ABSTRACT

Title of dissertation: Residual Arnoldi Methods : Theory, Package, and Experiments

Che-Rung Lee, Doctor of Philosophy, 2007

Dissertation directed by: Professor G.W. Stewart
Department of Computer Science

This thesis is concerned with the solution of large-scale eigenvalue problems. Although there are good algorithms for solving small dense eigenvalue problems, the large-scale eigenproblem has many open issues. The major difficulty faced by existing algorithms is the tradeoff of precision and time, especially when one is looking for interior or clustered eigenvalues.

In this thesis, we present a new method called the residual Arnoldi method. This method has the desirable property that certain intermediate results can be computed in low precision without effecting the final accuracy of the solution. Thus we can reduce the computational cost without sacrificing accuracy. This thesis is divided into three parts. In the first, we develop the theoretical background of the residual Arnoldi method. In the second part, we describe RAPACK, a numerical package implementing the residual Arnoldi method. In the last part, numerical experiments illustrate the use of the package and show the practicality of the method.

Residual Arnoldi Methods : Theory, Package, and Experiments

by

Che-Rung Lee

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2007

ACKNOWLEDGMENTS

I am grateful to all the people who have made this thesis possible and because of whom my graduate experience has been one that I will cherish forever.

First and foremost I'd like to thank my advisor, Professor G. W. Stewart for giving me an valuable opportunity to work on a challenging topic and supporting me financially. I am grateful for the time he has taken from his busy schedule to help me with this dissertation. It has been a pleasure to work with and learn from such an extraordinary individual.

I would also like to thank Professor Dianne P. O'Leary. She always expressed her sympathy when I need the most. Thanks are due to Professor James A. Reggia, Professor Howard C. Elman and Professor John E. Osborn for agreeing to serve on my thesis committee.

I owe my deepest thanks to my family—my mother and father who have always stood by me and guided me through my career and have supported me against impossible odds at times. Words cannot express the gratitude I owe them. My sisters have also given me their support.

I'd like to express my gratitude to the friends from my church and from the Maryland Chinese Bible Study Group. They have treated me as a family member and supported me through their prayers. I would also like to thank my roommates and other friends for enriching my graduate life in many ways. They helped make my journey through graduate school a delightful one.

Finally, thank everyone that I may have failed to acknowledge, and above all thanks to God.

TABLE OF CONTENTS

List of Tables vii

List of Algorithms viii

List of Figures ix

1 Introduction 1
- 1.1 Synopsis 1
- 1.2 Eigenvalue problems 4
  - 1.2.1 Perturbation theory 5
- 1.3 Krylov Subspace methods 6
  - 1.3.1 Krylov subspaces 6
  - 1.3.2 Rayleigh–Ritz method 8
  - 1.3.3 The residual 10
- 1.4 Shift-invert enhancement and subspace restarting 11
  - 1.4.1 Shift-invert enhancement 11
  - 1.4.2 Subspace restarting 11
- 1.5 The residual Arnoldi method 14
  - 1.5.1 Errors in subspace expansion 15
  - 1.5.2 Shift-invert enhancement 16
  - 1.5.3 Switching targets 18
  - 1.5.4 Error control 18
- 1.6 Related work 19
  - 1.6.1 Preconditioning 20
  - 1.6.2 The Jacobi-Davidson method 21
  - 1.6.3 The inexact Krylov method 22

2 Theory 23
- 2.1 Preliminary 23
  - 2.1.1 The RA method 23
  - 2.1.2 The SIRA method 24
  - 2.1.3 Residual and approximation 26
- 2.2 Residual Arnoldi relations 28
- 2.3 The error matrix 30
- 2.4 Convergence of the RA method 32
- 2.5 Convergence of the SIRA method 35

3 RAPACK 42
3.1 Algorithms 43
3.1.1 Algorithm for the RA mode 43
3.1.2 Algorithms for the SIRA mode 46
3.2 Candidate selection, deflation and subspace restarting 46
3.2.1 Candidate selection 46
3.2.2 Deflation 47
3.2.3 Subspace restarting 50
3.3 Complex eigenpairs in real nonsymmetric matrices 50
3.3.1 Subspace expansion 51
3.3.2 Computation of Ritz vectors and residuals 54
3.3.3 Deflation and subspace restarting 55
3.4 Time complexity and memory requirement 55
3.4.1 Time complexity 56
3.4.2 The SIRA mode 58
3.4.3 Memory requirements 59
3.5 Implementation 59
3.5.1 Reverse communication 60
3.5.2 The RA subroutine 61
3.5.3 Program configuration 62
3.5.4 Candidate management 62
3.5.5 Computational information 64
3.6 Integration with other methods 64
3.6.1 Inexact Krylov method 65
3.6.2 Successive inner-outer residual Arnoldi process 67
3.6.3 The complex shift-invert method 68

4 Experiments 70
4.1 Eigentest 70
4.2 ARPACK and RAPACK 71
4.2.1 Mode 1 and the RA mode 72
4.2.2 Mode 3 and the SIRA mode 75
4.3 SRRIT and RAPACK 77
4.4 Inexact Krylov method and the SIRA mode 79
4.5 Complex shift-invert for real eigenproblem 83
4.5.1 Discussion 86

5 Conclusions 88
5.1 Summary 88
5.2 Ongoing and future research 90
5.2.1 The harmonic residual Arnoldi method and the refined residual Arnoldi method 90
5.2.2 The block residual Arnoldi method 92
5.2.3 Other research directions 95

Bibliography 96

LIST OF TABLES

3.1 Algorithmic difference of computational modes 43

3.2 Time complexity for each task in the RA mode. 58

4.1 Mode 1 of ARPACK and mode RA of RAPACK. 73

4.2 Mode 3 of ARPACK and mode SIRA of RAPACK. 75

4.3 Comparison of RAPACK and SRRIT for successive inner-outer process with various number of stages. 80

4.4 Inexact Krylov method, mode 3 of ARPACK, and mode SIRA of RAPACK. 81

4.5 Result of complex shift invert methods. 85

4.6 Computed eigenvalues of complex shift invert methods. 86

LIST OF ALGORITHMS

1.1 One step of the Arnoldi process 8

1.2 The Rayleigh–Ritz method for approximation extraction. . . . 9

1.3 The implicitly restarted Arnoldi method (IRAM). 12

1.4 The Krylov-Schur restarting algorithm. 13

1.5 The subspace expansion step of the residual Arnoldi method. . 14

2.1 The RA iteration. 24

2.2 The SIRA iteration. 25

3.1 The basic algorithm for the RA mode in RAPACK. 45

3.2 The algorithm for candidate computation. 48

3.3 The deflation algorithm in RAPACK. 49

3.4 The algorithm of subspace restarting. 51

3.5 The double vector method in the subspace expansion. 53

5.1 The block residual Arnoldi method. 93

LIST OF FIGURES

1.1 The convergence in four different subspaces. The x-axis denotes iterations, and the y-axis is the projected error $\|x - U_k U_k^* x\|$. 16

1.2 The residual Arnoldi method with shift-invert enhancement . . 18

1.3 residual Arnoldi method with target switch. 19

1.4 residual Arnoldi method with different relative errors 20

3.1 The state transition diagram of the RA subroutine. 63

4.1 The projection error of approximated subspace. 73

4.2 The projection error of approximated subspace. 74

4.3 The number of matrix-vector multiplications for mode 3 and the SIRA mode. 76

4.4 The convergence of mode 3 and the SIRA mode. 77

4.5 Experimental result of RA mode with the successive inner-outer process. (step=10^{-3}) 79

4.6 The precision requirements for the inexact Krylov method, the SIRA mode and the mode 3 of ARPACK. 82

4.7 The convergence of the inexact Krylov method and the mode 3 of ARPACK. 83

4.8 The spectrum of the problem. 84

4.9 Convergence of mode 3 and SIRA mode. 85

4.10 Number of matrix vector multiplications required by mode 3 and SIRA mode. 86

5.1 The residual Arnoldi method, the harmonic residual Krylov method and the refined residual Arnoldi method. The x-axis represent the number of iterations, and y-axis is for projected error. 91

5.2 The block residual Arnoldi method. The x-axis represent the number of iterations, and y-axis is for projected error. 94

Chapter 1. Introduction

This chapter gives an overview of the entire thesis, as well as the necessary background. Section 1.1 is the synopsis of the thesis. Section 1.2 introduces the eigenvalue problem and related definitions and theorems. Section 1.3 and 1.4 describe the basic techniques used for solving large eigenvalue problems. Section 1.5 presents the residual Arnoldi method and illustrates its numerical properties via some simple examples. Related work that has appeared in the literature will be discussed in the last section.

1.1 Synopsis

In scientific computing matrix eigenvalue problems arise naturally from many sources; for example, numerical methods for solving the finite element formulation of the wave equation [10]. A number of problems in statistic and stability analysis can be also reduced to matrix eigenvalue problems; for example, the total least square problem and Markov chain models.

When the matrices in problem are large, methods for dense matrices, such as the QR method [41], may be impractical. First, many large matrices are sparse, and therefore are stored in compact data structures. The methods for dense eigenvalue problems are designed to compute the full eigensystem, which includes all the eigenvalues and eigenvectors. In general, the computed eigenvectors are dense, and therefore it is impractical to store and manipulate them. Second, in some applications, matrices are not even formed explicitly. They are represented by formulas or functions, and the only operation that can be performed on them is matrix-vector multiplication. The decompositions that are required by methods for dense matrices are not suitable for such matrices.

To solve large-scale eigenvalue problems, subspace methods are usually preferable. The idea of a subspace method is to generate a subspace, and extract individual eigenvector approximations from an associated small eigenproblem.

In subspace methods, Krylov subspaces are frequently used. They have two properties that make them popular. First, a Krylov subspace can be generated by using only matrix-vector multiplications. Specifically, given a matrix A and a unit vector u, a Krylov subspace of dimension k is a subspace spanned by vector

$$u, Au, A^2u, \dots, A^{k-1}u.$$

Second, a Krylov subspace usually contains good eigenvector approximations, especially to those whose eigenvalues are on the periphery of the spectrum.

To compute interior eigenvalues (and corresponding eigenvectors), people use a technique called shift-invert enhancement to transform the spectrum of a matrix A. Specifically, consider the matrix

$$S = (A - \sigma I)^{-1}.$$

This matrix has the same eigenvectors as A, but the eigenvalues near σ in A appear on the periphery of the spectrum of S. Note that S does not have to be formed explicitly. Instead, every matrix-vector multiplication $v = Su$ can be effected by solving a linear system $(A - \sigma I)v = u$.

One drawback of Krylov subspace methods is that the matrix-vector product or solutions of a linear equation need be computed exactly. This is an undesirable property because in several applications [8, 34, 43], the cost of matrix-vector multiplication is proportional to the required precision, which means that the more the required accuracy, the more expensive the computation will be. In the case of shift-invert enhancement, this means that the linear systems must be solved to the same accuracy as the desired accuracy of the final eigenpair approximations. If iterative methods such as CG or GMRES [31], are used in solving the linear system, an accurate solution implies a large number of iterations.

In this thesis, we consider a new subspace method for solving large-scale eigenvalue problems, the *residual Arnoldi method*. This method has numerical properties similar to those of Krylov subspace methods when matrix-vector products are computed exactly. However, when there are errors in the computation, classical Krylov subspace methods can fail to find any good approximations, while the residual Arnoldi method can still compute certain target eigenpairs to full accuracy. Moreover, the residual Arnoldi method possesses some special properties which are not shared by ordinary Krylov subspace methods. For example, the residual Arnoldi method can work when starting with an arbitrary subspace.

We also consider an algorithm that integrates shift-invert enhancement and the residual Arnoldi method, called the SIRA method. After computing the residual of a target eigenpair, the SIRA method solves the shifted linear system with the residual as the right hand side. In this case, the linear systems can be solved to a relatively low accuracy, and the target eigenpair approximation will still converge to desired precision.

The goal of this thesis is to get a better understanding of the residual Arnoldi method. In Chapter 2, we supply the theoretical explanation for understanding the special properties of the residual Arnoldi method and the SIRA method. First, we show that there exists an error matrix E_k at iteration k of the algorithm, such that the subspace generated by the residual Arnoldi method with error is the same as a Krylov subspace of the matrix $A + E_k$. Unfortunately, E_k is not small, and we cannot use standard perturbation theory to obtain a convergence proof. However, under reasonable assumptions, it is possible to show that the the norm of the product $E_k x$, where x is the target eigenvector, is proportional to the norm of residual. From this, we can prove the convergence of the residual Arnoldi method.

Chapter 3 describes a numerical package, RAPACK, that implements the residual Arnoldi method and the SIRA method for solving large-scale eigenvalue problems. Besides its ability to tolerate errors in the computation, RAPACK has several desirable features, which come from a combination of techniques such as subspace restarting and reverse communication. The implementation of RAPACK raises additional questions, such as selecting candidate eigenpairs corresponding to the target and the treatment of complex eigenpairs. We will treat these topics along with the analysis of time and storage complexity of RAPACK.

To have better understanding of the performance of the residual Arnoldi method, we have designed several experiments that compare RAPACK with other packages. These results are in Chapter 4. The benchmarks in our experiments are generated by a novel matrix generator, Eigentest, which is specially developed to test software for solving large eigenvalue problems. We compare RAPACK with ARPACK and SRRIT. When compared with ARPACK, RAPACK shows a slight disadvantage in the RA mode, but is much better than ARPACK in the SIRA mode. RAPACK also outperforms SRRIT when we compare the ability of using existing subspace to initial the process. Other experiments, concerning inexact matrix-vector multiplication and complex shift-invert enhancement, are also included in this chapter.

This thesis, of course, has not solved all problems about this method.

Many special properties of the residual Arnoldi method need further investigation. In addition, several new directions for future work have emerged from this study. We summarize them in the last chapter.

1.2 Eigenvalue problems

Let A be an $n \times n$ matrix. A pair (λ, x) that consists of a scalar λ and a nonzero vector x is called a *right eigenpair* of A if it satisfies

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

The scalar λ is called an *eigenvalue* of A, and x is the corresponding *eigenvector*. Similarly, a *left eigenpair* (λ, y) of A is defined by the equation

$$y^* A = \lambda y^*, \tag{1.2}$$

where y^* is the conjugate transpose of y. Unless otherwise stated, the term eigenpair will indicate a right eigenpair. In addition, all eigenvectors are assumed to be normalized. In this thesis, the norm used is 2-norm.

If A has n linearly independent eigenvectors, we can write

$$A = X \Lambda X^{-1}, \tag{1.3}$$

where X contains n eigenvectors of A in its columns and Λ is a diagonal matrix containing the corresponding eigenvalues [41, p.9]. If A is Hermitian, i.e. $A = A^*$, then X can be taken to be a unitary matrix, so that the eigenvectors of A are pairwise orthogonal.

The set of eigenvalues of a matrix A is called the *spectrum* of A, denoted by $\Lambda(A)$. Eigenvalues need not be distinct. A distinct eigenvalue is called a *simple* eigenvalue, while a repeated eigenvalue is called a *multiple* eigenvalue. In the latter case, the eigenvectors of repeated eigenvalues are not necessarily unique, but the subspace they span is.

A *similarity transformation* of A is a mapping that transforms A into another matrix $B = U^{-1}AU$, where U is a nonsingular matrix. The matrices A and B are said to be *similar*. The matrix B has the same spectrum as A; and for each eigenvector x of A, B has a corresponding eigenvector $U^{-1}x$.

It can be shown that for any matrix A, there exists a unitary matrix U, such that U^*AU is an upper triangular matrix T [41]. This relation

$$A = UTU^* \tag{1.4}$$

is called a *Schur decomposition.* The columns of U are called *Schur vectors* and the diagonal elements of T are called *Schur values.* The Schur decomposition of a matrix is not unique, because the diagonal elements of T can be made to appear in any order of the diagonal of T.

A real nonsymmetric matrix can have complex eigenpairs. If (λ, x) is a complex eigenpair of A, then its conjugate $(\bar{\lambda}, \bar{x})$ is also an eigenpair of A. On the other hand, all eigenvalues of a real symmetric matrix A are real.

1.2.1 Perturbation theory

In this subsection, we will introduce the *first order perturbation theory* of a simple eigenpair - i.e. an eigenpair whose eigenvalue has multiplicity one.

Let $\tilde{A} = A + E$ be the perturbed matrix for some error matrix E. Let (λ, x) be a simple eigenpair of A and (λ, y) be the corresponding left eigenpair. According to [42, chapter4, theorem 2.3], for all sufficiently small E, there exists a unique eigenvalue $\tilde{\lambda}$ of $\tilde{A}$ such that

$$\tilde{\lambda} = \lambda + \frac{y^* E x}{y^* x} + O(\|E\|^2). \tag{1.5}$$

It follows that

$$|\tilde{\lambda} - \lambda| \leq \frac{\|y\|\|x\|\|E\|}{|y^* x|} + O(\|E\|^2).$$

The quantity

$$\frac{\|y\|\|x\|}{|y^* x|} \tag{1.6}$$

is called the *condition number* of the eigenvalue λ, because it indicates the sensitivity of λ to a small perturbation of A.

Let $\tilde{x}$ be the eigenvector of $\tilde{A}$ corresponding to $\tilde{\lambda}$. We will measure the difference of x and $\tilde{x}$ by the angle between them. The first order perturbation theory of eigenvectors [41] shows that

$$|\sin \angle(x, \tilde{x})| \leq \frac{\|E\|}{\mathrm{sep}(\tilde{\lambda}, L)}. \tag{1.7}$$

Here L is the matrix defined by

$$L = X_\perp^* A X_\perp, \tag{1.8}$$

where $X_\perp$ is an orthonormal basis for the orthogonal complement of the space spanned by x and

$$\operatorname{sep}(\tilde{\lambda}, L) = \|(\tilde{\lambda} I - L)^{-1}\|^{-1}. \tag{1.9}$$

The function $\operatorname{sep}(\mu, L)$ is a lower bound on the separation of μ and the spectrum of L. Specifically, in [41, Theorem 3.14]

$$\operatorname{sep}(\mu, L) \le \min_{\lambda \in \Lambda(L)} |\mu - \lambda|. \tag{1.10}$$

From (2.9) it is not hard to see that the sep function is continuous with respect to its arguments. In fact, one can prove that [41, p.54]

$$|\operatorname{sep}(\mu + \epsilon, L + F) - \operatorname{sep}(\mu, L)| \le |\epsilon| + \|F\|.$$

Equation (1.7) shows that $\operatorname{sep}(\tilde{\lambda}, L)^{-1}$ defines the condition number for the eigenvector x.

