolerances.

We also considered the SAINV/RIFNR preconditioner for this problem, however, here we did not use a pre-ordering. We recall the process briefly. First, an SAINV approximation is formed for B, that is, $B^{-1} \approx ZZ^*$. This is followed by RIFNR (RIF applied to the normal equations) [8] for Z^*C. Details concerning the convergence of adaptBlockVarIIp(2,24) are presented in Table 4.2. Preconditioner statistics are reported in Table 4.3. Notably, here all of these operators have about the same stability and accuracy measurements, however, it is clear from Figure 4.3 and Figure 4.4 that the important quality of the preconditioners is due to the quality of the RIF. In fact the factorized SAINV for M is the same for both $\tau = 1e{-}01$ and $\tau = 1e{-}02$, and therefore we see the same stability and accuracy measurements at both levels.

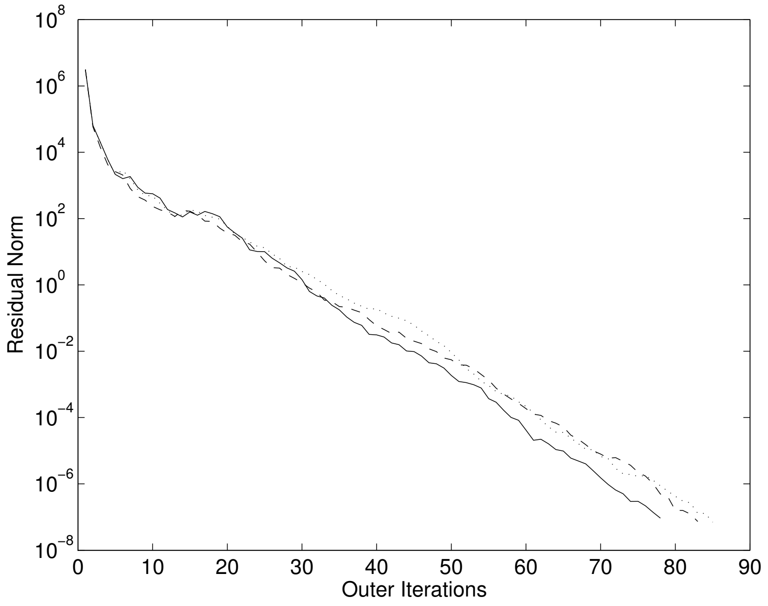


Figure 4.3: Convergence histories of smallest Ritz value of (D, M) computed by adaptBlockVarIIp(2,24) preconditioned by SAINV/RIFNR. SAINV is constructed with various tolerances ($\tau = 1\text{e}{-01}$ (solid), $\tau = 1\text{e}{-02}$ (dotted), and $\tau = 1\text{e}{-03}$ (dashed)) and RIFNR is constructed with fixed drop tolerance and post-filtration tolerance $\tau = 1\text{e}{-01}$.