1.3 Krylov Subspace methods

Krylov subspace methods are frequently used to compute a few selected eigenpairs of a large matrix [41, 31]. The advantage of such methods is that no decomposition of the original matrix required. A typical Krylov subspace method consists of three steps: subspace expansion, Rayleigh–Ritz approximation, and convergence testing. We will discuss each in the following subsections.

1.3.1 Krylov subspaces

Given a matrix A and a nonzero vector u_1, the k-th order Krylov subspace $\mathcal{K}_k(A, u_1)$ is $\operatorname{span}\{u_1, Au_1, A^2u_1, \cdots, A^{k-1}u_1\}$. As can be seen, the generation of a Krylov subspace only needs matrix-vector multiplications, which is computationally desirable for large and sparse matrix.

An important property of Krylov subspaces is that the sequence of subspaces may contain increasingly accurate approximations to certain eigenvectors of A. In [30] Saad gives a convergence theory for these eigenvectors when their eigenvalues lie on the periphery of the spectrum. Specifically, the tangent of the angle between an eigenvector x and its best approximation contained in $\mathcal{K}_k(A, u_1)$ is shown to decrease at least linearly with the expansion of subspaces. The proof uses the fact that any vector v in $\mathcal{K}_k(A, u_1)$

can be represented as $p_{k-1}(A)u_1$ for some $k-1$th order polynomial p_{k-1}. Therefore, the problem of identifying the best approximation of an eigenvector x in the subspace $\mathcal{K}_k(A, u_1)$ is the same as finding a polynomial p_{k-1} such that $|\tan\angle(x, p_{k-1}(A)u_1)|$ is minimized. Equivalently, if λ is the eigenvalue corresponding to x, the polynomial p_{k-1} satisfying

$$\min_{\substack{\deg(p_k)\le k\\ p_k(\lambda_i)=1}} \max_{\substack{\mu\in\Lambda(A)\\ \mu\ne\lambda}} |p_k(\mu)| \tag{1.11}$$

produces the best approximation $p_k(A)u_1$ to x.

The solution of this minimization problem does not have a simple analytical form in general. However, upper bounds can be obtained using Chebyshev polynomials. To simplify the discussion, we only present the result for the dominant eigenpair here. Let A be a symmetric matrix of order n with eigenvalues

$$\lambda_1 > \lambda_2 \ge \cdots \ge \lambda_n,$$

and corresponding eigenvectors $x_1, x_2, \cdots, x_n$. Let

$$\eta = \frac{\lambda_1 - \lambda_2}{\lambda_2 - \lambda_n}.$$

Let v be the best approximation of x_1 in $\mathcal{K}_k(A, u_1)$. Then

$$\tan\angle(x_1, v) \le \frac{\tan\angle(x_1, u_1)}{c_{k-1}(1+2\eta)}, \tag{1.12}$$

where c_{k-1} is the Chebyshev polynomial of degree $k-1$.

It can be shown that when $|t|>1$,

$$c_k(t) = (1+\sqrt{t^2-1})^k + (1+\sqrt{t^2-1})^{-k}. \tag{1.13}$$

Therefore, equation (1.12) implies at least the linear convergence of the best approximation in $\mathcal{K}_k(A, u_1)$ to the dominant eigenvector of A. In practice, the convergence is often mildly superlinear.

The classical algorithm for Krylov subspace expansion is the Arnoldi process, which generates orthonormal bases for the sequence of Krylov subspaces. Suppose $U_k = (u_1\ u_2\ \cdots\ u_k)$ is an orthogonal basis for $\mathcal{K}_k(A, u_1)$. The following algorithm explains how the Arnoldi process expands U_{k+1} for $\mathcal{K}_{k+1}(A, u_1)$ from U_k.

Algorithm 1.1 One step of the Arnoldi process

```
1. Compute $v_k = Au_k$.
2. Orthogonalize $v_k$ against $U_k$ and normalize the result to $u_{k+1}$.
3. $U_{k+1} = \begin{pmatrix} U_k & u_{k+1} \end{pmatrix}$.
```

In the above algorithm the orthogonalization step can be represented as

$$Au_k = U_k h_k + \beta_k u_{k+1}, \tag{1.14}$$

where $h_k = U_k^* A u_k$ and $\beta_k = \|(I_k - U_k U_k^*)Au_k\|$. If we let

$$\hat{h}_i = \begin{pmatrix} h_i \\ \beta_i \\ 0_{k-i-1} \end{pmatrix},$$

and $H_k = \begin{pmatrix} \hat{h}_1 & \cdots & \hat{h}_{k-1} & h_k \end{pmatrix}$, then we have the *Arnoldi relation* [41, ch.5]:

$$AU_k = U_k H_k + \beta_k u_{k+1} e_k^*. \tag{1.15}$$

Note that H_k is an upper Hessenberg matrix of order k. In fact, H_k is a special matrix, called *Rayleigh quotient*. We will discuss its properties in the next subsection.

1.3.2 Rayleigh–Ritz method

The Rayleigh–Ritz method is the most commonly used method for extracting eigenvector approximations from a subspace $\mathcal{U}$. Algorithm 1.2 describes Rayleigh–Ritz method for obtaining an approximation to the eigenpair (λ, x) of A.

The matrix H is called the *Rayleigh quotient*. If the dimension of $\mathcal{U}$ is k, then H is a $k \times k$ matrix. When k is much smaller than the order of A, the eigenvalue problem of H can be easily solved by dense matrix algorithms, such as the QR algorithm [41]. The eigenpair approximation (μ, Uy) is called a *Ritz pair*. Specifically, the scalar μ is called a *Ritz value*, and the vector Uy is called a *Ritz vector*. The method can be justified informally by observing

Algorithm 1.2 The Rayleigh–Ritz method for approximation extraction.

```
1. Let $U$ be an orthogonal basis of $\mathcal{U}$.
2. Compute $H = U^*AU$.
3. Compute an eigenpair $(\mu, y)$ of $H$ such that $\mu \sim \lambda$.
4. Return $(\mu, Uy)$ as the approximation.
```

that if $\mathcal{U}$ contains x then H will have eigenpair (λ, U^*x) and the Ritz pair $(\lambda, UU^*x) = (\lambda, x)$ recovers the eigenpair of A.

In [20], Jia and Stewart provided a perturbation analysis of Ritz pairs, which explains under what circumstances this method works. As long as the subspace $\mathcal{U}$ contains good eigenvector approximations, the Rayleigh–Ritz method will produce accurate eigenvalue approximations. Specifically, the eigenvalue λ of A is an eigenvalue of the matrix $H + E$ for some matrix E bounded by

$$\|E\|_2 \leq \tan\theta \|A\|_2, \tag{1.16}$$

where θ is the angle of the eigenvector x and the best approximation in $\mathcal{U}$. By a theorem of Elsense [11], as θ converges to zero, an eigenvalue of H will converge to λ.

The bound for the Ritz vector is more complicated. A theorem in [39] says

$$\sin\angle(x, Uy) \leq \sin\theta\sqrt{1 + \frac{\|h\|^2}{\operatorname{sep}(\lambda, N)^2}}, \tag{1.17}$$

where h and N are defined in the equation

$$\begin{pmatrix} y \\ Y_\perp \end{pmatrix} H(y, Y_\perp) = \begin{pmatrix} \mu & h \\ 0 & N \end{pmatrix}.$$

Equation (1.17) implies the Ritz vector may fail to be accurate when $\operatorname{sep}(\lambda, N)$ is small, even when the best approximation in $\mathcal{U}$ is close to x.

When the subspace $\mathcal{U}$ is a Krylov subspace generated by the Arnoldi process, the Rayleigh quotient can be constructed implicitly [31]. From the Arnoldi relation,

$$AU_k = U_kH_k + \beta_k u_{k+1},$$

it can be shown that the Rayleigh quotient is

$$\begin{aligned} U_k^* A U_k &= U_k^*(U_k H_k + \beta_k u_{k+1}) \\ &= H_k. \end{aligned}$$

As will be shown in (2.13), H_k can be obtained from the orthogonalization process.

1.3.3 The residual

The quality of an approximate eigenpair is usually measured by its residuals. Let (μ, z) be an approximation to an eigenpair (λ, x) of A, with $\|z\| = 1$. The residual of (μ, z) is defined as

$$r = Az - \mu z. \tag{1.18}$$

If (μ, z) is an eigenpair of A, r is a zero vector and conversely. Otherwise, the norm of r represents the size of a backward error. Specifically, in [41, Theorem 1.3], it is shown that there is a matrix $E = rz^*$ whose norm is $\|r\|$ such that (μ, z) is an eigenpair of the matrix $A - E$. In other word, a small residual implies that (μ, z) is an eigenpair of a small perturbation of A.

When the approximation is computed by the Rayleigh–Ritz method, and the subspace is generated by the Arnoldi method, the residual can be obtained from the Arnoldi relation. Let $(\mu_k, U_k y_k)$ be a Ritz pair computed from an orthogonal basis U_k of $\mathcal{K}_k(A, u_1)$. The Arnoldi relation for A and U_k is

$$AU_k = U_k H_k + \beta_k u_{k+1} e_k^*.$$

Hence,

$$\begin{aligned} r_k &= AU_k y_k - \mu_k U_k y_k \\ &= (U_k H_k + \beta_k u_{k+1} e_k^*) y_k - \mu_k U_k y_k \\ &= U_k(H_k y_k - \mu_k y_k) + \beta_k u_{k+1} e_k^* y_k \\ &= \beta_k \eta_k u_{k+1}, \end{aligned}$$

where η_k is the last element of y_k. Therefore, the norm of the residual can be estimated by $\beta_k|\eta_k|$. Note that this estimation can diverge significantly from the actual value, $\|AU_k y_k - \mu_k U_k y_k\|$, because of the truncation error [7].

1.4 Shift-invert enhancement and subspace restarting

In this section, we will introduce two common techniques in the solution of large-scale eigenvalue problems. The first is shift-invert enhancement, which speeds up convergence by transforming the spectrum of the original matrix in such as way that the desired eigenvalue is well separated from its neighbors. The second technique, used for memory management, is subspace restarting, which shrinks the dimension of the subspace when it reaches a prescribed limit.

1.4.1 Shift-invert enhancement

Shift-invert enhancement [41, p.66] replaces the original matrix A with a shifted and inverted matrix $S = (A - \sigma I)^{-1}$. This matrix S has the same eigenvectors as A, but differs in its eigenvalues. Specifically, (λ, x) is an eigenpair of A if and only if $((\lambda - \nu)^{-1}, x)$ is an eigenpair of S.

The choice of σ depends on which eigenvalues are desired. When σ is chosen to be near a particular eigenvalue λ, the transformed eigenvalue $\hat{\lambda} = (\lambda - \sigma)^{-1}$ will be large. Moreover, if σ is close enough to λ, then $\hat{\lambda}$ will be well separated in the transformed spectrum, which will speed up the convergence of the Arnoldi process.

A problem arises when shift-invert enhancement is used to find complex eigenvalues of real nonsymmetric matrices. A complex shift value will convert the arithmetic from real plane to complex plane. In [28], Parlett and Saad proved that the real part or the imaginary part of a complex shift-inverted matrix can be used alone for subspace expansion. Let $\sigma = \alpha + i\beta$ be a complex shift value. Then the shift-inverted matrix S is complex. Let $S = S_r + iS_i$, where S_r and S_i are real matrices. They proved that

$$\begin{aligned} S_r &= (A - \alpha I)(A - \sigma I)^{-1}(A - \bar{\sigma} I)^{-1} \\ S_i &= \beta(A - \sigma I)^{-1}(A - \bar{\sigma} I)^{-1}. \end{aligned}$$

Although S_r differs from $[(A - \sigma I)(A - \bar{\sigma} I)]^{-1}$, empirically, both matrices have similar performance.

1.4.2 Subspace restarting

Memory management is critical for Krylov subspace methods. As the subspace is expanded, the required storage may exhaust the available memory.

At some point, therefore, one must restart the Krylov sequence. The problem here is to truncate the basis while preserving information required for continuing convergence. This subsection will introduce two subspace restarting algorithms that overcome this difficulty.

The first algorithm is the implicitly restarted Arnoldi method (IRAM) [36], which uses the implicit-shift QR algorithm [41] on the Rayleigh quotient to filter out unwanted approximations. At the same time, it maintains the Arnoldi relation for the restarted subspace so that the Arnoldi process can be continued. Algorithm 1.3 sketches its process.

Algorithm 1.3 The implicitly restarted Arnoldi method (IRAM).

1. Let U_m be the subspace to be restarted that satisfies the Arnoldi relation.
$$AU_m = U_m H_m + \beta_m u_{m+1} e_m^*.$$
2. Let $\mu_1, \mu_2, \cdots, \mu_m$ be the eigenvalues of H_m and assume $\mu_1, \mu_2, \cdots, \mu_k$ are desired.
3. Let p be the filtering polynomial such that $p(\mu_j)$ is small for $j = k{+}1..m$ compared to $p(\mu_1), \cdots p(\mu_k)$.
4. Run the implicit-shift QR (ISQR) algorithm to produce $p(H_m) = QR$ such that
   (a) $p(H_m)$ is upper Hessenberg and
   (b) the first k subdiagonal elements of $p(H_m)$ are not too small.
5. The restarted subspace $U_k = U_m Q(:, 1:k)$.

Morgan [26, Theorem 3] points out that the subspace, generated by IRAM, contains the Krylov subspace $\mathcal{K}_k(A, z_i)$ for each desired Ritz vector z_i. This nice property keeps all the desired approximations converging in the new subspace. Recently, Lehoucq showed a connection between IRAM and subspace iteration [24]. The convergence of IRAM has also been established [5, 6].

The second algorithm is Krylov–Schur restarting, proposed by Stewart [40], which employs the Krylov decomposition, a generalization of the Arnoldi relation, and the Schur decomposition in the construction of restarting subspace. A Krylov decomposition of order k is defined by

$$AU_k = U_k B_k + u_{k+1} b_{k+1}^*, \tag{1.19}$$

where U_k is an orthogonal basis of a subspace and B_k is the Rayleigh quotient. Equation (1.19) generalizes the Arnoldi relation by allowing B_k and b_{k+1}^* to be arbitrary. This generalization makes the Krylov decomposition of a subspace invariant under two transformations: similarity transformations and translations. A similarity translation is obtained by postmultiplying (1.19) by a nonsingular matrix Q,

$$A(U_k Q) = (U_k Q)(Q^{-1} B_k Q) + u_{k+1}(b_{k+1}^* Q). \tag{1.20}$$

A translation shifts u_{k+1} by a vector $U_k a$,

$$AU_k = U_k(B_k + a b_{k+1}^*) + \tilde{u}_{k+1}(\gamma b_k^*), \tag{1.21}$$

where γ and $\tilde{u}_{k+1}$ are defined by

$$\gamma \tilde{u}_{k+1} = u_{k+1} - U_k a,$$

and $\tilde{u}_{k+1}$ is orthonormal to U_k.

The algorithm of the Krylov-Schur restarting is sketched in algorithm 1.4

Algorithm 1.4 The Krylov-Schur restarting algorithm.

1. Given a Krylov-Schur decomposition

$$AU_m = U_m B_m + u_{m+1} b_{m+1}^*.$$

2. Compute a Schur decomposition of B_m, $B_m = Q_m^* T_m Q_m$ such that the desired Ritz values are the first k diagonal elements of T_m.

3. The restarted subspace is $U_k = U_m Q_m(:, 1 : k)$.

A block version of the Krylov-Schur restarting algorithm has also been proposed by Zhou and Saad [46].

1.5 The residual Arnoldi method

Under certain circumstance, the Krylov approximations may stop converging. This situation is called the *stagnation.* For example, when an error ϵ in u_{k+1} is introduced at an early stage of computation, the convergence will stagnate at the precision level of ϵ. In other words, Krylov subspace methods demand computations to be full accuracy, at least at the beginning of process.

This precision requirement is undesirable in many occasions, especially when the cost of matrix vector multiplication increases with the desired precision. This happens, for instance, when shift-invert enhancement is applied. In many applications, the linear system may be too large to solve directly. In that case, iterative methods, like CG or GMRES [29, 16], are used. As can be imagined, the cost of these iterative methods increases with the required precision.

The residual Arnoldi (RA) method[1] uses the residuals of a selected approximate eigenpair in subspace generation. The algorithm that expands a given orthonormal basis U_k is sketched in Algorithm 1.5.

Algorithm 1.5 The subspace expansion step of the residual Arnoldi method.

```
1. Compute the Rayleigh quotient, $H_k = U_k^* A U_k$, and its eigendecomposition.
2. Select an eigenpair $(\mu_k, y_k)$ from $H_k$ and compute the Ritz pair $(\mu_k, U_k y_k)$.
3. Compute the residual $r_k = AU_k y_k - \mu_k U_k y_k$.
4. Orthogonalize $r_k$ against $U_k$ and normalize the result to be $u_{k+1}$.
5. $U_{k+1} = (U_k \ \ u_{k+1})$.
```

The approximation $(\mu_k, U_k y_k)$ selected in step 2 is called a *candidate*, and the eigenpair that candidate approximates is called a *target.* Theoretically, the subspace generated by the residual Arnoldi method is identical to that generated by the Arnoldi process. However, when there are errors, the

[1]The phenomena underlying the residual Arnoldi method were discovered by Sleipen, Stewart and Van der Vorst in 2001.

convergence of the candidates do not stagnate.

In this section, we will illustrate the numerical properties of the RA method by examples. The first example compares the RA method and the Arnoldi process with and without errors. The second example illustrates how shift-invert enhancement can be integrated into the RA method. The third example demonstrates the ability of the RA method to converge from an initial subspace that contains good Ritz approximations. The last example discusses the effects of the size of the errors.

1.5.1 Errors in subspace expansion

Let A be a 100×100 general matrix with eigenvalues $(1, 0.95, \cdots, 0.95^{99})$ and randomly generated eigenvectors[2]. Four different subspaces are expanded with the same random initial vector. The first two subspaces are generated by the Arnoldi process, but the second subspace has relative errors 10^{-3} in each vector used to expand the subspace. The third and fourth subspaces are expanded by the residual Arnoldi method. Similarly, relative errors 10^{-3} are added in each iteration in the fourth case. For the residual Arnoldi method, the dominant Ritz pair is selected as a candidate.