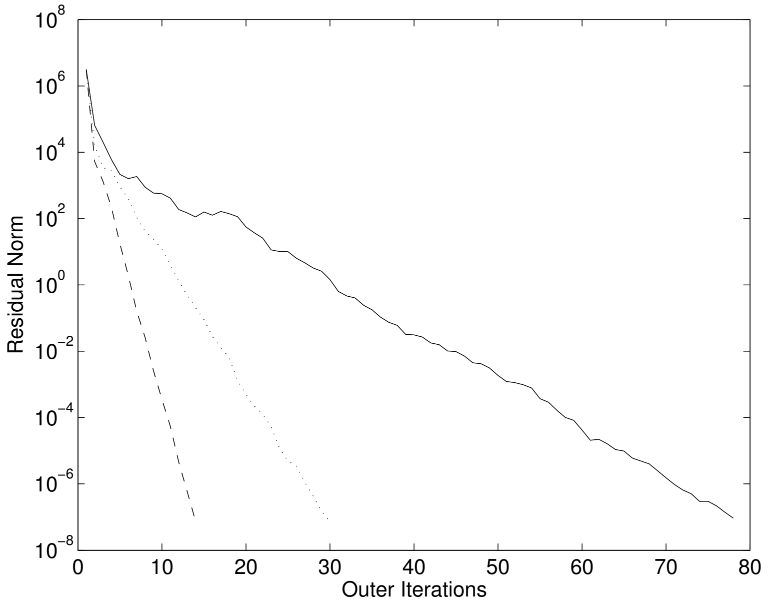


Figure 4.4: Convergence histories of smallest Ritz value of (D, M) computed by adaptBlockVarIIp(2,24) preconditioned by SAINV/RIFNR. RIFNR is constructed with various drop and post-filtration tolerances ($\tau = 1\text{e}{-01}$ (solid), $\tau = 1\text{e}{-02}$ (dotted), and $\tau = 1\text{e}{-03}$ (dashed)) and SAINV is constructed with fixed drop tolerance tolerance $\tau = 1\text{e}{-01}$.

4.4.2 Example: The Platzman model

We first consider the eigenvalue problem from Platzman's oceanographic models. This problem is part of the Harwell-Boeing collection [16], and we obtained it from the Matrix Market [39]. This problem is well-known as a difficult eigenvalue problem as is discussed in [27], among other places. The eigenvalues of interest are located in the interval $(0.0001, 0.024)$, and these correspond to interior eigenvalues of the problem (A, I). The original formulation is skew-symmetric, and we are dealing with the negative square of that, therefore the eigenvalues come in pairs, with the exception of an isolated singleton at zero.

For our experiments, we seek the eigenvalues near 0.024 and use the (D, B) formulation as described above. We are working with the matrix corresponding to the North Atlantic submodel. We consider n preconditioning strategies: the ILU strategy for D, the ILQ strategy, and the RIFNR strategy, all described in Section 4.3, and the IQR via incomplete Householder strategy described in Section 4.2. BlockVarIIp(2, 10) was executed and details of the runs are contained in Table 4.4. Statistics for the preconditioners considered are collected in Table 4.5. For these preconditioners, the relative accuracy and stability measurements were computed in the following manners. For the RIFNR and the ILQ preconditioners, we computed $\|A^*A - LL^*\|_F/\|A^*A\|_F$, and $\|I - L^{-1}A^*AL^{-*}\|_F$ as relative accuracy and stability respectively. Similarly, for the IQR we computed $\|A^*A - R^*R\|_F/\|A^*A\|_F$ as the relative accuracy measure and $\|I - R^{-*}A^*AR^{-1}\|_F$ as the stability measure. In the case of the ILU preconditioner, however, relative accuracy was measured by the quantity

$$\|AA^* - LUU^*L^*\|_F/\|AA^*\|_F \tag{4.20}$$

and stability by

$$\|I - U^{-1}L^{-1}AA^*U^{-*}L^{-*}\|_F. \tag{4.21}$$

Also, since $C = A - 0.024I$ is highly indefinite, both the L and U factors had to be stored, as we cannot expect that the two are related.

For the RIFNR and the IQR preconditioners, the post-filtration tolerances were taken to be the same as the drop tolerances τ. Also, the two numbers in column three for the ILU preconditioner correspond to the number of nonzero elements in L and U respectively. The IQR preconditioner constructed with drop tolerance $\tau = 1\text{e}{-02}$ contained an element on the order of $1\text{e}{-17}$ as the last entry in the diagonal, making the factorization extremely unstable. This factorization was repaired by putting an element on the order of the drop tolerance in the last diagonal entry. Statistics are reported for the repaired factor, and this is the reason for the * marking. In general, however, the IQR preconditioner outperformed the others in terms of matvecs at all levels, despite its instability. It was also the densest preconditioner by far. We

Method	τ	matvecs	preapps	$\|r_1\|$	$\|r_2\|$
NONE	—	12565	5422	5.322e−10	9.799e−10
RIFNR	1e−01	4665	2022	6.243e−10	7.142e−10
	1e−02	878	382	6.259e−10	8.563e−10
	1e−03	279	122	1.650e−11	4.050e−11
	1e−04	186	82	1.700e−14	5.311e−15
ILU	1e−01	10702	4642	4.561e−10	9.146e−10
	1e−02	463	202	2.385e−10	2.346e−10
	1e−03	186	82	9.818e−13	4.607e−13
	1e−04	141	62	7.529e−12	5.643e−10
ILQ	1e−01	18356	7902	4.768e−10	9.586e−10
	1e−02	1984	862	7.532e−10	8.662e−10
	1e−03	232	102	6.850e−11	6.185e−10
	1e−04	186	82	3.041e−13	5.082e−13
IQR	1e−01	1293	562	6.605e−10	7.559e−10
	1e−02	232	102	3.125e−10	6.484e−10
	1e−03	140	62	2.162e−10	2.112e−10
	1e−04	94	42	6.861e−11	1.280e−10

Table 4.4: Convergence characteristics of BlockVarIIp(2, 10) applied to PLAT362 for various preconditioners.