Figure 1.1 shows convergence results for these four subspaces. The y-axis represents the projected error, which is defined as $\|x - U_k U_k^* x\|$ where x is the dominant eigenvector and U_k is the orthogonal basis of the generated subspace. The solid line in the figure is for the dominant eigenpair; the dashed line is for the subdominant eigenpair.

In figure 1.1(a), both approximations converge to machine precision. When errors are added, in figure 1.1(b), both approximations stagnate around 10^{-3}. Figure 1.1(c) displays essentially the same convergence as figure 1.1(a), except that the subdominant eigenvector levels off toward the end of process. The reason is that the residual used in subspace expansion becomes nothing but rounding errors once the dominant approximation reaches machine precision. In figure 1.1(d), the subdominant approximation stagnates as it does in figure 1.1(b); however, the dominant approximation converges to machine precision as it does when no errors are added.

[2]The examples given here are from the private communication of G.W. Stewart.

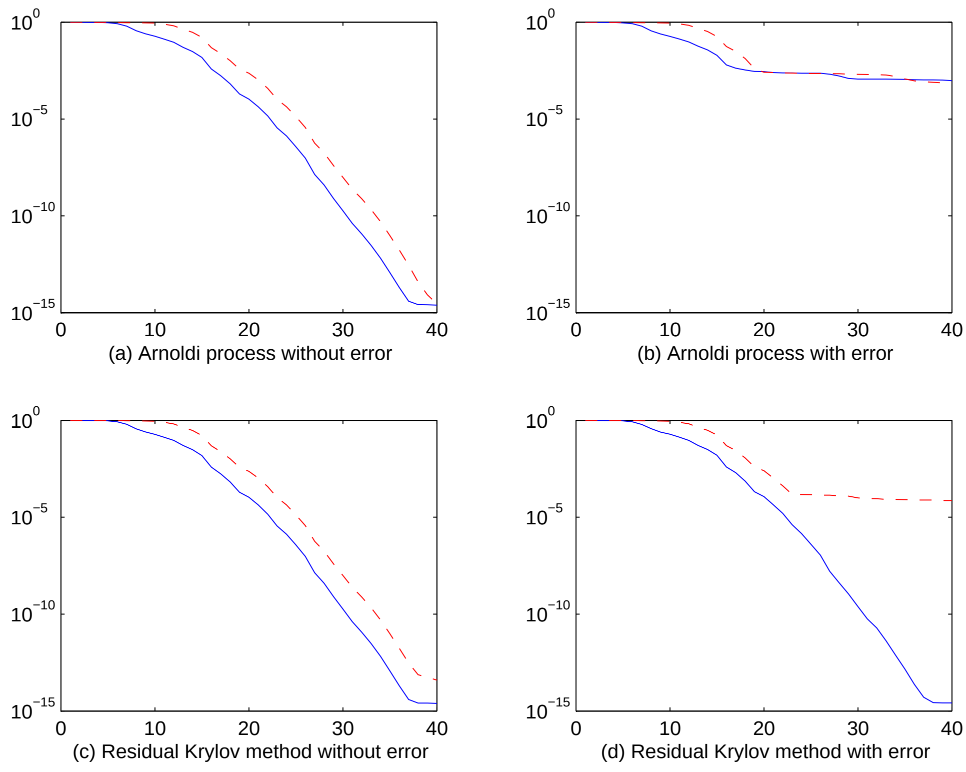

Figure 1.1: The convergence in four different subspaces. The x-axis denotes iterations, and the y-axis is the projected error $\|x - U_k U_k^* x\|$.

1.5.2 Shift-invert enhancement

The example suggests the possibility that in shift-invert enhancement the resulting linear system can be solved to less than full accuracy. Let $S = (A - \sigma I)^{-1}$. The residual of a candidate (ν, z) from RA applied to S is

$$r_S = Sz - \nu z = (A - \sigma I)^{-1} z - \nu z. \tag{1.22}$$

Let ϵ be the error in r_S; i.e., the precision of the computed r is bounded by $\epsilon \|r\|$. At the beginning, when $\|r\|$ is large, the linear system $(A - \sigma I)v = z$ can be solved with low accuracy. However, as the approximation converges, $\|r\|$ becomes smaller and smaller, and the system must be solved to higher and higher accuracy. Thus, the direct application of the RA method does little to reduce the computational costs.

Here we consider an alterative approach to integrate shift-invert enhancement with the residual Arnoldi method. First, we compute a candidate Ritz pair using the original matrix A, say (μ, Uy), and its residual $r_A = AUy - \mu Uy$. Then we solve the equation

$$(A - \sigma I)v = r_A \tag{1.23}$$

and use v in subspace expansion.

This method works because v is equivalent to the residual r_S in (1.22), up to a scalar. First, it is easy to show that if (μ, Uy) is an eigenpair approximation of A, then

$$(\nu, z) = \left(\frac{1}{\mu - \sigma}, Uy\right)$$

is an eigenpair approximation to an eigenpair of S (though (μ, Uy) is not a Ritz pair of S). Specifically,

$$\begin{aligned} v &= Sr_A \\ &= S(AUy - \mu Uy) \\ &= S[(A - \sigma I)Uy + (\sigma - \mu)Uy] \\ &= Uy + (\sigma - \mu)SUy \\ &= (\sigma - \mu)[SUy - (\mu - \sigma)^{-1}Uy] \\ &= (\sigma - \mu)r_S. \end{aligned} \tag{1.24}$$

Mathematically, v should be parallel to the vector generated by the r_S. However, this method has the advantage that the linear systems can be solved imprecisely, because the error produced from this system is always proportional to $\|r\|$. Therefore, no matter how small the residual is, the system can be solved to a constant precision. This method is called the *shift-invert residual Arnoldi method* (SIRA).

Figure 1.2 plots the course of a run of the SIRA method. The experimental setting is the same as before, and the shift value is 1.3 in the shift-invert enhancement. The linear systems are solved to relative precision 10^{-3} in every iteration. As can be seen, the convergence of the selected approximation, the dominant one, is speeded up by the shift-invert enhancement, but the subdominant approximation, owing to the imprecise solutions, stagnates at the level around 10^{-3}.

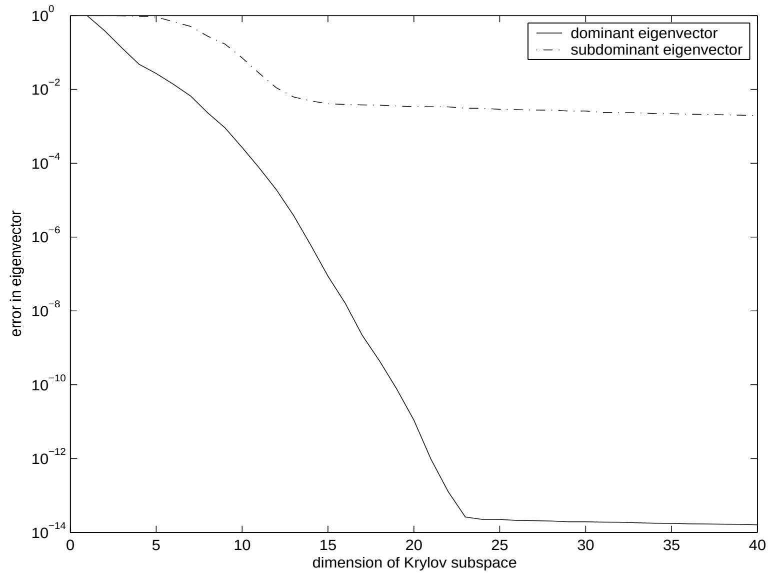

Figure 1.2: The residual Arnoldi method with shift-invert enhancement

1.5.3 Switching targets

In the previous experiments, the residual Arnoldi method exhibited convergence only for the candidate when errors are introduced. The following example will examine the situation when the candidate is changed in the middle of the process. The same example is used in the experiment. First, we choose the approximations to the dominant eigenpair as a candidate, and add errors of size 10^{-3} during the computation. At iteration 30, we switch the candidate to the approximation of the subdominant eigenpair, and observe the change of convergence.

Figure 1.3 displays the experimental result, in which the approximation of the subdominant eigenpair stagnates at the level slightly below 10^{-3} before iteration 30. However, after the switching, the new candidate starts to converge. On the other hand, the approximation of the dominant eigenpair starts to level off after iteration 30.

1.5.4 Error control

The size of relative error in the residual Arnoldi method influences the convergence rate of the selected approximation, at least up to a point. If the

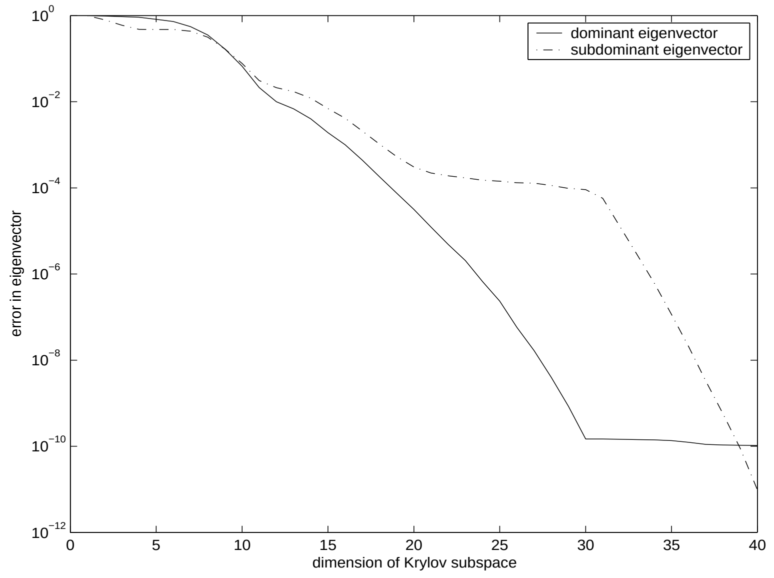

Figure 1.3: residual Arnoldi method with target switch.

relative error is large, the convergence will be slow. As the error decreases, the convergence rate increases — up to a certain point, after which reducing the error has no effect on the convergence. Figure 1.4 compares the convergence of three different error levels: 1, 10^{-1} and 10^{-3}. As can be seen, when the relative error is 1, the convergence is slow; when the error decreases, the convergence becomes faster. But after the error reaches 10^{-3}, the convergence remains the same. Even we decrease the error size, the convergence cannot be made faster.

1.6 Related work

In this section, we will survey some related methods. First, the technique called preconditioning [25] is introduced. Then the Jacobi-Davidson method [35, 14] for solving large eigenvalue problems will be discussed. Finally, a method that relaxes the precision requirement of the Arnoldi process, the inexact Krylov method, is presented in our discussion.

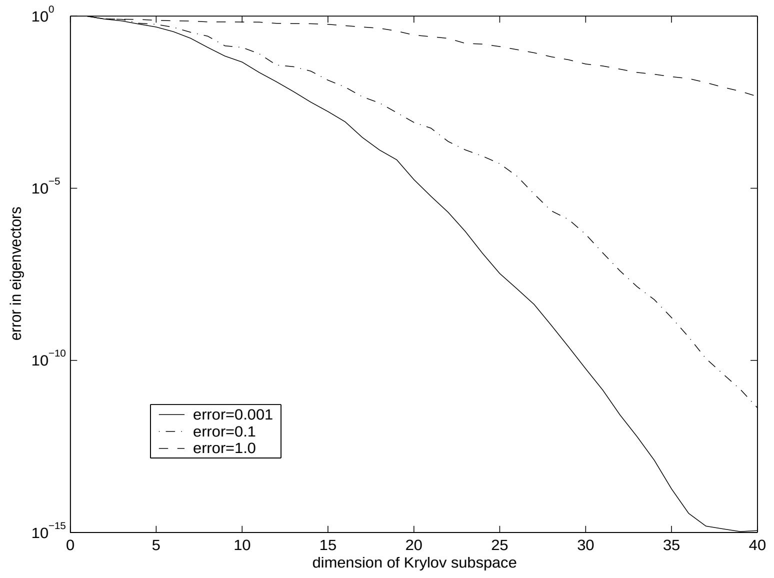

Figure 1.4: residual Arnoldi method with different relative errors

1.6.1 Preconditioning

Preconditioning is a widely used technique for solving linear systems and eigenvalue problems that transforms the original problem to an easier one. To be meaningful, the transformation itself should be simple and the transformed problem should be in the same solution space.

For large eigenvalue problems, the commonly used preconditioning is the *Cayley transformation*, which transforms the original matrix A to a matrix T_c,

$$T_c = (A - \sigma I)^{-1}(A - \mu I), \tag{1.25}$$

where σ is called the *pole* and μ is the *zero*. The Cayley transformation changes each eigenvalue λ_i of A to the eigenvalue

$$\zeta_i = \frac{\lambda_i - \mu}{\lambda_i - \sigma},$$

of T_c. The pole can be chosen so that the ζ_i corresponding to the desired eigenvector is large compared to others. In this transformation, one must solve a linear equation for $(A - \sigma I)^{-1}$. This linear system, however, can be solved inexactly and still get some desired eigenpairs converging. The preconditioning that uses the Cayley transformation with inexact linear solving is called the *inexact Cayley transformation*.

The inexact Cayley transformation has been used in many algorithms [2]. For instance, in the integration with the Arnoldi process, the transformed matrix T_c is multiplied to the last generated basis vector instead of A.

The inexact Cayley transformation for the Arnoldi process has many similarities with the shift-invert residual Arnoldi method, which uses the vector v, generated from $(A - \sigma I)v = (A - \mu I)Uy$, in subspace expansion. First, both methods use some kind of shift-invert enhancement on a shifted matrix, and the linear system can be solved inexactly. Second, the convergence for both methods is local, which means only one or few eigenpairs, whose eigenvalue is in proximity to σ, can converge at a time. Other eigenpairs will stagnate at the level of the precision used in solving linear systems.

However, there are significant differences between these two methods. First, the pole and zero defined in the inexact Cayley transformation cannot be changed without totally restarting the subspace, because the Arnoldi process requires to keep a Krylov subspace for T_c. Second, the choice of the pole and zero must satisfy a lot of constraints to make the inexact Cayley transformation work, as described in [2, chapter 11]. These problems do not occur in the SIRA method.

1.6.2 The Jacobi-Davidson method

The Jacobi-Davidson method, proposed by Sleijpen and Van der Vorst [35], is a Newton–based method, which improves the existing approximation, generated from the Rayleigh–Ritz method, by orthogonal corrections. Specifically, let (μ, z) be the current approximation and r be its residual. The orthogonal correction v is computed by solving the following equation,

$$(I - zz^*)(\tilde{A} - \mu I)(I - zz^*)v = -r, \tag{1.26}$$

where $\tilde{A}$ is a perturbation of A. The orthogonal correction v is then used in the subspace expansion.

The Jacobi-Davidson method and the shift-invert residual Krylov method are alike in certain aspects. Algorithmically, both methods select an approximation and use its residual in subspace expansion. Moreover, the residual in each method is enhanced by some sort of shift-invert matrix. Numerically, both methods allow only one eigenpair to converge at a time when errors are introduced, and the candidate can be switched at any time without fully restarting the subspace.

The most significant differences between these two methods is the Jacobi-Davidson method requires an orthogonal correction for the current approximation. Analytically, the convergence of the Jacobi-Davidson method is supported by the analysis of Newton's method, while the shift-invert residual Arnoldi method is studied through the classical analysis of Krylov subspace, as we shall see in Chapter 2.

1.6.3 The inexact Krylov method

Recently, the inexact Krylov method, which relaxes the precision requirement of the Arnoldi process, has been studied in [13, 34, 18]. In the inexact Krylov method, the matrix-vector multiplication can be computed more and more inaccurately as approximations converge. Although this property has been analyzed and applied to solving large linear system, the similar phenomenon has been observed in solving eigenvalue problems [18].

In [34] and [13], the inexact Krylov method for linear system solvers, such as the Generalized Minimal Residual Method (GMRES) [32], is analyzed through the the concept of the *residual gap*. The residual of a linear system $Ax = b$ is defined as $r = b - Az$, where z is the computed solution. The residual gap compares two kinds of residuals: One is the *true residual* r_m, given from the definition; another is the *computed residual* $\tilde{r}_m$, produced by the inexact Krylov method. The definition of the residual gap δ is given as follows,

$$\delta = \|r_m - \tilde{r}_m\|. \tag{1.27}$$

It is easy to show that the norm of the computed residual is bounded by $\|r_m\| + \delta$. As long as δ is small, the convergence of the exact method will guarantee a small computed residual. On the other hand, if δ is large, even when the real residual converges to zero, the computed residual need not.

The conclusion is that as the process converges, matrix-vector multiplication may be computed with decreasing accuracy without increasing the residual gap.

The convergence properties of the inexact Krylov method look very different from those of the residual Arnoldi method's. In the inexact Krylov method the relative error decreases as the approximations converge; in the residual Arnoldi method it increases.

Chapter 2. Theory

As shown in the examples in Chapter 1, the residual Arnoldi (RA) method and the shift-invert residual Arnoldi (SIRA) method have many nice numerical properties, especially when there are errors in the computation. This chapter is devoted to the convergence theory of these two methods. First, the algorithms for both methods and some preliminary results are presented in section 2.1. Section 2.2 derives the residual Arnoldi relation, which reveals some basic properties of the RA method. In addition, a backward error analysis is provided. Section 2.3 discusses the numerical properties of the backward error though a special example. Section 2.4 completes the analysis with the convergence theorem for the RA method; and a similar result for the SIRA is given in section 2.5.

2.1 Preliminary

In this section, we will introduce the RA and the SIRA algorithms, as well as the notation that will be used in this chapter. In addition, some preliminary results about the residual and the approximation and assumptions needed for the analysis will be presented.

The problem that we are concerned with is the eigenproblem of a nonsymmetric matrix A. We assume that the norm of A is always one, and that all eigenvectors and eigenvector approximations are normalized. Without further specification, the norm used in this Chapter will be the 2-norm.

2.1.1 The RA method

We begin with the RA method. Algorithm 2.1 sketches the procedure of one iteration of the RA method, which takes an orthonormal basis U, and expands it into a one dimension larger orthonormal basis.

The first two steps are just the standard Rayleigh–Ritz procedure. In step 2, the selected Ritz pair is called a *candidate*, and the eigenpair of A

Algorithm 2.1 The RA iteration.

```
1. Compute the Rayleigh quotient, $H = U^*AU$, and its eigendecomposition.
2. Select an eigenpair $(\mu, y)$ from $H$ and compute the Ritz pair $(\mu, Uy)$.
3. Compute the residual $r = AUy - \mu Uy$.
4. Orthogonalize $r$ against $U$ and normalize the result to be $u$.
5. $U = (U \;\; u)$.
```

that candidates approximate is called a *target*. In what follows symbols with the subscript k denote the variables in the kth iteration. We use $(\mu_k, z_k) = (\mu_k, U_k y_k)$ to represent the candidate pair in the kth iteration, and use (λ, x) for the target. The pair (μ_k, y_k) is called the *primitive Ritz pair.*

It can be shown that Algorithm 2.1 generates the same subspace as the classical Arnoldi process when every step is computed exactly. (A simple justification for this argument will be given in the next section.) Here, we are concerned with the convergence of the candidate when relatively large errors are introduced in the computation. Specifically, let r_k be the exact residual, and $\tilde{r}_k$ be the one used in subspace expansion. The introduced error is defined as

$$f_k = \tilde{r}_k - r_k.$$

Empirically, we have observed that f_k can be as large as $10^{-3}\|r_k\|$, and the candidate approximations can still converge to the target with rate similar to the rate without error.

In the analysis, we say f_i satisfies the *relative error condition* if

$$\|f_i\| \leq \epsilon \|r_k\|, \tag{2.1}$$

for $\epsilon \in (0, 1)$. This condition is always assumed in what follows.

2.1.2 The SIRA method

The second method, SIRA, is presented in Algorithm 2.2. The first three steps are identical to the RA method. In step 4, it applies shift-invert enhancement to the computed residual. Let $S = (A - \sigma I)^{-1}$. (Here we always assume that $(A - \sigma I)$ is invertible.) Then, the solution $v_k = Sr_k$, called the *enhanced vector*, is used in subspace expansion.

Algorithm 2.2 The SIRA iteration.

```
1. Compute the Rayleigh quotient, $H = U^*AU$, and its eigendecomposition.
2. Select an eigenpair $(\mu, y)$ from $H$ and compute the Ritz pair $(\mu, Uy)$.
3. Compute the residual $r = AUy - \mu Uy$.
4. Solve the linear system $(A - \sigma I)v = r$.
5. Orthogonalize $v$ against $U$ and normalize the result to be $u$.
6. $U = (U \;\; u)$.
```

Note that this algorithm is different from the RA method for matrix S, which is just Algorithm 2.1 with A replaced by S. The first difference is the SIRA computes the Ritz pair from the Rayleigh quotient of A, not of S. Second, the SIRA solves the linear system after the residual is computed, instead of solving it to compute the residual. However, it can be shown that when the same initial vector is used in exact arithmetic both methods generate the same subspace. A complete proof of this argument can be justified by induction. Here we only show that the bases generated by the SIRA satisfy the recursion

$$\mathrm{span}\{U_{k+1}\} = \mathrm{span}\{U_k\} \cup \mathrm{span}\{SU_k\}.$$

Starting with a simple identity

$$\begin{aligned} SA &= (A - \sigma I)^{-1}A \\ &= (A - \sigma I)^{-1}((A - \sigma I) + \sigma I) \\ &= I + \sigma S, \end{aligned} \tag{2.2}$$

we have

$$v_k = Sr_k = S(Az_k - \mu_k z_k) = z_k + (\sigma - \mu_k)Sz_k, \tag{2.3}$$

which is a linear combination of z_k and Sz_k. Since $z_k = U_k y_k$, $\operatorname{span}\{U_{k+1}\} = \operatorname{span}\{U_k\} \cup \operatorname{span}\{v_k\} = \operatorname{span}\{U_k\} \cup \operatorname{span}\{SU_k\}$.

The key property of SIRA is that the linear system can be solved to low accuracy, and the candidate approximations will still converge to the correct target, with the rate accelerated by shift-invert enhancement. Specifically, let $\tilde{v}_i$ be the computed approximate solution. The low accuracy requirement means the *relative residual* [29, 1.13.2],

$$\epsilon = \frac{\|(A - \sigma I)\tilde{v}_i - r_i\|}{\|r_i\|}, \tag{2.4}$$

can be relatively large, say 10^{-3}. If we let v_i be the exact solution, $v_i = Sr_i$, and f_i be the *error*, $f_i = \tilde{v}_i - v_i$, one can show that

$$\begin{aligned} \|f_i\| &= \|\tilde{v}_i - v_i\| \\ &= \|S(A - \sigma I)(\tilde{v}_i - v_i)\| \\ &\leq \|S\|\|(A - \sigma I)\tilde{v}_i - r_i\| \\ &= \|S\|\|r_i\|\frac{\|(A - \sigma I)\tilde{v}_i - r_i\|}{\|r_i\|} \\ &= \|S\|\|r_i\|\epsilon. \end{aligned} \tag{2.5}$$

If $\|S\|\epsilon < 1$, the error f_i satisfies the relative error condition (2.1). In the discussion of the SIRA, $\|S\|\epsilon < 1$ is always assumed. Also, we assume that f_i is produced entirely from solving the linear system. Other steps are computed exactly.

2.1.3 Residual and approximation

In this subsection, we are concerned with the relations of residual and eigenpair approximation. Let z be an approximation to an eigenvector x (z is not necessarily a Rayleigh–Ritz approximation). The eigenvalue approximation μ can be computed by

$$\mu = z^* A z. \tag{2.6}$$

The residual of the approximation (μ, z) is defined as

$$r = Az - \mu z.$$

It can be shown that the eigenvalue approximation μ, computed in (2.6), minimizes $\|r\|$ [41, pg.63].

In practical computations, $\|r\|$ is usually taken to reflect the error. Let $e = z - x$ be the *error* of the eigenvector approximation. The following inequality shows a large residual implies a poor eigenvector approximation.

$$\begin{array}{rcll} \|r\| & \leq & \|Az - \lambda z\| & (\text{optimality of } \mu) \\ & = & \|Ae - \lambda e\| & (Ax = \lambda x) \\ & \leq & 2\|e\| & (\|A\|, \lambda \leq 1) \\ & = & 2\|z - x\| & \end{array} \tag{2.7}$$

In practice, the concern is more about when a small residual implies a good approximation. Unfortunately, this may not be true. Let $(x\ X_\perp)$ be unitary and let

$$\begin{pmatrix} x^* \\ X_\perp^* \end{pmatrix} A \begin{pmatrix} x & X_\perp \end{pmatrix} = \begin{pmatrix} \lambda & h^* \\ 0 & L \end{pmatrix}.$$

Then in [20], it has been shown that

$$\frac{1}{\sqrt{2}}\|z - x\| \leq |\sin\angle(x, z)| \leq \frac{\|r\|}{\operatorname{sep}(\mu, L)}, \tag{2.8}$$

where

$$\operatorname{sep}(\mu, L) = \|(\mu I - L)^{-1}\|^{-1}. \tag{2.9}$$

The quantity $\operatorname{sep}(\mu, L)$ is bounded above by the distance between μ and the spectrum of L; however, it can be much smaller. In that case, a small residual does not imply a good eigenvector approximation.

In our analysis, the eigenpair approximation comes from the Rayleigh–Ritz approximation, in which the relevant value of sep is $\operatorname{sep}(\mu_k, L)$. To simplify the proofs, we only consider the cases that the target eigenvector and its approximations are well conditioned. A few assumptions are made for the convergence analyses.

Assumption 1. *The target eigenpair (λ, x) is simple (i.e., λ is of multiplicity one).*

Hence, $\operatorname{sep}(\lambda, L) > 0$.

Assumption 2. *There is a constant $C_1 > 0$ such that $\operatorname{sep}(\mu_k, L) \geq C_1$.*

The second assumption requires, in addition to Assumption 1, the Ritz value μ_k to be sufficiently near λ. Since the Ritz value μ_k satisfies (2.6), a converging Ritz vector z_k, and a large enough k suffices for this assumption to hold.

Now we consider the perturbed matrix $\tilde{A} = A+E$. From the fundamental perturbation theory, we have the following lemma [42].

Lemma 2.1. *There are positive constants C_2, C_3 and C_4 such that if*

$$\|E\| \leq C_2 \tag{2.10}$$

then there is a unique eigenpair $(\tilde{\lambda}, \tilde{x})$ of $\tilde{A} = A + E$ such that

$$\|\tilde{x} - x\| \leq C_3\|E\| \tag{2.11}$$

and

$$\operatorname{sep}(\lambda, \tilde{L}) \geq C_4 \tag{2.12}$$

The goal of our analysis is not to build a new convergence theory from scratch. Instead, we will build the analysis based on the convergence of the Ritz vectors of perturbed systems. To this end, we make the following assumption about the candidate vectors $\tilde{z}_k$ of the perturbed system.

Assumption 3. *There is a positive constant $C_5 \leq C_2$ such that if*

$$\|E\| \leq C_5$$

then there are constants $\tilde{\kappa}_1, \tilde{\kappa}_2, \ldots$, independent of E, with $\lim_k \tilde{\kappa}_k = 0$ such that

$$\|\tilde{z}_k - \tilde{x}\| \leq \tilde{\kappa}_k.$$

This assumption is called the *uniform convergence* of the $\tilde{z}_k$, which is reasonable for the targets well separated from the rest of spectrum.

2.2 Residual Arnoldi relations

In this section, the *residual Arnoldi relation* is derived to characterize certain numerical properties of the RA method. The derivation begins with the orthogonalization step. Let r_i be the exact residual, and $\tilde{r}_i = r_i + f_i$ be the computed one. The orthogonalization of $\tilde{r}_i$ against U_i can be written

$$\begin{aligned}(I - U_iU_i^*)\tilde{r}_i &= (I - U_iU_i^*)(r_i + f_i) \\ &= r_i - U_ig_i + f_i^{\perp}\end{aligned}$$

where $g_i = U_i^* r_i$, and $f_i^\perp = (I - U_i U_i^*) f_i$. Let $\rho_i = \|(I - U_i U_i^*)\tilde{r}_i\|$ and let $u_{i+1} = \rho_i^{-1}(I - U_i U_i^*)\tilde{r}_i$. Then, the above equation becomes

$$\rho_i u_{i+1} = r_i - U_i g_i + f_i^\perp. \tag{2.13}$$

By substituting $r_i = AU_i y_i - \mu_i U_i y_i$ in (2.13), we have

$$\rho_i u_{i+1} = (AU_i y_i - \mu_i U_i y_i) - U_i g_i + f_i^\perp.$$

Equivalently,

$$AU_i y_i = U_i(\mu_i y_i + g_i) + \rho_i u_{i+1} - f_i^\perp. \tag{2.14}$$

Now, let

$$\hat{g}_i = \begin{pmatrix} g_i \\ \rho_i \\ 0_{k-i-1} \end{pmatrix},$$

$$G_k = (\hat{g}_1 \ \cdots \ \hat{g}_{k-1} \ g_k), \tag{2.15}$$

and let Y_k be the upper triangular matrix consisting of y_i. We can combine the individual equations (2.14) to get

$$AU_k Y_k = U_k(Y_k M_k + G_k) + \rho_k u_{k+1} e_k^* + F_k^\perp, \tag{2.16}$$

where $F_k^\perp = (f_1^\perp \ \cdots \ f_k^\perp)$, and $M_k = \operatorname{diag}(\mu_1, \ldots, \mu_k)$.

Post-multiplying Y_k^{-1} in (2.16), we get the residual Arnoldi relation,

$$AU_k = U_k(Y_k M_k + G_k)Y_k^{-1} + \frac{\rho_k}{\eta_k} u_{k+1} e_k^* - F_k^\perp Y_k^{-1}, \tag{2.17}$$

where η_k is the last element of y_k.

When there is no error, it can be shown that (2.17) is an Arnoldi relation. According to the uniqueness of Arnoldi relations [41, Theorem 5.1], the only thing to verify is $(Y_k M_k + G_k)Y_k^{-1}$ is an upper Hessenberg matrix, which is easily seen because Y_k and Y_k^{-1} are upper triangular, M_k is diagonal, and G_k is upper Hessenberg.

In fact, by just moving $F_k^\perp Y_k^{-1}$ to the left hand side, one can obtain an Arnoldi relation,

$$(A + F_k^\perp Y_k^{-1} U_k^*)U_k = U_k(Y_k M_k + G_k)Y_k^{-1} + \frac{\rho_k}{\eta_k} u_{k+1} e_k^*, \tag{2.18}$$

The above equation gives us the backward error expression, in which U_k spans a Krylov subspace of a perturbed matrix $\tilde{A}_k = A + E_k$, where

$$E_k = F_k^{\perp} Y_k^{-1} U_k^*. \tag{2.19}$$

According to the theory of Krylov subspaces [41], U_k contains good approximations to the eigenvectors of $\tilde{A}_k$. The candidate approximation is common to both, as the following theorem shows.

Theorem 2.2. *Let $\tilde{H}_k$ be the Rayleigh quotient of $\tilde{A}$ and H_k be the Rayleigh quotient of A. The eigenpair (μ_k, y_k) of H_k is an eigenpair of $\tilde{H}_k$.*

Proof. From (2.18), the Rayleigh quotient of $\tilde{A}$ is $\tilde{H}_k = (Y_i M_i + G_i)Y_i^{-1}$. Similarly, the Rayleigh quotient of A can be derived from (2.17),

$$\begin{aligned} H_k &= U_k^* A U_k \\ &= (Y_k M_k + G_k)Y_k^{-1} - U_k^* F_k^{\perp} Y_k^{-1}. \end{aligned} \tag{2.20}$$

The matrix $U_k^* F_k^{\perp}$ is strictly lower triangular, since the ith column of $F_k^{\perp}$ is orthogonal to the first i columns of U_k. Consequently, the last column of $U_k^* F_k^{\perp}$ is zero. Therefore,

$$\begin{aligned} \mu_k y_k &= H_k y_k \\ &= ((Y_k M_k + G_k)Y_k^{-1} + U_k^* F_k^{\perp} Y_k^{-1})y_k \\ &= (Y_k M_k + G_k)Y_k^{-1} y_k + U_k^* F_k^{\perp} e_k \\ &= \tilde{H}_k y_k. \end{aligned}$$

□

It follows that,

$$z_k = U_k y_k = U_k \tilde{y}_k = \tilde{z}_k. \tag{2.21}$$

2.3 The error matrix

The error matrix E_k plays an important role in the convergence analysis. As stated in Lemma 2.1, if $\|E_k\|$ is smaller than a constant C_2, then the target vector of $\tilde{A}$ will be similar to A's. Empirically, $\|E_k\|$ is around the level of ϵ. Unfortunately, owing to our limited knowledge, no satisfactory proof could be made for the validation of this property. Here we make some heuristic justifications for a very special case. First, we assume that in the coordinate system of U_k, x is of the form $\sqrt{1-|\beta|^2}(1,\beta,\beta^2,\ldots)^*$, for some $|\beta| < 1$. Second, we assume that the primitive Ritz vector y_i is in the same direction as $U_i^* x$,

$$y_i = \gamma_i \begin{pmatrix} 1 \\ \beta \\ \vdots \\ \beta^{i-1} \end{pmatrix},$$

where $\gamma_i = \sqrt{(1-|\beta|^2)/(1-|\beta|^{2i})}$.

As a result, the matrix Y_k can be decomposed as

$$Y_k = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ & \beta & & \beta \\ & & \ddots & \vdots \\ & & & \beta^{k-1} \end{pmatrix} \begin{pmatrix} \gamma_1 & & & \\ & \gamma_2 & & \\ & & \ddots & \\ & & & \gamma_k \end{pmatrix},$$

and its inverse is

$$Y_k^{-1} = \begin{pmatrix} \gamma_1^{-1} & & & \\ & \gamma_2^{-1} & & \\ & & \ddots & \\ & & & \gamma_k^{-1} \end{pmatrix} \begin{pmatrix} 1 & -\beta^{-1} & & \\ & \beta^{-1} & -\beta^{-2} & \\ & & \ddots & \\ & & & \beta^{-(k-1)} \end{pmatrix}.$$

Therefore, the first column of $F_k^\perp Y_k^{-1}$ is $f_1^\perp$; and the ith column, $i > 1$, is

$$\frac{1}{\gamma_i \beta^{i-1}} f_i^\perp - \frac{1}{\gamma_{i-1}\beta^{i-1}} f_{i-1}^\perp,$$

whose norm can be bounded by

$$\frac{\sqrt{1-|\beta|^{2i}}\|f_i^\perp\| + \sqrt{1-|\beta|^{2(i-1)}}\|f_{i-1}^\perp\|}{\sqrt{1-|\beta|^2}|\beta|^{i-1}} \le \frac{\|f_i^\perp\| + \|f_{i-1}^\perp\|}{\sqrt{1-|\beta|^2}|\beta|^{i-1}}. \tag{2.22}$$

According to the relative error condition (2.1), $\|f_i\| \le \epsilon\|r_i\|$. In addition, by (2.7), $\|r_i\| \le 2\|x - z_i\|$. In the U-coordinate system,

$$\begin{aligned} \|x - z_i\|^2 &= (\gamma_i - \sqrt{1-|\beta|^2})^2(1 + \beta^2 + \cdots + \beta^{2(i-1)}) + \\ &\quad (1-|\beta|^2)(\beta^{2i} + \beta^{2(i+1)} + \cdots) \\ &= (1/\sqrt{1-|\beta|^{2i}} - 1)^2 + |\beta|^{2i}. \end{aligned} \tag{2.23}$$

Because $|\beta| < 1$,

$$\begin{aligned}\frac{1}{\sqrt{1-|\beta|^{2i}}} - 1 &= \frac{1-\sqrt{1-|\beta|^{2i}}}{\sqrt{1-|\beta|^{2i}}} \\ &\leq \frac{1-(1-|\beta|^{i})}{\sqrt{1-|\beta|^{2i}}} \\ &\leq \frac{|\beta|^{i}}{\sqrt{1-|\beta|^{2}}}.\end{aligned}$$

Equation (2.23) can be further bounded $\|x - z_i\| \leq \sqrt{2}|\beta|^i/\sqrt{1-|\beta|^2}$. Consequently,

$$\|f_i^{\perp}\| \leq \|f_i\| \leq \epsilon\|r_i\| \leq 2\sqrt{2}\frac{\epsilon|\beta|^i}{\sqrt{1-|\beta|^2}}. \tag{2.24}$$

With (3.16), (2.22) can be further bounded

$$\frac{\|f_i^{\perp}\| + \|f_{i-1}^{\perp}\|}{\sqrt{1-|\beta|^2}|\beta|^{i-1}} \leq \frac{2\|f_{i-1}^{\perp}\|}{\sqrt{1-|\beta|^2}|\beta|^{i-1}} \leq \frac{4\sqrt{2}\epsilon}{1-|\beta|^2}. \tag{2.25}$$

Therefore, $\|E_k\|$ can be bounded by $4\sqrt{2n}\epsilon/(1-|\beta|^2)$.