Method	τ	nnz	relative accuracy	stability
RIFNR	1e−01	769	2.960e−02	4.564e+01
	1e−02	3251	4.539e−03	1.207e+01
	1e−03	10850	6.789e−04	4.609e+00
	1e−04	31018	8.656e−05	1.849e+00
ILU	1e−01	2222, 2416	1.683e−02	1.899e+10
	1e−02	7886, 10467	2.313e−03	8.268e+01
	1e−03	30706, 30225	3.163e−04	4.644e+00
	1e−04	42412, 41997	2.124e−05	1.210e−01
ILQ	1e−01	2990	1.539e−02	2.519e+05
	1e−02	16106	6.038e−03	1.838e+05
	1e−03	33107	1.014e−03	1.746e+02
	1e−04	43131	1.391e−04	4.335e+00
IQR	1e−01	15800	5.227e−03	1.076e+10
	1e−02*	40064	9.376e−03	2.268e+08
	1e−03	47357	5.621e−05	1.287e+00
	1e−04	48508	2.088e−06	4.661e−02

Table 4.5: Preconditioner statistics for various preconditioners applied to interior formulation for PLAT362.

again stress that should a dropping strategy with an eye for sparsity be developed while maintaining the quality we see here, in terms of reducing the number of matrix-vector multiplies required to get convergence, we feel that this method could be very competitive. The RIFNR preconditioner performed well, giving good convergence performance while maintaining moderate sparsity. Finally, we would be terribly remiss if we failed to mention that shift-and-invert Lanczos algorithms perform very well on this particular example, see [27] for discussions specific to this problem.

Chapter 5

Conclusions and Future Work

We have presented a block generalization of the algorithm found in [25] which is capable of resolving multiple and clustered eigenvalues while maintaining good convergence. We have also developed an adaptive version of this algorithm which moves towards the goal of constructing a black-box inverse free algorithm for computing a few of the algebraically smallest eigenpairs of a symmetric definite pencil. Moreover, we have presented and analyzed some preconditioning schemes which accelerate the convergence of this method. With respect to preconditioning, we have shown some ways of constructing preconditioners so that a simple transformation may be applied which allows application of our block algorithm to the task of locating interior eigenvalues of a symmetric definite pencil.

There are many directions for future research based on this work. First of all, more work should be done to make the incomplete B^{-1}-orthogonal Householder process of Chapter 4 truly viable. As mentioned in Section 4.2.6, one way of doing this may be to appeal to truncated Neumann series as in ILUS [13]. Also, we may consider using a factored sparse approximate inverse to apply B^{-1} at each step of the iteration.

Constructing good preconditioners for the matrix $D = (A-\mu B)B^{-1}(A-\mu B)$ does not, however, give a truly inverse free method for locating the eigenvalues nearest the target μ. An inverse free method may be obtainable by appealing to harmonic Ritz pairs [30], [41], however, convergence is very frequently poor, and preconditioning would play a very large role. As of yet, it is not clear how to properly construct preconditioners to yield good harmonic Ritz pairs. We hope that understanding how to extract good approximations to interior eigenpairs will lead to an algorithm for more general pencils.

Although the convergence of the block algorithm mimics that of the algorithm upon which it is based, a formal convergence theory such as that in [25] has yet to be established. Formalizing our understanding of the convergence behavior of the algorithm is a top priority.

Finally, we anticipate polishing our research codes and distributing them publicly.

Bibliography

[1] M. A. Ajiz and A. Jennings, *A robust incomplete Choleski-conjugate gradient algorithm*, Int. J. Numer. Methods. Eng., 20 (1984), pp. 949–966.

[2] P. Arbenz, U. L. Hetmaniuk, R. B. Lehoucq, and R. S. Tuminaro, *A Comparison of Eigensolvers for Large-scale 3D Modal Analysis using AMG-Preconditioned Iterative Methods*, Technical Report SAND 2005-0282J, Sandia National Laboratories, 2005.