Although this justification is just made for a very special case, it captures two important features of E_k. First, when Y_k^{-1} is scaled so that its diagonal elements are one, its large elements tend to lie near the diagonal. Second, the last element η_k of the primitive Ritz vector y_k is roughly in the size of the norm of residual $\|r_k\|$, which makes $\|f_k\|/|\eta_k|$ in the order of ϵ.

In the following analysis, we assume that

Assumption 4. *There exist a constant C_6, such that $\|E_k\| \leq \epsilon C_6$ for all k.*

2.4 Convergence of the RA method

In this section, a convergence theorem is derived for the RA method, which shows the candidate residual approaches zero as the process goes. In the proof, we assume that $\epsilon \leq C_5/C_6$. Hence from Assumption 4 and the relative error condition (2.1), $\|E_k\| \leq C_5 \leq C_2$.

The proof starts with the relation between the target pair (λ, x) of A and the eigenpair $(\tilde{\lambda}_k, \tilde{x}_k)$ of $\tilde{A}_k$ that the Ritz pair $(\tilde{\mu}_k, \tilde{z}_k)$ approximates. From

(2.8),

$$\begin{aligned} \frac{1}{\sqrt{2}}\|x - \tilde{x}_k\| &\le \sin\angle(x, \tilde{x}_k) \\ &\le \frac{\|\tilde{A}_k x - \lambda x\|}{\mathrm{sep}(\lambda, \tilde{L}_k)} \\ &= \frac{\|(A + E_k)x - \lambda x\|}{\mathrm{sep}(\lambda, \tilde{L}_k)} \\ &= \frac{\|E_k x\|}{\mathrm{sep}(\lambda, \tilde{L}_k)} \end{aligned} \tag{2.26}$$

By Lemma 2.1, $\mathrm{sep}(\lambda, \tilde{L}_k)^{-1} < 1/C_4$, so

$$\|x - \tilde{x}_k\| \le \sqrt{2}\|E_k x\|/C_4.$$

The following lemma proves that $E_k x$ decreases along with the residual r_k.

Lemma 2.3. *Let (λ, x) be the target of A, (μ_k, z_k) be the candidate, and r_k be the candidate residual. If the relative error condition holds, then*

$$\|E_k x\| \le \epsilon\|r_k\|\left(1 + \frac{C_6}{C_1}\right). \tag{2.27}$$

Proof. Let $x = \alpha_k z_k + q_k$, where $\alpha_k = z_k^* x$ is the cosine of $\angle(z_k, x)$. Consequently, $\|q_k\|$ is $|\sin\angle(z_k, x)|$. From (2.8),

$$\|q_k\| \le \frac{\|r_k\|}{\mathrm{sep}(\mu_k, L)}.$$

Hence,

$$\begin{aligned} E_k x &= \alpha_k E_k z_k + E_k q_k \\ &= \alpha_k F_k^{\perp} Y_k^{-1} U_k^* z_k + E_k q_k. \end{aligned}$$

Since $z_k = U_k y_k$, the first term becomes $\alpha_k F_k^{\perp} e_k = \alpha_k f_k^{\perp}$. Therefore, $\|E_k x\|$ can be bounded by

$$\begin{aligned} \|E_k x\| &\le |\alpha_k|\|f_k^{\perp}\| + \|E_k\|\|q_k\| \\ &\le \epsilon\|r_k\| + \|E_k\|\frac{\|r_k\|}{\mathrm{sep}(\mu_k, L)} \\ &\le \epsilon\|r_k\| + \epsilon C_6 \frac{\|r_k\|}{\mathrm{sep}(\mu_k, L)} \\ &= \epsilon\|r_k\|\left(1 + \frac{C_6}{\mathrm{sep}(\mu_k, L)}\right). \end{aligned}$$

From Assumption 2 $\text{sep}(\mu_k, L) \geq C_1 > 0$, we have

$$\|E_k x\| \leq \epsilon \|r_k\| \left(1 + \frac{C_6}{C_1}\right).$$

□

Substituting (2.27) into (2.26), we get

$$\|x - \tilde{x}_k\| \leq \epsilon \|r_k\| \frac{\sqrt{2}}{C_4} \left(1 + \frac{C_6}{C_1}\right).$$

To simplify the notation, we set

$$C_7 = \sqrt{2}(1 + C_6/C_1)/C_4. \tag{2.28}$$

The above inequality can then be written as

$$\|x - \tilde{x}_k\| \leq \epsilon \|r_k\| C_7. \tag{2.29}$$

The second step of the proof is to build the relation of the Ritz vector z_k and perturbed eigenvector $\tilde{x}_k$. According to the uniform convergence assumption,

$$\|z_k - \tilde{x}_k\| = \|\tilde{z}_k - \tilde{x}_k\| \leq \tilde{\kappa}_k. \tag{2.30}$$

Finally, the convergence theory of the RA method is given in the following theorem.

Theorem 2.4. *If $\epsilon < 1/(2C_7)$,*

$$\|r_k\| \leq \frac{2\tilde{\kappa}_k}{1 - 2C_7\epsilon}, \tag{2.31}$$

where r_k is the residual computed by the residual Arnoldi method.

Proof. From (2.29) and (2.30), one has

$$\begin{aligned} \|z_k - x\| &\leq \|z_k - \tilde{x}_k\| + \|\tilde{x}_k - x\| \\ &\leq \tilde{\kappa}_k + C_7 \|r_k\| \epsilon. \end{aligned}$$

Also, (2.7) gives

$$\|r_k\| \leq 2\|z_k - x\| \leq 2(\tilde{\kappa}_k + C_7\|r_k\|\epsilon).$$

If $2C_7\epsilon < 1$, we can solve this inequality for $\|r_k\|$ to get

$$\|r_k\| \leq \frac{2\tilde{\kappa}_k}{1 - 2C_7\epsilon}.$$

$\square$

When the error is small, $\|r_k\|$ converges in the rate similar to the RA method without errors. However, as ϵ grows, $(1 - 2C_7\epsilon)$ approaches zero and the convergence slows. This prediction is born out by Figure 1.1.

2.5 Convergence of the SIRA method

In this section, we will derive a convergence theory for the SIRA method. First, we derive the *shift-invert residual Arnoldi* (SIRA) relation. The backward error relation is used to construct the convergence theory for the candidate approximations.

The derivation of the SIRA relation starts from the orthogonalization process. Let v_i be the exact enhanced vector, and $\tilde{v}_i$ be the computed one. Step 5 in Algorithm 2.2 can be expressed as

$$\begin{aligned} (I - U_iU_i^*)\tilde{v}_i &= (I - U_iU_i^*)(v_i + f_i) \\ &= v_i - U_ig_i + f_i^{\perp}, \end{aligned}$$

where $g_i = U^*v_i$ and $f_i^{\perp} = (I - U_iU_i^*)f_i$. If we let the orthogonalized vector be ρ_iu_{i+1}, the above equation can be expressed as

$$v_i = U_ig_i + \rho_iu_{i+1} - f_i^{\perp}.$$

By combining (2.3) and above equation, we have

$$\begin{aligned} v_i &= U_ig_i + \rho_iu_{i+1} - f_i^{\perp} \\ &= z_k + (\sigma - \mu_k)Sz_k \\ &= U_iy_i + (\sigma - \mu_i)SU_iy_i \end{aligned}$$

Equivalently,

$$(\sigma - \mu_i)SU_iy_i = U_i(g_i - y_i) + \rho_iu_{i+1} - f_i^{\perp}. \qquad (2.32)$$

Using the same definition of G_k and Y_k in (2.15), we can combine (2.32) to get,

$$SU_kY_k(\sigma I - M_k) = U_k(G_k - Y_k) + \rho_k u_{k+1}e_k^* - F_k^\perp, \tag{2.33}$$

where $M_k = \mathrm{diag}(\mu_i)$, and $F_k^\perp = \begin{pmatrix} f_1^\perp & f_1^\perp & \cdots & f_k^\perp \end{pmatrix}$. The SIRA relation can be obtained by post-multiplying $(\sigma I - M_k)^{-1}Y_k^{-1}$ on both sides,

$$\begin{aligned} & SU_k \qquad (2.34) \\ = \; & U_k(G_k - Y_k)(\sigma I - M_k)^{-1}Y_k^{-1} + \frac{\rho_k u_{k+1}e_k^*}{(\sigma - \mu_k)\eta_k} - F_k^\perp(\sigma I - M_k)^{-1}Y_k^{-1}, \end{aligned}$$

where η_k is the last element of y_k.

The backward error expression can be obtained by moving $F_k^\perp(\sigma I - M_k)^{-1}Y_k^{-1}$ to the left hand side,

$$\begin{aligned} & (S + F_k^\perp(\sigma I - M_k)^{-1}Y_k^{-1}U_k^*)U_k \qquad (2.35) \\ = \; & U_k(G_k - Y_k)(\sigma I - M_k)^{-1}Y_k^{-1} + \frac{\rho_k}{(\sigma - \mu_k)\eta_k}u_{k+1}e_k^*. \end{aligned}$$

From it, one can define the error matrix

$$E_k = F_k^\perp(\sigma I - M_k)^{-1}Y_k^{-1}U_k^*. \tag{2.36}$$

It can be shown that when $F_k^\perp = 0$, the subspace generated by SIRA method is identical to the Krylov subspace $\mathcal{K}_k(S, u_1)$, since $(G_k - Y_k)(\sigma I - M_k)^{-1}Y_k^{-1}$ is upper Hessenberg.

By letting $\hat{F}_k^\perp = F_k^\perp(\sigma I - M_k)^{-1}$, we see that E_k has a structure similar to the error matrix in (2.19). Let $\hat{f}_i^\perp$ be the ith column of $\hat{F}_k^\perp$. From (2.5),

$$\|\hat{f}_i^\perp\| \le \frac{\|f_i\|}{|\sigma - \mu_i|} \le \frac{\|S\|\|r_i\|\epsilon}{|\sigma - \mu_i|}. \tag{2.37}$$

The value of $\|S\|$ and $|\sigma - \mu_i|^{-1}$ is controlled by the selection of σ. If σ is extremely close to λ, then both bounds can be enormous. To avoid that, we make the following assumption.

Assumption 5. *There is a constant C_8 such that $\|S\| \le C_8$ and $|\sigma - \mu_k|^{-1} \le C_8$.*

With Assumption 5, $\|\hat{f}_i^\perp\|$ can be bounded by $C_8^2\|r_i\|\epsilon$.

For the error matrix E_k, we cannot give a formal proof to bound its norm. Empirically, we had found that $\|E_k\|$ is proportional to ϵ. A justification for the special case of E_k can be done as we did in Section 2.3. Here, we just simply make an assumption for the norm of E_k.

Assumption 6. *There is a constant C_9 such that $\|E_k\| \le \epsilon C_9$ for all k.*

Let's assume that $\|E_k\| \le \epsilon C_9 \le C_5$, where C_5 is the constant in Assumption 3. According to the uniform convergence assumption of the perturbed matrix, U_k should contain good approximations to the eigenvectors of $\tilde{S}_k$. However, in the computation, the Ritz approximation z_k is generated for A, not for S or $\tilde{S}_k$. Here, by assuming that $\tilde{S}_k$ is invertible, we define

$$\tilde{A}_k = \tilde{S}_k^{-1} + \sigma I. \tag{2.38}$$

Equivalently,

$$\tilde{S} = S + E_k = (\tilde{A}_k - \sigma I)^{-1}.$$

By cross-multiplying S^{-1} and $\tilde{S}_k^{-1}$, the above equation becomes

$$(A - \sigma I) = (\tilde{A}_k - \sigma I) + (\tilde{A}_k - \sigma I)E_k(A - \sigma I).$$

After some algebraic operations,

$$\tilde{A}_k = A - (\tilde{A}_k - \sigma I)E_k(A - \sigma I). \tag{2.39}$$

Let $\hat{E}_k = \tilde{A}_k - A = -(\tilde{A}_k - \sigma I)E_k(A - \sigma I)$. The norm of $\hat{E}_k$ is bounded by $\|\tilde{A}_k - \sigma I\|\|E_k\|\|A - \sigma I\|$. Individually, $\|E_k\|$ is assumed to be less than ϵC_9; the norm of $A - \sigma I$ can be bounded by $1 + |\sigma|$. To bound the first term, we use (2.39),

$$\begin{aligned} \|\tilde{A}_k - \sigma I\| &= \|A - \sigma I - (\tilde{A}_k - \sigma I)E_k(A - \sigma I)\| \\ &\le \|A - \sigma I\| + \|\tilde{A}_k - \sigma I\|\|E_k\|\|A - \sigma I\| \\ &\le \|A - \sigma I\| + \|\tilde{A}_k - \sigma I\|\epsilon C_8(1 + |\sigma|). \end{aligned}$$

If $\epsilon < 1/((1 + |\sigma|)C_9)$, then

$$\|\tilde{A}_k - \sigma I\| \le \frac{\|A - \sigma I\|}{1 - (1 + |\sigma|)C_9\epsilon}. \tag{2.40}$$

Consequently,

$$\|\hat{E}_k\| \le \frac{\|A - \sigma I\|^2 C_9\epsilon}{1 - (1 + |\sigma|)C_9\epsilon}. \tag{2.41}$$

With these properties, we can now consider the Ritz approximations of A and $\tilde{A}_k$. The following theorem shows the relation of the primitive Ritz vectors.

Theorem 2.5. *Let $H_k = U_k^* A U_k$ and $\tilde{H}_k = U_k^* \tilde{A}_k U_k$; and let y_k be an eigenvector of H_k, $\tilde{y}_k$ be an eigenvector of $\tilde{A}_k$. Then*

$$|\sin\angle(y_k, \tilde{y}_k)| \le \frac{\epsilon \|r_k\| \|\tilde{A}_k - \sigma I\|}{\operatorname{sep}(\mu_k, \tilde{L})}, \tag{2.42}$$

where sep *function is defined as in* (2.9).

Proof. From (2.39), $\tilde{H}_k$ can be derived

$$\begin{aligned} \tilde{H}_k &= U_k^* \tilde{A}_k U_k \\ &= U_k^* A U_k - U_k^* (\tilde{A}_k - \sigma I) E_k (A - \sigma I) U_k \\ &= H_k - U_k^* (\tilde{A}_k - \sigma I) E_k (A - \sigma I) U_k. \end{aligned}$$

Let (μ_k, y_k) be an eigenpair approximation to $\tilde{H}_k$, and p_k be its residual.

$$\begin{aligned} p_k &= \tilde{H}_k y_k - \mu_k y_k \\ &= H_k y_k - U_k^* (\tilde{A}_k - \sigma I) E_k (A - \sigma I) U_k y_k - \mu_k y_k \\ &= -U_k^* (\tilde{A}_k - \sigma I) E_k (A - \sigma I) U_k y_k \end{aligned}$$

Now, $E_k = F_k^\perp (\sigma I - M_k)^{-1} Y_k^{-1} U_k^*$. Therefore,

$$\begin{aligned} E_k (A - \sigma I) U_k y_k &= F_k^\perp (\sigma I - M_k)^{-1} Y_k^{-1} U_k^* (A - \sigma I) U_k y_k \\ &= F_k^\perp (\sigma I - M_k)^{-1} Y_k^{-1} (H_k - \sigma I) y_k \\ &= F_k^\perp (\sigma I - M_k)^{-1} Y_k^{-1} (\mu_k - \sigma I) y_k \\ &= -F_k^\perp e_k = -f_k^\perp \end{aligned}$$

From (2.8),

$$\begin{aligned} \sin\angle(y_k, \tilde{y}_k)| &\le \frac{\|p_k\|}{\operatorname{sep}(\mu_k, \tilde{L})} \\ &\le \frac{\|U_k^* (\tilde{A}_k - \sigma I) E_k (A - \sigma I) U_k y_k\|}{\operatorname{sep}(\mu_k, \tilde{L})} \\ &\le \frac{\|\tilde{A}_k - \sigma I\| \|f_k^\perp\|}{\operatorname{sep}(\mu_k, \tilde{L})} \\ &\le \frac{\epsilon \|\tilde{A}_k - \sigma I\| \|r_k\|}{\operatorname{sep}(\mu_k, \tilde{L})} \end{aligned}$$

□

If ϵ is small enough, it can be shown that

$$\|y_k - \tilde{y}_k\| \le \epsilon \|r_k\| C_{10}, \tag{2.43}$$

for a constant C_{10}.

A proof for the convergence of the SIRA method, similar to the one in section 2.4, proceeds as follows. Let x be the target eigenvector of A, and $\tilde{x}_k$ be the eigenvector of $\tilde{A}_k$ that z_k approximates. Note that x is also an eigenvector of S, and $\tilde{x}_k$ is also an eigenvector of $\tilde{S}_k$. First, we derive a bound on $\|x - \tilde{x}\|$. Let $\theta = 1/(\lambda - \sigma)$, the target eigenvalue of S. Equation (2.8) implies

$$\begin{aligned} \frac{1}{\sqrt{2}}\|x - \tilde{x}_k\| &\le \sin\angle(x, \tilde{x}_k) \\ &\le \frac{\|\tilde{S}_k x - \theta x\|}{\operatorname{sep}(\theta, \tilde{N}_k)} \\ &= \frac{\|(S + E_k)x - \theta x\|}{\operatorname{sep}(\theta, \tilde{N}_k)} \\ &= \frac{\|E_k x\|}{\operatorname{sep}(\theta, \tilde{N}_k)}, \end{aligned} \tag{2.44}$$

where $\tilde{N}_k$ is defined as follows,

$$\begin{pmatrix} \tilde{x}_k^* \\ \tilde{X}_k^* \end{pmatrix} \tilde{S}_k \begin{pmatrix} \tilde{x}_k & \tilde{X}_k \end{pmatrix} = \begin{pmatrix} \tilde{\theta}_k & \tilde{h}_k^* \\ 0 & \tilde{N}_k \end{pmatrix},$$

in which $(\tilde{x}_k, \tilde{X}_k)$ is unitary.

Here we assume that ϵ is small enough that $\|E_k\|$ is smaller than C_2, the condition for Assumption 3. Then, Lemma 2.1 guarantees

$$\|x - \tilde{x}_k\| \le \frac{\sqrt{2}\|E_k x\|}{C_4}. \tag{2.45}$$

The following lemma shows that $\|E_k x\|$ goes to zero along with $\|r_k\|$.

Lemma 2.6. *Let (λ, x) be the target, (μ_k, z_k) be the candidate, and r_k be the candidate residual. If the relative error condition holds, then*

$$\|E_k x\| \le \epsilon \|r_k\| \left(C_8^2 + \frac{C_9}{C_1} \right), \tag{2.46}$$

where C_1, C_8, C_9 are constants defined in Assumption 2, Assumption 5 and Assumption 6.

Proof. In this proof, we treat z_k and x as the eigenvector approximation and eigenvector of A (not S). Let $x = \alpha_k z_k + q_k$, where $\alpha_k = z_k^* x = \cos\angle(z_k, x)$. Therefore, $\|q_k\| = |\sin\angle(z_k, x)|$. From (2.8),

$$\|q_k\| \leq \frac{\|r_k\|}{\text{sep}(\mu_k, L)}.$$

Therefore, $E_k x$ can be expressed as

$$\begin{aligned} E_k x &= \alpha_k E_k z_k + E_k q_k \\ &= \alpha_k F_k^{\perp}(\sigma I - M_k)^{-1} Y_k^{-1} U_k^* z_k + E_k q_k \\ &= \frac{\alpha_k}{\sigma - \mu_k} f_k^{\perp} + E_k q_k. \end{aligned}$$

Its norm can be bounded as follows.

$$\begin{aligned} \|E_k x\| &\leq \frac{|\alpha_k|}{|\sigma - \mu_k|}\|f_k^{\perp}\| + \|E_k\|\|q_k\| \\ &\leq \frac{\epsilon\|S_k\|}{|\sigma - \mu_k|}\|r_k\| + \|E_k\|\frac{\|r_k\|}{\text{sep}(\mu_k, L)} \\ &\leq \epsilon C_8^2\|r_k\| + \epsilon C_9 \frac{\|r_k\|}{\text{sep}(\mu_k, L)} \\ &\leq \epsilon\|r_k\|\left(C_8^2 + \frac{C_9}{C_1}\right). \end{aligned}$$

□

Combining (2.45) and Lemma 2.6, we could give the following bound

$$\|x - \tilde{x}_k\| \leq \frac{\sqrt{2}\epsilon\|r_k\|}{C_4}\left(C_8^2 + \frac{C_9}{C_1}\right).$$

If we let $C_{11} = \sqrt{2}(C_8^2 + C_9/C_1)/C_4$, the above inequality can be written as

$$\|x - \tilde{x}_k\| \leq \epsilon\|r_k\|C_{11}. \tag{2.47}$$

The next step is to show the convergence of the Ritz vector z_k to the perturbed eigenpair $\tilde{x}_k$. From (2.43),

$$\|z_k - \tilde{z}_k\| = \|U_k y_k - U_k \tilde{y}_k\| \leq \epsilon\|r_k\|C_{10}.$$

By the uniform convergence assumption,

$$\|\tilde{z}_k - \tilde{x}_k\| \leq \tilde{\kappa}_k.$$

By the triangle inequality,

$$\|z_k - \tilde{x}_k\| \leq \|z_k - \tilde{z}_k\| + \|\tilde{z}_k - \tilde{x}_k\| \leq \epsilon\|r_k\|C_{10} + \tilde{\kappa}_k. \tag{2.48}$$

The following theorem completes the convergence proof.

Theorem 2.7. *If* $\epsilon < 1/2(C_{10} + C_{11})$,

$$\|r_k\| \leq \frac{2\tilde{\kappa}_k}{1 - 2(C_{10} + C_{11})\epsilon}. \tag{2.49}$$

Proof. From (2.47) and (2.48), one has

$$\begin{aligned} \|z_k - x\| &\leq \|z_k - \tilde{x}_k\| + \|\tilde{x}_k - x\| \\ &\leq \tilde{\kappa}_k + \epsilon\|r_k\|C_{10} + \epsilon\|r_k\|C_{11}. \end{aligned}$$

From (2.7),

$$\|r_k\| \leq 2\|z_k - x\| \leq 2(\tilde{\kappa}_k + \epsilon\|r_k\|(C_{10} + C_{11})).$$

If $2(C_{10} + C_{11})\epsilon < 1$, $\|r_k\|$ can be bounded by

$$\|r_k\| \leq \frac{2\tilde{\kappa}_k}{1 - 2\epsilon(C_{10} + C_{11})}.$$

□

Chapter 3. RAPACK

RAPACK is a numerical package which employs the residual Arnoldi method to solve large eigenproblems. The package is implemented in Fortran 95. Here are some of its features.

- Two computational methods, RA and SIRA, are implemented in RAPACK.
- RAPACK allows the computation to start with an arbitrary subspace that contains proper eigenvector approximations.
- The SIRA mode allows imprecise results of shift-invert enhancement.
- The storage requirement of RAPACK is moderate. Users can adjust the memory usage according to their computational resources.
- The package is independent of the format of the matrix. RAPACK permits any representation of matrix, but requires user to provide the necessary matrix operations.
- RAPACK provides several commonly used criteria for spectrum selection. Users may also define their own criteria.
- Users can inspect intermediate results during the computation and adaptively reconfigure certain parameters.
- In the computation of real nonsymmetric eigenproblems, RAPACK works in real arithmetic when dealing with complex eigenpairs.

In this chapter, we will treat the design and the implementation of RAPACK. Section 3.1 gives an overview of the algorithms used by RAPACK. Section 3.2 addresses some implementation details, including candidate selection, deflation, and subspace restarting. Section 3.3 discuss the process of handling complex eigenpairs of real matrices. Section 3.4 analyzes the time complexity and the storage requirement of the algorithm. Section 3.5

discusses the design decisions made in RAPACK. Section 3.6 shows how RAPACK can be used to implement other algorithms for eigenvalue problems, for example the inexact Krylov method.

3.1 Algorithms

RAPACK has two computational modes, RA mode and SIRA mode, corresponding to two residual Arnoldi algorithms. The RA mode, requiring only matrix vector multiplication, is suitable for computing the eigenvalues on the periphery of the spectrum. The SIRA mode applies shift-invert enhancement to the computed residual. As we have seen, the linear systems can be solved imprecisely. Table 3.1 compares some aspects of the two modes. Since the SIRA mode must work with shift-invert enhancement, we also present the RA mode with shift-invert enhancement to contrast the difference between the two. In the table, A denotes the original matrix; and S denotes the shift-inverted matrix $(A - \sigma I)^{-1}$. In addition, r_A represents the residual of A and r_S is the residual of S.

	RA mode for A	RA mode for S	SIRA mode
Spectrum of matrix	A	S	A
Subspace expanded by	r_A	r_S	Sr_A
Appr. converges to	eigenpair of A	eigenpair of S	eigenpair of A
Matrix multiplication	required	not required	required
Solving linear system	not required	required	required
Precision requirement for solving linear system	N/A	high	low

Table 3.1: Algorithmic difference of computational modes

3.1.1 Algorithm for the RA mode

The RA mode, as shown in Algorithm 3.1, consists of two stages. The first stage is designed to generate a list of candidates. If an initial subspace is available, then it will be used to obtain the candidates. Otherwise, candidates will be generated by the Arnoldi process. The second stage of the algorithm implements the residual Arnoldi method. First, it computes the Rayleigh quotient from the subspace and the eigendecomposition of the Rayleigh quotient. Then, the algorithm selects a Ritz pair as the candidate and computes its residual. If the residual is small enough, the candidate is considered to be a converged eigenpair approximation. After that, the algorithm either selects another candidate or stops, depending on how many converged approximations have been computed. If the residual is not small enough, then the algorithm continues the subspace expansion by residuals. Algorithm 3.1 describes the entire process, except for the details of candidate selection, deflation and subspace restarting, which will be discussed in the next section.

An iteration in RAPACK consists of a single subspace expansion. The first stage has a maximum number of iterations, the maximum dimension m of the subspace. The maximum number of iterations for the second stage is $t - m$, so that the total number of iterations does not exceed t. Later in this chapter, variables will be indexed with the iteration counter $iter$ when necessary.

In the implementation, the Rayleigh quotient $H = U^*AU$ is computed cumulatively; i.e. the Rayleigh quotient in the current iteration, denoted H_k, is calculated by updating the Rayleigh quotient from previous iteration H_{k-1}. Although the Rayleigh quotient is not required in the first stage, it is still computed for use in the second stage. If the process starts with a single vector, the generated subspace is a Krylov subspace. According to the Arnoldi relation, we have $AU_k = U_{k+1}\hat{H}_k$ where

$$\hat{H}_k = \begin{pmatrix} \hat{H}_{k-1} & h_k \\ 0 & \beta_k \end{pmatrix} \tag{3.1}$$

is a $(k + 1) \times k$ Hessenberg matrix and h_k and β_k are computed during the orthogonalization process: namely $v_k = U_k h_k + \beta_k u_{k+1}$. The Rayleigh quotient H_k is the upper square matrix of $\hat{H}$; i.e., $H_k = \hat{H}_k(1{:}k, 1{:}k)$.

If the subspace is not Krylov, H_k cannot be computed from (3.1). However, it can be still computed cumulatively.

$$H_k = U_k^* A U_k = \begin{pmatrix} U_{k-1}^* \\ u_k^* \end{pmatrix} A\,(U_{k-1}, u_k) = \begin{pmatrix} H_{k-1} & U_{k-1}^* A u_k \\ u_k^* A U_{k-1} & u_k^* A u_k \end{pmatrix}. \tag{3.2}$$

To avoid the computation of AU_{k-1} in every iteration, RAPACK accumulates Au_k in a variable V_k. This matrix V_k is also used in the computation of the residual. Hence, only one matrix vector multiplication is required per iteration.

Algorithm 3.1 The basic algorithm for the RA mode in RAPACK.

Input: Matrix A; Initial subspace U_1; The maximum dimension of the subspace m; The maximum number of iterations t; Required precision for computed eigenpair τ; The number of desired eigenpairs s.

Output: a list of eigenpair approximations L.

```
     (The initialization stage)
1.   Let $U = U_1$, $u = U(:,1)$, $V = []$.
2.   for $k = 1$ to $m$.
3.       Compute $v = Au$ and let $V = [V, v]$.
4.       if dim($U$) > dim($V$)
5.           Let $u = U(:,k)$
6.       else
7.           $u$ =Orthogonalize($U, v$); $U = [U, u]$.
8.       end if.
9.   end for.
     (The residual Arnoldi stage)
10.  for $k = m + 1$ to $t$.
11.      Compute Rayleigh quotient $H$ and its eigendecomposition.
12.      do
13.          Select an eigenpair $(\mu, y)$ of $H$.
14.          Compute the residual $r = Vy - \mu Uy$ for the Ritz pair $(\mu, Uy)$.
15.          if ($\|r\| \leq \tau$)
16.              if (Number of converged approximations $\geq s$)
17.                  return $L$.
18.              else
19.                  Add $(\mu, Uy)$ to $L$.
20.              end if
21.          end if
22.      while ($\|r\| \leq \tau$)
23.      Deflate all new converged $\mu$ from the spectrum of $H$.
24.      $u$ =Orthogonalize($U, r$).
25.      if (dim($U$) $\geq m$)
26.          Restart the subspace.
27.      end if
28.      Compute $v = Au$ and let $U = [U, u]$ and $V = [V, v]$.
29.  end for
```

3.1.2 Algorithms for the SIRA mode

The SIRA mode is similar to Algorithm 3.1, except one additional step is required before step 24: namely to solve the linear system $(A-\sigma I)v = r$ and use v in the orthogonalization of the step 24.

3.2 Candidate selection, deflation and subspace restarting

It is well known that the Rayleigh Ritz method can produce spurious eigenvalue approximations when the subspace contains converged eigenvectors, and this is the reason for deflation [2]. In RAPACK, the subspace U is divided into two subspaces U_1 and U_2, $U = (U_1\ U_2)$, where U_1 spans the subspace that contains all converged eigenvectors and U_2 spans the rest space of U. This complicates the algorithms for candidate selection, deflation, and restarting. We will describe each of them in this section.

3.2.1 Candidate selection

According to the partition of the subspaces, the Rayleigh quotient H can be decomposed into blocks.

$$H = U^*AU = \begin{pmatrix} U_1^* \\ U_2^* \end{pmatrix} A(U_1\ U_2) = \begin{pmatrix} U_1^*AU_1 & U_1^*AU_2 \\ U_2^*AU_1 & U_2^*AU_2 \end{pmatrix} = \begin{pmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{pmatrix}.$$

Since U_1 spans the subspace that contains converged eigenvectors, U_1 can be treated as an invariant subspace. Therefore, for some matrix M_1, $AU_1 = U_1M_1$, up to the convergence precision τ, and hence $H_{21} = U_2^*AU_1 = U_2^*U_1M_1 = 0$. The Rayleigh quotient then becomes

$$H = \begin{pmatrix} H_{11} & H_{12} \\ 0 & H_{22} \end{pmatrix}, \tag{3.3}$$

or

$$H = \begin{pmatrix} H_{11} & 0 \\ 0 & H_{22} \end{pmatrix} \tag{3.4}$$

if A is Hermitian.

The above decomposition separates the spectrum of H into two disjoint parts: the spectrum of H_{11} and the spectrum of H_{22}. It is obvious that candidates should be selected only from the spectrum of H_{22}. Let (μ, y_2) be an eigenpair of H_{22}, in which μ is the candidate Ritz value. We assume μ is a simple eigenvalue of H_{22}. If the matrix A is Hermitian, the Ritz vector is $U_2 y_2$ because it is orthogonal to U_1. If A is not Hermitian, the Ritz vector needs to be computed from the eigenvector of the entire matrix H. Let

$$y = \begin{pmatrix} y_1 \\ \hat{y}_2 \end{pmatrix},$$

be the eigenvector of H corresponding to the eigenvalue μ, $Hy = \mu y$.

$$Hy = \begin{pmatrix} H_{11} & H_{12} \\ 0 & H_{22} \end{pmatrix} \begin{pmatrix} y_1 \\ \hat{y}_2 \end{pmatrix} = \mu \begin{pmatrix} y_1 \\ \hat{y}_2 \end{pmatrix}.$$

The above equation is equivalent to the following equations.

$$H_{11} y_1 + H_{12} \hat{y}_2 = \mu y_1 \tag{3.5}$$
$$H_{22} \hat{y}_2 = \mu \hat{y}_2 \tag{3.6}$$

The second equation implies $\hat{y}_2 = y_2$, and the first equation gives a formula to compute y_1.

$$y_1 = -(H_{11} - \mu I)^{-1} H_{12} y_2 = (\mu I - H_{11})^{-1} H_{12} y_2. \tag{3.7}$$

Thus, the eigenvector y can be constructed by concatenating y_1 and y_2 and the Ritz vector can be computed directly in the form $U_1 y_1 + U_2 y_2$.

The only problem with this method is that μ may also be an eigenvalue of H_{11}. In that case, matrix $(\mu I - H_{11})$ is not invertible. In the implementation of RAPACK, this problem is avoidable by adding some perturbation, less than τ, to the matrix $(\mu I - H_{11})$. Algorithm 3.2 sketches the process of candidate computation.

3.2.2 Deflation

The major task of deflation is to expand the subspace U_1 by the converged eigenvectors and to shrink the subspace U_2 by removing these vectors. The deflation algorithm in RAPACK can deflate all converged eigenpairs in an iteration.

The algorithm is described as follows. Suppose

$$(\mu^{(1)}, Uy^{(1)}), (\mu^{(2)}, Uy^{(2)}), \cdots, (\mu^{(j)}, Uy^{(j)})$$

Algorithm 3.2 The algorithm for candidate computation.

```
Input:   Rayleigh quotient $H$.
         Tolerance $\tau$.
Output:  An eigenpair $(\mu, y)$ of $H$.

1.  Let $H = \begin{pmatrix} H_{11} & H_{12} \\ 0 & H_{22} \end{pmatrix}$.
    Compute the eigendecomposition of $H_{22} = Y_2 M_2 Y_2^{-1}$.
2.  Select an eigenpair of $H_{22}$, $(\mu, y_2)$.
3.  Compute the pivoted LU decomposition of $-(H_{11} - \mu I) = PLU$.
4.  If any diagonal element of $L$ is of magnitude less than $\tau$,
    replace it with $\tau$.
5.  Solve the system $PLUy_1 = H_{12}y_2$.
6.  Let $y = \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}$.
```

are the converged approximations to be deflated. First, the Schur decomposition of H_{22} is computed,

$$H_{22} = WTW^*,$$

where W is unitary and T is upper triangular. Next, the eigenvalues of T are reordered so that the first j diagonal entries of T are $\mu^{(1)}, \mu^{(2)}, \cdots, \mu^{(j)}$. The reordering can be achieved by a unitary matrix Q,

$$H_{22} = WQQ^*TQQ^*W^* = \hat{W}\hat{T}\hat{W}^*,$$

where $\hat{W} = WQ$ and $\hat{T} = Q^*TQ$. Finally, the Rayleigh quotient and the subspaces are updated by $\hat{W}$ and $\hat{T}$. The Rayleigh quotient after deflation is

$$H = \begin{pmatrix} H_{11} & H_{12}\hat{W} \\ 0 & \hat{T} \end{pmatrix}.$$

The subspaces U and V are updated by post-multiplying a matrix B,

$$B = \begin{pmatrix} I & 0 \\ 0 & \hat{W} \end{pmatrix}.$$

Algorithm 3.3 The deflation algorithm in RAPACK.

```
Input:    Eigenvalue approximations $\mu^{(1)}, \mu^{(2)}, \cdots, \mu^{(j)}$;
          Rayleigh quotient $H$ and subspace $U$ and $V$.
Output:   Updated Rayleigh quotient $\hat{H}$ and subspaces $\hat{U}$ and $\hat{V}$.

1.  $H = \begin{pmatrix} H_{11} & H_{12} \\ 0 & H_{22} \end{pmatrix}$.
    Compute the Schur decomposition of $H_{22} = WTW^*$.
2.  Reorder the eigenvalues of $T$.
      $\hat{W} = WQ$ and $\hat{T} = Q^*TQ$ such that
    $\mu^{(1)}, \mu^{(2)}, \cdots, \mu^{(j)}$ are the first $j$ diagonal elements of $\hat{T}$.
3.  Let $\hat{H} = \begin{pmatrix} H_{11} & H_{12}\hat{W} \\ 0 & \hat{T} \end{pmatrix}$,
    and $\hat{V} = VB$ and $\hat{U} = UB$, where $B = \begin{pmatrix} I & 0 \\ 0 & \hat{W} \end{pmatrix}$.
```

Let $\hat{W} = \begin{pmatrix} W_1 & W_2 \end{pmatrix}$, where W_1 consists of the first j columns of $\hat{W}$ and W_2 has the remaining columns of $\hat{W}$. The subspace U_1 after deflation becomes $\begin{pmatrix} U_1 & U_2W_1 \end{pmatrix}$ and the updated subspace U_2 is U_2W_2.