[3] Z. Bai, D. Day, and Q. Ye, *ABLE: An Adaptive Block Lanczos Method for Non-Hermitian Eigenvalue Problems*, SIAM J. Matrix Anal. Appl., 20 (1999), pp. 1060–1082.

[4] Z.-Z. Bai, I. S. Duff, and A. J. Wathen, *A class of incomplete Orthogonal factorization methods. I: methods and theories*, Report 99/13, Oxford University Computing Laboratory, 1999.

[5] M. Benzi, *Preconditioning Techniques for Large Linear Systems: A Survey*, J. Comp. Phys., 182 (2002), pp. 418–477.

[6] M. Benzi, J. K. Cullum, and M. Tůma, *Robust Approximate Inverse Preconditioning for the conjugate gradient method*, SIAM J. Sci. Comp., 22 (2000), pp. 1318–13xx.

[7] M. Benzi and M. Tůma, *A Robust Incomplete Factorization Preconditioner for Positive Definite Matrices*, Num. Lin. Alg. Appl., 10 (2003), pp. 385–400.

[8] ——, *A Robust Preconditioner with Low Memory Requirements for Large Sparse Least Squares Problems*, SIAM J. Sci. Comp., 25 (2003), pp. 499–512.

[9] L. Bergamaschi, G. Gambolati, and G. Pini, *Asymptotic Convergence of Conjugate Gradient Methods for the Partial Symmetric Eigenproblem*, Num. Lin. Alg. Appl., 4 (1996), pp. 69–84.

[10] A. Bouras and V. Fraysse, *A Relaxation Strategy for the Arnoldi Method in Eigenproblems*, Technical Report TR/PA/00/16, CERFACS, 2000.

[11] A. Canning, L. wang Wang, A. Williamson, and A. Zunger, *Parallel Empirical Pseudopotential Electronic Structure Structure Calculations for Million Atom Systems*, J. Comp. Phys., 160 (2000), pp. 29–41.

[12] E. Chow and Y. Saad, *Experimental study of ILU preconditioners for indefinite matrices*, J. Comput. Appl. Math., 86 (1997), pp. 387–414.

[13] ——, *ILUS: An Incomplete LU Preconditioner in Sparse Skyline Format*, Technical Report UMSI-95-78, University of Minnesota Supercomputing Institute, 1997.

[14] J. K. Cullum and R. A. Willoughby, *Lanczos Algorithms for Large Symmetric Eigenvalue Computations Vol. I: Theory*, vol. 41 of Classics in Applied Mathematics, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2002. Reprint of the 1985 original.

[15] J. W. Demmel, *Applied Numerical Linear Algebra*, SIAM, Philadelphia, PA, 1997.

[16] I. S. Duff, R. G. Grimes, and J. G. Lewis, *Users' Guide for the Harwell-Boeing Sparse Matrix Collection (Release I)*, Technical Report TR/PA/92/86, CERFACS, 1992.

[17] D. R. Fokkema, G. L. G. Sleijpen, and H. A. V. D. Vorst, *Jacobi-Davidson style QR and QZ algorithms for the reduction of Matrix Pencils*, SIAM J. Sci. Comput., 20 (1998), pp. 94–125.

[18] R. Freund, *Band Lanczos Method (Section 4.6)*, in Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide, Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst, eds., Philadelphi, PA, 2000, SIAM.

[19] G. Gambolati, G. Pini, and F. Sartoretto, *An Improved iterative optimization technique for the leftmost eigenpairs of large symmetric matrices*, J. Comp. Phys., 74 (1988), pp. 41–60.

[20] A. George and J. Liu, *Householder Reflectors versus Givens Rotations in Sparse Orthogonal Decompositions*, Lin. Alg. Appl., 88/89 (1987), pp. 223–238.

[21] R. Geus, *The Jacobi-Davidson algorithm for solving large sparse symmetric eigenvalue problems with application to the design of accelerator cavities*, PhD Thesis, ETH Zürich, 2002.

[22] G. H. Golub and R. Underwood, *The Block Lanczos Method for Computing Eigenvalues*, in Mathematical Software III, J. R. Rice, ed., New York, NY, 1977, Academic Press, pp. 361–377.