To verify the correctness of the algorithm, we need to prove two things. First, every eigenvector approximation $Uy^{(i)}$ is contained in the subspace $\begin{pmatrix} U_1 & U_2W_1 \end{pmatrix}$ after deflation. Second, the updates on Rayleigh quotient and subspaces are correct. The proof of the first argument starts with the decomposition of a converged eigenvector $Uy^{(i)} = U_1y_1^{(i)} + U_2y_2^{(i)}$. Since $y_2^{(i)}$ is an eigenvector of H_{22} corresponding to $\mu^{(i)}$, $y_2^{(i)}$ will be contained in W_1. Therefore, the vector $U_1y_1^{(i)} + U_2y_2^{(i)}$ is in the subspace $\begin{pmatrix} U_1 & U_2W_1 \end{pmatrix}$.

The validity of the updated Rayleigh quotient and subspaces is shown as follows. Since $\hat{W}$ is unitary, matrix B is also unitary. The updated matrix $\hat{U} = UB$ therefore remains orthogonal. The updated matrix $\hat{V}$ is

$VB = AUB = A\hat{U}$. The updated Rayleigh quotient equals

$$\begin{aligned} \hat{U}^* A \hat{U} &= B^* U^* A U B = B^* H B \\ &= \begin{pmatrix} I & 0 \\ 0 & \hat{W}^* \end{pmatrix} \begin{pmatrix} H_{11} & H_{12} \\ 0 & H_{22} \end{pmatrix} \begin{pmatrix} I & 0 \\ 0 & \hat{W} \end{pmatrix} \\ &= \begin{pmatrix} H_{11} & H_{12}\hat{W} \\ 0 & \hat{W}^* H_{22} \hat{W} \end{pmatrix} \\ &= \begin{pmatrix} H_{11} & H_{12}\hat{W} \\ 0 & \hat{T} \end{pmatrix}. \end{aligned}$$

3.2.3 Subspace restarting

RAPACK uses the Krylov Schur method [40] in subspace restarting. The Krylov Schur method, as introduced in Chapter 1, consists of three steps.

1. Select the Ritz pairs that are to be retained.
2. Compute the Schur decomposition of the Rayleigh quotient and reorder the Schur vectors such that the desired Ritz pairs are in the front of the Schur decomposition.
3. Update the subspace and Rayleigh quotient.

The algorithm is similar to the deflation algorithm. The difference is that in the deflation algorithm the selected Ritz pairs are converged approximations and the dimension of the subspace remains the same. In subspace restarting, the selected eigenpairs are the eigenpairs to be kept in the restarted subspace, and the unselected eigenpairs will be discarded. The process of subspace restarting is given in Algorithm 3.4.

3.3 Complex eigenpairs in real nonsymmetric matrices

For real nonsymmetric matrices, complex conjugate eigenvalues pose problems for the residual Arnoldi method. If a candidate is complex, so is its residual, and eventually every operation will be in complex arithmetic, since the residual Arnoldi method uses residuals in subspace expansion. One way to solve this problem is to use the complex arithmetic in the beginning. However, if all the targets are real, that is inefficient. In this section, we describe how RAPACK circumvents complex arithmetic for complex targets.

Algorithm 3.4 The algorithm of subspace restarting.

```
Input:   Dimension of restarted subspace $j$;
         Rayleigh quotient $H$ and subspace $U$ and $V$.
Output:  Updated Rayleigh quotient $\hat{H}$ and subspaces $\hat{U}$ and $\hat{V}$.

1.   $H = \begin{pmatrix} H_{11} & H_{12} \\ 0 & H_{22} \end{pmatrix}$.
     Compute the Schur decomposition of $H_{22} = WTW^*$.
2.   Let $\hat{\jmath} = j - \dim(H_{11})$.
     Select desired eigenvalues $\mu^{(1)}, \mu^{(2)}, \cdots, \mu^{(\hat{\jmath})}$ from $H_{22}$.
3.   Reorder the eigenvalues of $T$.
       $\hat{W} = WQ$ and $\hat{T} = Q^*TQ$ such that
       $\mu^{(1)}, \mu^{(2)}, \cdots, \mu^{(\hat{\jmath})}$ are the first $\hat{\jmath}$ diagonal elements of $\hat{T}$.
4.   Let $\hat{W}_1 = \hat{W}(:, 1{:}\hat{\jmath})$ and $\hat{T}_{11} = \hat{T}(1{:}\hat{\jmath}, 1{:}\hat{\jmath})$.
5.   Let $\hat{H} = \begin{pmatrix} H_{11} & H_{12}\hat{W}_1 \\ 0 & \hat{T}_{11} \end{pmatrix}$,
     and $\hat{V} = VB$ and $\hat{U} = UB$, where $B = \begin{pmatrix} I & 0 \\ 0 & \hat{W}_1 \end{pmatrix}$.
```

3.3.1 Subspace expansion

In the Arnoldi process, the subspace generated for a real matrix is real, because the vectors used in subspace expansion are always real. The only problem for computing a complex target is when a complex shift is used in the shift-invert enhancement. In that case, the shift-inverted matrix is complex, and so is the generated subspace.

Several methods have been proposed to avoid this problem. One is the double shift strategy. Let $\sigma = \alpha + i\beta$ be a complex shift. The double shift strategy solves the following equation

$$(A - \sigma I)(A - \bar{\sigma} I)v = u_k, \tag{3.8}$$

and uses v in subspace expansion. Since the matrix

$$(A - \sigma I)(A - \bar{\sigma} I) = A^2 - 2\alpha A + \sigma\bar{\sigma} I$$

is real, the generated subspace remains real.

In [28], Parlett and Saad proposed a method, which solves the complex linear system $(A-\sigma I)v = u_k$, and uses the real part (or the imaginary part) of the solution in subspace expansion. It can be shown that the real part of v is equivalent to

$$v_{\text{real}} = (A-\alpha I)(A-\bar{\sigma}I)^{-1}(A-\sigma I)^{-1}u_k,$$

and the imaginary part equals

$$v_{\text{imag}} = \beta(A-\bar{\sigma}I)^{-1}(A-\sigma I)^{-1}u_k.$$

This can be verified easily by computing

$$\begin{aligned} v &= v_{\text{real}} + iv_{\text{imag}} \\ &= [(A-\alpha I) + i\beta I](A-\bar{\sigma}I)^{-1}(A-\sigma I)^{-1}u_k \\ &= [A-(\alpha - i\beta)I](A-\bar{\sigma}I)^{-1}(A-\sigma I)^{-1}u_k \\ &= (A-\bar{\sigma})(A-\bar{\sigma}I)^{-1}(A-\sigma I)^{-1}u_k \\ &= (A-\sigma I)^{-1}u_k \end{aligned}$$

If the real part is used, the generated subspace is a Krylov subspace for the matrix $(A-\alpha I)(A-\bar{\sigma}I)^{-1}(A-\sigma I)^{-1}$; if the imaginary part is used, the subspace is Krylov for the matrix $\beta(A-\bar{\sigma}I)^{-1}(A-\sigma I)^{-1}$. Thus, both matrices are real and contain some information about the double shifted matrix.

The residual Arnoldi method has more problems with complex targets than the ordinary Arnoldi method. First, a complex candidate generates a complex residual. This complex residual has parallel real and imaginary parts if the generated subspace is a Krylov subspace of a real matrix. However, this property is not valid in general. Second, when a complex shift is used in the shift-invert enhancement, Parlett and Saad's algorithm cannot be applied directly, because neither part of the solution alone is a useful vector.

In RAPACK, the generated subspace cannot be assumed to be a Krylov subspace, because of errors and the fact that the initial subspace can be arbitrary. A double vector method that generates a real subspace for possibly complex candidates is given in Algorithm 3.5. It uses both parts of a complex vector in subspace expansion if necessary. Although up to twice the storage for the subspace may be required, this method guarantees the convergence of complex approximations.

Let v_{real} and v_{imag} be vectors for the real part and the imaginary part of the complex vector that is used for subspace expansion. The decision of whether to use both vectors is made by checking their linear dependence. Instead of measuring their angle, the normalized difference of vectors v_{real} and v_{imag},

$$\delta = \min\left\{ \left| \frac{v_{\text{real}}}{\|v_{\text{real}}\|} - \frac{v_{\text{imag}}}{\|v_{\text{imag}}\|} \right|, \left| \frac{v_{\text{real}}}{\|v_{\text{real}}\|} + \frac{v_{\text{imag}}}{\|v_{\text{imag}}\|} \right| \right\}, \tag{3.9}$$

is evaluated. When δ is small enough, only one vector, whichever has the larger magnitude, is used in subspace expansion. Otherwise, both vectors are added into subspace one by one. The order is decided by their norm, the larger one first.

Algorithm 3.5 The double vector method in the subspace expansion.

```
Input:   $z = z_{\text{real}} + iz_{\text{imag}}$ the complex vector to be used;
         Required precision $\epsilon$ for solving linear system;
         Subspace $U$ and $V$.
Output:  Updated subspaces $U$ and $V$.

 1.  Let $z_{\max} = \arg\max(\|z_{\text{real}}\|, \|z_{\text{imag}}\|)$
     and $z_{\min} = \arg\min(\|z_{\text{real}}\|, \|z_{\text{imag}}\|)$.
 2.  Let $u_1$ be the normalized result of orthogonalizing
     $z_{\max}$ against $U$.
 3.  $U = [U, u_1]$.
 4.  $\delta = \min\left\{ \left| \frac{z_{\max}}{\|z_{\max}\|} - \frac{z_{\min}}{\|z_{\min}\|} \right|, \left| \frac{z_{\max}}{\|z_{\max}\|} + \frac{z_{\min}}{\|z_{\min}\|} \right| \right\}$.
 5.  if ( $\delta < \epsilon$) then
         /* $z_{\text{real}}$ and $z_{\text{imag}}$ are nearly parallel.*/
 6.      Compute $v_1 = Au_1$ and let $V = [V, v_1]$.
 7.  else
 8.      Let $u_2$ be the normalized result of orthogonalizing
         $z_{\min}$ against $U$.
 9.      $U = [U, u_2]$.
10.      Compute $(v_1, v_2) = A(u_1, u_2)$ and let $V = [V, v_1, v_2]$.
11.  end if
```

Note that the algorithms maintain the subspace $V = AU$ in step 6 and 10. If both vectors are used, two matrix vector multiplications are invoked to match the dimension of subspace U. The matrix V is required to be real, which means matrix A must be real. Thus, a complex shift-inverted matrix $S = (A - \sigma I)^{-1}$ cannot be plugged into the RA mode directly. When a complex shift is considered in the RA mode, user should use either the double shift strategy, or Parlett and Saad's method. In section 3.6.3, we will demonstrate how it is implemented.

This limitation does not apply to the SIRA mode because the Rayleigh quotient is for A not for S. And the shift value does not need to be a constant in the SIRA mode. If user wants to solve a real nonsymmetric eigenproblem with a complex shift, the SIRA mode is a better choice.

3.3.2 Computation of Ritz vectors and residuals

Complex eigenpairs also cause some problems in the computation of Ritz vectors and residuals. In order to process them, RAPACK must provide storage and procedures to handle these complex vectors. One way is to use complex data type for all cases, no matter whether the Ritz pair is real or complex. However, when it is real, unnecessary cost will be paid. Remember that complex arithmetic requires twice storage and four times computation compared to real arithmetic. Another solution is to create two sets of variables: one is real and another is complex. During the computation, decision on which one should be used is dynamically made. One obvious drawback for this method is the waste of storage if only one kind of data type is needed in computation. The third way to treat this problem, which is used in RAPACK, is to put everything in real arithmetic. When the Ritz pair is complex, RAPACK just replaces it with an identical real matrix representation.

In Algorithm 3.2, the candidate computation starts by computing the eigenpairs of H_{22}, which is a submatrix of the Rayleigh quotient that contains non-converged Ritz values. Suppose $(\mu, y_2) = (\alpha + i\beta, w_2 + iz_2)$ is an eigenpair of H_{22} that is picked by RAPACK. To compute the Ritz vector, one must solve the linear equation

$$-(H_{11} - \mu I)y_1 = H_{12}y_2 \tag{3.10}$$

and use $(y_1^T \; y_2^T)^T$ as primitive Ritz vector. It is a complex linear system since μ and y_2 are complex. A real linear system equivalent to (3.10) is described as follows.

$$\begin{pmatrix} \alpha I - H_{11} & -\beta I \\ \beta I & \alpha I - H_{11} \end{pmatrix} \begin{pmatrix} w_1 \\ z_1 \end{pmatrix} = \begin{pmatrix} H_{22}w_2 \\ H_{22}z_2 \end{pmatrix}, \tag{3.11}$$

where $y_1 = w_1 + iz_1$.

The residual computation can be converted to real arithmetic in the same way. Let $(\mu, Uy) = (\alpha + i\beta, Uw + iUz)$ be the complex candidate. Its residual is

$$\begin{aligned} r &= AUy - Uy\mu \\ &= A(Uw + iUz) - (Uw + iUz)(\alpha + i\beta) \\ &= \left[AU(w, z) - U(w, z) \begin{pmatrix} \alpha & \beta \\ -\beta & \alpha \end{pmatrix} \right] \begin{pmatrix} 1 \\ i \end{pmatrix}. \end{aligned}$$

Without multiplication by the vector $(1\ i)^T$, the residual r can be represented in a two column matrix, one column for the real part and another for its imaginary part.

3.3.3 Deflation and subspace restarting

Since the complex eigenpairs in a real matrix are always conjugate, when one complex eigenpair approximation is converged, its conjugate one must be converged too. In this case, RAPACK will deflate both approximations from the subspace, even if the conjugate one is not a desired eigenpair. The same scenario happens in subspace restarting. If a complex eigenvector is selected into restarting subspace, its conjugate pair will be in the restarting subspace too.

The algorithm to achieve this goal is quite simple, because RAPACK use real Schur decomposition to compute the eigenvalues of H_{22}. The reordering algorithm for the real Schur decomposition will keep conjugated eigenpairs together when reordering them.

3.4 Time complexity and memory requirement

In this section, we analyze the time complexity and the memory requirements for RAPACK. As usual, we start with the algorithm for the normal mode, and then discuss one for the SIRA mode.

3.4.1 Time complexity

Since most calculations in RAPACK consist of paired additions and multiplications, we will call such a pair a flam. Other operations, such as searching and sorting, are denoted by a flcm, a floating point comparison. The time complexity analysis for memory allocation and data transfer will not be covered in this section.

Here is the notation used in the analysis. The scalar n is the order of the matrix A; m is the maximum dimension of the subspace; s is the number of desired eigenpairs, which is also the dimension of the restarted subspace. The function $f(n)$ denotes the time complexity for matrix vector multiplication. Similarly, $g(n, \epsilon)$, denotes the average time complexity for solving a linear system to the accuracy ϵ.

In the analysis, we employ the O-notation [44] to simplify the results, such that only the dominant cost is addressed. The O-notation, the asymptotic upper bound, for a function $g(n)$ is defined as

$$\begin{aligned} O(g(n)) &= f(n) \text{ if there exist positive constants } c \text{ and } n_0 \\ & \quad \text{such that } 0 \leq f(n) \leq cg(n) \text{ for all } n \geq n_0. \end{aligned} \tag{3.12}$$

We also assume that $n \gg m > s$, since in practice, we are only concerned with the large matrices.

An *iteration* is defined to be a single expansion at the subspace. The scalar k denotes the total number of iterations that performed. An *epoch* is defined as the period between two consecutive restarting steps. The integer p will be the total number of epochs, which roughly equals k/m.

In the first stage of Algorithm 3.1, the major cost is the orthogonalization, $O(nm^3)$ flam, and matrix vector multiplication, $O(mf(n))$ flam. In the second stage, the computation is divided into several operations: Rayleigh quotient computation, eigenvalue computation, target selection, residual computation, deflation, subspace restarting, and orthogonalization. Here, we first consider the time complexity of one epoch. Then we compute the time complexity for the entire process.

The computation of the Rayleigh quotient, in each iteration, takes $2nm+n$ flam for U^*v, u^*V and u^*v. This adds up to $2nm^2 + nm$ flam in one epoch. The eigenvalue decomposition of the Rayleigh quotient requires $O(l^3)$ flam for the Rayleigh quotient of order l. In one epoch, the time complexity will be $O(m^4)$. To compute Ritz vector and residual, an additional $2nm$ will be needed. The target selection sorts the Ritz values in each iteration, which takes $O(m^3)$ flcm in one epoch. The residual computation requires nm^2+nm flam in one epoch. And the orthogonalization, same with the workload in the first stage, takes $nm^2 + mf(n)$ flam in one epoch.

Subspace restarting is performed once per epoch. The first step, selecting desired Ritz pairs, does not cost anything because the Ritz pairs are already ranked by the process of the target selection. The second step, computing Schur decomposition and reordering the Schur pairs, takes $O(m^3)$ flam. The third step, updating subspaces and Rayleigh quotient, takes $nsm + sm$ flam, because the dimension of the restarted subspace is s.