[23] G. H. Golub and C. F. VanLoan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, MD, 3rd ed., 1996.

[24] G. H. Golub and Q. Ye, *Inexact Inverse Iteration for Generalized Eigenvalue Problems*, BIT, 40 (2000), pp. 671–684.

[25] ——, *An Inverse Free Preconditioned Krylov Subspace Method for Symmetric Generalized Eigenvalue Problems*, SIAM J. Sci. Comp., 24 (2002), pp. 312–334.

[26] G. H. Golub, Z. Zhang, and H. Zha, *Large sparse symmetrci eigenvalue problems with homogeneous linear constraints: The Lanczos process with inner-outer iterations*, Lin. Alg. Appl., 309 (2000), pp. 289–306.

[27] R. G. Grimes, J. G. Lewis, and H. D. Simon, *A Shifted Block Lanczos Algorithm for Solving Sparse Symmetric Generalized Eigenproblems*, SIAM J. Matrix. Anal. Appl., 15 (1994), pp. 228–272.

[28] L. Hoffnung. Private Communication, 2005.

[29] R. A. Horn and C. R. Johnson, *Matrix Analysis*, Cambridge University Press, Cambridge, UK, 1985.

[30] Z. Jia, *The convergence of harmonic Ritz values, harmonic Ritz vectors, and refined harmonic Ritz vectors*, Math. Comp., 156–156 (2004), pp. 289–309.

[31] T. Kato, *Perturbation Theory for Linear Operators*, Springer-Verlag, Berlin, 1966.

[32] A. V. Knyazev, *Preconditioned Eigensolvers—an oxymoron?*, Electron. Trans. Numer. Anal., 7 (1998), pp. 104–123.

[33] ——, *Preconditioned Eigensolvers (Section 11.3)*, in Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide, Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst, eds., Philadelphi, PA, 2000, SIAM.

[34] ——, *Toward the Optimal Preconditioned Eigensolver: Locally Optimal Block Preconditioned Conjugate Gradient*, SIAM J. Sci. Comp., 23 (2001), pp. 517–541.

[35] A. V. Knyazev and K. Neymeyr, *A geometric theory for preconditioned inverse iteration III: A short and sharp convergence estimate for generalized eigenvalue problems*, Lin. Alg. App., 358 (2003), pp. 95–114.

[36] ——, *Efficient Solution of Symmetric Eigenvalue Problems using Multigrid Preconditioners in the Locally Optimal Block Conjugate Gradient Method*, Elec. Trans. Num. Anal., 15 (2003), pp. 38–55.

[37] D. S. Mackey, N. Mackey, and F. Tisseur, *$\mathbb{G}$-reflectors: analogues of Householder transformations in scalar product spaces*, Numerical Analysis Report 420, Manchester Centre for Computational Mathematics, 2003.

[38] D. S. Mackey, N. Mackey, and F. Tisseur, *Structured Tools for Structured Matrices*, Electronic J. Linear Algebra, 10 (2003), pp. 106–145.

[39] Matrix Market. http://math.nist.gov/MatrixMarket/.

[40] J. H. Money and Q. Ye, *Algorithm 845: EIGIFP: A MATLAB Program for Solving Large Symmetric Generalized Eigenvalue Problems*, ACM Trans. Math. Softw., 31 (2005), pp. 270–279.

[41] R. B. Morgan, *Computing interior eigenvalues of large matrices*, Lin. Alg. Appl., 74 (1991), pp. 1441–1456.

[42] K. Neymeyr, *A geometric theory for preconditioned inverse iteration I: Extrema of the Rayleigh quotient*, Lin. Alg. App., 332 (2001), pp. 61–85.

[43] ——, *A geometric theory for preconditioned inverse iteration II: Convergence estimates*, Lin. Alg. App., 332 (2001), pp. 87–104.

[44] ——, *A Hierarchy of Preconditioned Eigensolvers for Elliptic Differential Operators*, Habilitationsschrift, Mathematishces Institut, Universität Tübingen, 2001.

[45] A. T. Papadopoulos, I. S. Duff, and A. J. Wathen, *Incomplete Orthogonal Factorization Methods Using Givens Rotations II: Implementation and Results*, Report 02/07, Oxford University Computing Laboratory, 2002.

[46] B. N. Parlett, *The Symmetric Eigenvalue Problem*, vol. 20 of Classics in Applied Mathematics, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1998. Corrected reprint of the 1980 original.

[47] A. Ruhe, *Implementation Aspects of Band Lanczos Algorithms for Computation of Eigenvalues of Large Sparse Symmetric Matrices*, Math. Comp., 33 (1979), pp. 680–687.

[48] Y. Saad, *On the Rates of Convergence of the Lanczos and the Block-Lanczos Methods*, SIAM J. Num. Analysis., 17 (1980), pp. 687–706.

[49] ——, *Preconditioning techniques for nonsymmetric and indefinite linear systems*, J. Comput. Appl. Math., 29 (1988), pp. 89–105.

[50] ——, *Numerical methods for large eigenvalue problems*, Algorithms and Architectures for Advanced Scientific Computing, Manchester University Press, Manchester, 1992.

[51] ——, *ILUT: A dual threshold incomplete LU factorization*, Numer. Linear Algebra Appl., 1 (1994), pp. 387–402.

[52] ——, *Iterative Methods for Sparse Linear Systems*, PWS Publishing Company, Boston, MA, 1996.

[53] B. Smith, J. Boyle, B. Garbow, Y. Ikebe, V. Klema, and C. B. Moler, *Matrix Eigensystem Routines—EISPACK Guide*, vol. 6 of Lecture Notes in Computer Science, Springer-Verlag, Berlin, 1974.

[54] Y. Sun, *The Filter Algorithm for Solving Large-Scale Eigenproblems from Accelerator Simulations*, PhD Thesis, Stanford University, 2003.

[55] M. Tůma. Private Communication, 2005.

[56] U. Trottenberg, C. Oosterlee, and A. Schüller, *Multigrid*, Academic Press, London, 2001.

[57] R. Underwood, *An Iterative Block Lanczos Method for the Solution of Large Sparse Symmetric Eigenproblems*, PhD Thesis, Stanford University, 1975.

[58] M. Verbeek, J. Cullum, and W. Joubert, *AMGToolBox*, 2002.

[59] X. Wang, *Incomplete Factorization Preconditioning for Linear Least Squares Problems*, PhD Thesis, University of Illinois at Urbana-Champaign, 1993.

[60] X. Wang, K. A. Gallivan, and R. Bramley, *CIMGS: An Incomplete Orthogonal Factorization Preconditioner*, SIAM J. Sci. Comput., 18 (1997), pp. 516–536.

[61] L. wang Wang and A. Zunger, *Solving Schroedinger's equation around a desired energy: Application to silicon quantum dots*, J. Chem. Phys., 100 (1994), pp. 2394–2397.

[62] J. H. Wilkinson, *The Algebraic Eigenvalue Problem*, Oxford University Press, New York, 1965.

[63] Q. Ye, *An adaptive block Lanczos algorithm*, Numerical Algorithms, 12 (1996), pp. 97–110.

Vita

1. Background.
    (a) Date of Birth: 21 November 1977
    (b) Place of Birth: Clearwater, FL
2. Academic Degrees.
    (a) B.S., Mathematics (Computer Science minor), University of Arkansas, 1999.
    (b) M.S., Mathematics, University of Kentucky, 2002.
3. Professional Experience.
    (a) Teaching Assistant, Mathematics Department, University of Kentucky, Fall 1999–Spring 2005
    (b) Research Assistant under Dr. Qiang Ye, University of Kentucky, Spring 2003–Fall 2004, Summer 2005.
    (c) Software Engineering Intern, Raytheon E-Systems, St. Petersburg, FL, Summer 1996, 1997
    (d) Software Engineering Intern, Honeywell, Clearwater, FL, Summer 1998, 1999
    (e) Intern under Dr. Noël M. Nachtigal, Sandia National Laboratories, Livermore, CA, Summer 2001, 2002, 2003
4. Publications.
    (a) A Network Diversion Vulnerability Problem. *(with Ariel Cintron-Arias, Norman Curet, Lisa Denogean, Robert Ellis, Corey Gonzalez, and Shobha Oruganti)* IMA Mathematical Modeling in Industry Summer 2000 Program for Graduate Students, University of Minnesota, Technical Report 1752-5. February 2001.