The deflation is executed at most s times during the entire process, because more than one converged eigenpairs may be deflated together. Like the subspace restarting, the major cost of deflation is Schur decomposition, reordering and subspace updating, which takes $O(m^3) + nm^2 + m^2$ flam. In the case of deflating conjugate eigenpairs, it requires additional $O(m)$ flcm for searching the conjugate pair. The overall cost is bounded by $sO(m^3) + snm^2 + sm^2$ flam and $sO(m)$ flcm.

Since there are p epoches, the overall cost, including the operations in the first stage, is $sO(m^3)+snm^2+sm^2+p[O(m^4)+2nm^3+O(nsm)+mf(n)]$ flam and $sO(m) + pO(m^3)$ flcm. With the estimation of $p = k/m$, the complexity for Algorithm 3.1 becomes

$$\begin{array}{l}(s+k)O(m^3) + (s+k)O(nm^2) + kO(ns) + kf(n) \text{ flam} \\ \text{ and } kO(m^2) \text{ flcm}\end{array}$$

Moreover, because RAPACK requires $s < m \leq k$ and $m \leq n$, the above complexity can be further simplified

$$\begin{array}{ll} kO(nm^2) + kf(n) & \text{flam} \\ kO(m^2) & \text{flcm.} \end{array} \tag{3.13}$$

The detail time complexity of each task is listed in Table 3.2.

From (3.13) and Table 3.2, we see that the most time consuming tasks are matrix-vector multiplication, $kf(n)$, and orthogonalization, $kO(nm^2)$. A naive implementation of matrix-vector multiplication will cost n^2, which could make the matrix-vector multiplication be the most time consuming task of all. However, in many cases, such as a sparse matrix [27, 9] or a structure matrix [43], the matrix-vector multiplication can be done faster than $O(n^2)$. In that case, the orthogonalization may become the task that dominates the computation.

Task	Time complexity	
Matrix-vector multiplication	$kf(n)$	flam
Orthogonalization	$kO(nm^2)$	flam
Ritz pair computation	$k[O(nm) + O(m^3)]$	flam
Residual computation	$kO(nm)$	flam
Candidate selection	$kO(m^2)$	flcm
Restarting	$\frac{k}{m}[O(nms) + O(m^3)]$	flam
Deflation	$s[O(nm^2) + O(m^3)]$	flam

Table 3.2: Time complexity for each task in the RA mode.

The above observation may suggest that we shrink the subspace size m to reduce the cost of orthogonalization. However, there are many tradeoffs in the setting of these parameters. For example, a large subspace size m may, but not necessarily [12], reduce the number of iterations k, which could accelerate total processing time. But the time complexity of orthogonalization grows quadratically with m and the time complexity of Ritz pair computation, restarting, and deflation grows cubically with m. How to choose a proper m is still an interesting research topic.

3.4.2 The SIRA mode

The SIRA mode requires an additional step in the algorithm, the residual enhancement. Therefore, except for the original cost of the algorithm, it requires extra $g(n, \epsilon)$ in each iteration of the second stage. Also, when the targets are complex, it doubles the workloads of the residual computation and orthogonalization. The overall time complexity for the SIRA mode is

$$\begin{array}{ll} kO(nm^2) + kf(n) + (k-m)g(n,\epsilon) & \text{flam} \\ kO(m^2) & \text{flcm} \end{array} \tag{3.14}$$

At the first glance, one might find the SIRA mode is more expensive than the RA mode because of the additional term $(k - m)g(n, \epsilon)$. However, this may not be true since the SIRA mode usually has smaller k if a shift is correctly selected. The beauty of the SIRA mode is the function $g(n, \epsilon)$ can be very small, because the require precision ϵ is low. We will see some experimental results in the next chapter for this point.

3.4.3 Memory requirements

In RAPACK, the different computational modes have the same memory requirement. In the beginning of the process, RAPACK allocates the maximum required storage, which can be known from the order of the matrix n and the maximum dimension of the subspace m. The major storage requirement is for the two subspace matrices U and V, each of which is a $n \times (m+1)$ array. The reason for the extra column is for the case when the last target is complex.

Beside that, RAPACK allocates several $n \times 2$ arrays, U_1, V_1,R, Z and one $n \times 1$ array r. Matrix U_1 is the orthogonalization result and V_1 is for holding Au. Matrix R is for the residual, and Z is a scratch array used in orthogonalization and residual computation. The reason for the double columns is also to handle complex eigenpairs. The vector r is a scratch array that is used in the CSI method.

Three $(m+1) \times (m+1)$ real arrays are also allocated in RAPACK. They are designed for storing the Rayleigh quotient H, the eigendecomposition for H and the Schur decomposition for H. Two complex arrays, sized $(m+1) \times 1$, are used in holding the eigenvalues. An integer array with the same size is for the sorting result.

Overall, the storage requirement of RAPACK is $n \times (2m+11) + 3(m+1) \times (m+1)$ real, $2m+2$ complex and $m+1$ integer.

3.5 Implementation

RAPACK, implemented in Fortran 95, consists of five modules. `RAcontrol` defines the parameters and the variables that regulate the action of data flow and the iteration between functions. `RAutil` includes all the auxiliary subroutines and functions used by all the package. `RAsubspace` consists of a set of subroutines for subspace operations. `RAtarget` performs target management. `RAnsreal`, the major module that should be included in user programs, provides all the required subroutines for solving real nonsymmetric eigenproblems.

In this section, the details of implementation and the design philosophy of RAPACK will be discussed. First, we treat reverse communication, a technique for obtaining the assistance from the user's calling program. Then, the design of the main subroutine, RA, will be treated. Next, we will discuss the program configuration and candidate. Several functions that help users to obtain information about computational results will be treated in the last subsection.

3.5.1 Reverse communication

Reverse communication is needed because of the diverse representation of large matrices and the need for RAPACK to have the result of a variety of matrix operations. RAPACK also uses reverse communication in target selection and error handling.

The mechanism of the reverse communication simulates function invocation in general programming languages, except in the reverse direction. In the normal situation, when a program makes a function call, the system will push the current content of the program and the return address into stack, and then jump to the beginning of function when the function returns, the program will continue the execution from the return address. In reverse communication, the caller itself is a subroutine, and the callee is the main program. The function invocation is replaced by the return statement, and the return takes place when the subroutine is called again. When a subroutine initiates a reverse communication, it must specify which function the main program should perform before returning and pass the necessary *arguments* to the main program. This information can be communicated through the returned values or the global variables. During the reverse communication, the content of the subroutine and the return address is stored in variables in caller's scope or static local variables so that when the subroutine is called again, its content can be retained and the execution can be continued in the right place.

It is critical in the reverse communication process to ensure the correctness of the calling sequence. Unlike a normal function call, the caller, which is a subroutine, does not have any control on the callee, the main program. Therefore, the correct operation and the expected action on the arguments cannot be guaranteed. The user program could misuse variables or even destroy the content of the caller during the reverse communication. RAPACK employs two mechanisms to guard against these potential problems. First, all variables used in the computation are declared private. Actions on these variables can be only performed through predefined function calls. Second, RAPACK defines a standard calling sequence for reverse communication, which is as follows

1. The user program calls `QueryRequest` to obtain the request from RAPACK.
2. According to the request, the user program calls `GetRequest` to retrieve the arguments.
3. The user program performs the requested computation on the arguments.
4. After the computation, the user program calls `ReturnRequest` to return the computed results.
5. The user program calls the main subroutine `RA`.

Here `QueryRequest`, `GetRequest`, `ReturnRequest` and `RA` are subroutines defined in RAnsreal.

This calling sequence is guarded in two ways. First, during the reverse communication, the variable `request_state` keeps track of the actions that user programs perform. In the `RA` subroutine, `request_state` is equal to `REQUEST_NONE`. When user program calls `QueryRequest`, it becomes `REQUEST_ISSUED`. After user program gets the arguments from `GetRequest`, it turns to `REQUEST_SET`; and after the user program returns the result by calling `ReturnRequest`, `request_state` is set to `REQUEST_DONE`. Once the `RA` subroutine is called, it changes back to `REQUEST_NONE`. During the calling sequence, if `request_state` does not have correct values, RAPACK will give an error message and stops the program.

Another safeguard is that `GetRequest` and `ReturnRequest` check the data types of the arguments. For example, if RAPACK requests a real matrix vector multiplication, but user program passes a complex vector in `ReturnRequest`, RAPACK will regard it as an fatal error and stop the program.

3.5.2 The RA subroutine

The main subroutine in RAPACK, the RA subroutine, is implemented as a finite state machine. This implementation has several advantages. First, the reverse communication fits in the scheme naturally, because the next state serves the the return address in reverse communication. Second, the program in the subroutine is divided into into small blocks, each with a meaningful state label,which gives a clear overview of the algorithm.

The important decision for this design is how the program is partitioned. Figure 3.1 shows the states and the state transition diagram of `RA` subroutine. The state transition diagram in RAPACK can be viewed as a data flow graph, in which the nodes represent computation and the edges represent control flow [1]. The state transition diagram gives a simple overview of the entire process.

3.5.3 Program configuration

RAPACK provides two kinds of methods to configure system parameters. One sets the configurations through a function call; another uses a configuration file. The latter option has the advantage that users can change the configuration as without recompiling the program. The ability to change program parameters at run time is useful in many applications, as well as testing and tuning. Some programming languages, like Java, utilize the system properties to configure runtime parameters. Unfortunately, Fortran does not support such a mechanism. Instead, program configuration must be done before invoking the `RA` subroutine. This sequence of actions is enforced by RAPACK, through a technique similar to that used in reverse communication. An additional state is added to monitor whether package has been configured before the `RA` subroutine is entered. A failure to have configured will cause the termination of the program.

3.5.4 Candidate management

A candidate in RAPACK is not just a Ritz pair; it is an object with many attributes. Besides the Ritz value and the corresponding eigenvector of the Rayleigh quotient, it includes the norm of the residual of the Ritz pair. It also has the ranking attribute that represents the preference made by the candidate selection algorithm. Moreover, a candidate has some logical markers that indicate whether it has converged, or if it is to be kept in the restart subspace.

The `RAtarget` module is designed to manage the candidates. Instead of defining a data type for individual candidates and forming an array of that type for all the candidates, `RAtarget` stores each attribute separately. This allows RAPACK to manipulate each attribute more efficiently, as in the computation of primitive Ritz vectors or the sorting of Ritz values. Under this scheme, each candidate is identified by an index which indicates its

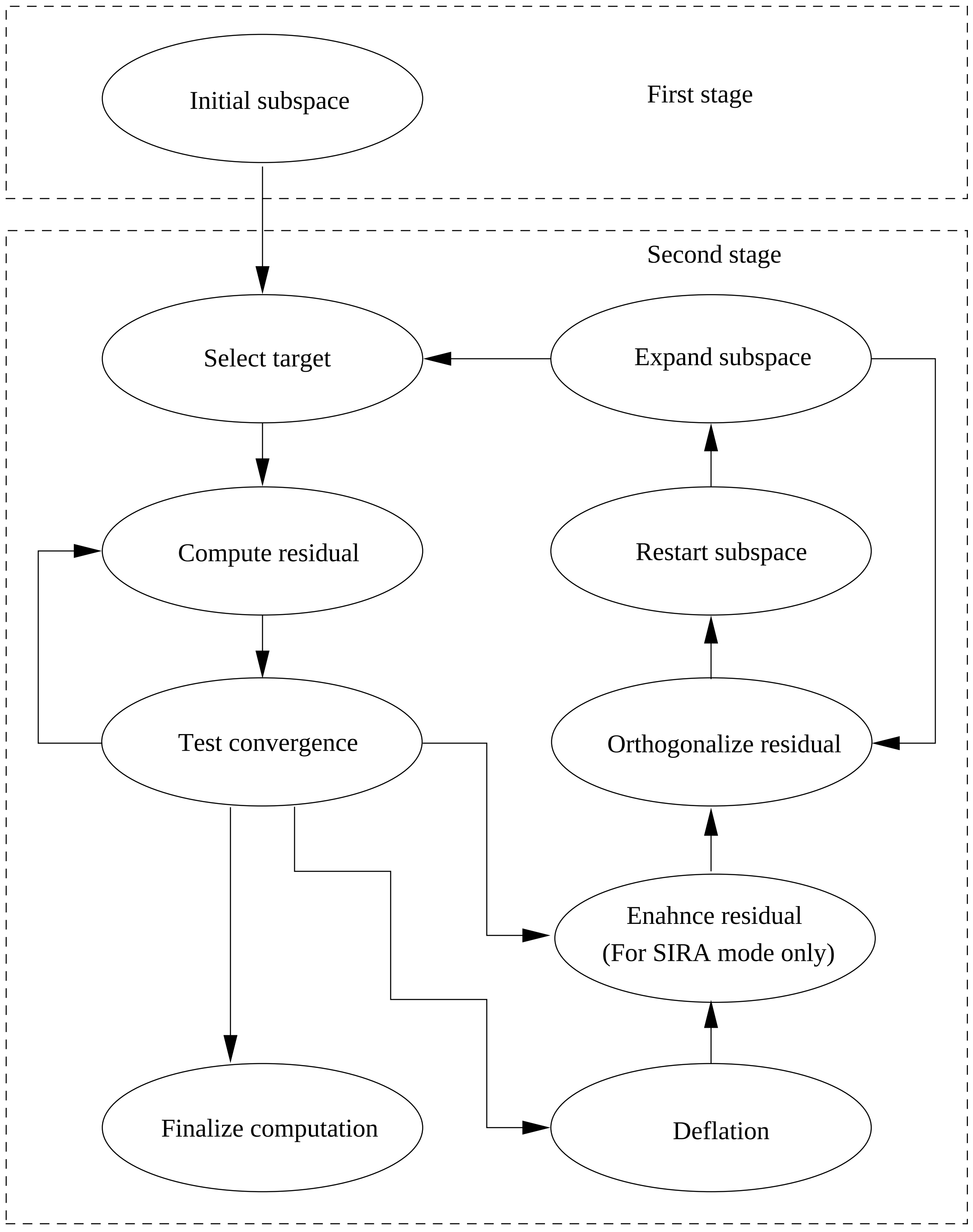

Figure 3.1: The state transition diagram of the RA subroutine.

location in the arrays where its attributes are stored. Unfortunately, this kind of implementation makes access to the attributes of each candidate more complicated. To mitigate the inconvenience of data access, RAPACK provides functions or subroutines for data retrieval, so that the caller need not worry about the underlying implementation.

3.5.5 Computational information

Intermediate computational results can be valuable for programs using RAPACK. For example, users should not have to wait for the termination of a run, which may take days or longer, to find out that convergence has stagnated. Instead, they should be able to stop the run or change parameters dynamically once problems are detected.

RAPACK offers several kinds of information about the course of the computation: the current configuration, intermediate results, debugging information, and some statistics. The current configuration can be acquired by the functions `QueryIntegerParameter`, `QueryTolerance` and `QueryShiftValue`. It is needed if parameters are configured dynamically. Intermediate results can be used not only to monitor the progress of the run, but also in adaptive algorithms that make computational decisions depending on the current computed results. Such information is supplied by the subroutine `GetResult`. Debugging information, a trace of computational results, can be output to a file by setting the parameter `DEBUG_FILE` to the name of the file. Operation counts, such as the number of reverse communications, may be obtained by using the function `GetStatistic`.

3.6 Integration with other methods

RAPACK can cooperate in the implementation of other methods. In this section, we will consider three such methods. The first one is the inexact Krylov method, which relaxes the precision required of matrix operations as the approximation converges. The second is the successive inner-outer residual Arnoldi process, which utilizes the convergence properties of Krylov subspaces to reduce the total cost. The third is the complex shift-invert method proposed by Parlett and Saad [28]. Each method will be discussed in an individual subsection. Experiments will be presented in Chapter 4.

3.6.1 Inexact Krylov method

The inexact Krylov method is an Arnoldi-like method that allows matrix-vector multiplication to be performed with decreasing accuracy as the approximation converges. The inexact Krylov method for solving linear systems has been studied in several places [8, 34, 13]. For eigenproblems, some results can also be found in [18, 34].

In RAPACK, matrix-vector multiplication is performed every time that the subspace is expanded by a vector. RAPACK maintains two subspace matrices, U and V. Matrix U is the orthonormal basis of the generated subspace; and matrix V is equal to AU, which means for every column u_i in U, there is a corresponding $v_i = Au_i$. Matrix V serves two purposes: the computation of Rayleigh quotient and the computation of residual, as we have presented in section 3.1.

Now, if the matrix-vector multiplication is computed inexactly in every iteration, say $v_i = Au_i + f_i$, then $V = AU + F$, where $F = (f_1, f_2, \cdots)$. The introduced errors will perturb the Ritz pairs and its residual. Let $(\tilde{\mu}, \tilde{y})$ be an eigenpair of computed Rayleigh quotient $U^*V = U^*AU + U^*F$. The first order perturbation theory of eigenvalue [41] shows there is a corresponding eigenpair (μ, y) of the actual Rayleigh quotient U^*AU, such that

$$\begin{aligned} \|\tilde{y} - y\| &\leq \kappa(y)\|Fy\| \\ |\tilde{\mu} - \mu| &\leq \kappa(\mu)\|Fy\|, \end{aligned}$$

where $\kappa(y)$ and $\kappa(\mu)$ are condition numbers of y and μ respectively. Here we assume that $\kappa(y)$ and $\kappa(\mu)$ are bounded by some constant C_1 and write

$$\begin{aligned} \tilde{y} &= y + \Delta y \\ \tilde{\mu} &= \mu + \Delta\mu, \end{aligned}$$

in which $\|\Delta y\| \leq C_1\|Fy\|$ and $|\Delta\mu| \leq C_1\|Fy\|$. Then, the residual that used in subspace expansion can be expressed as

$$\begin{aligned} \tilde{r} &= V\tilde{y} - \tilde{\mu}U\tilde{y} \\ &= (AU + F)(y + \Delta y) - (\mu + \Delta\mu)U(y + \Delta y) \\ &= (AUy - \mu Uy) + Fy + V\Delta y + \Delta\mu U\tilde{y} \\ &= r + Fy + V\Delta y + \Delta\mu U\tilde{y} \end{aligned}$$

The vector $r = AUy - \mu Uy$ is the exact residual. Therefore

$$\|\tilde{r} - r\| \leq C_2\|Fy\| + O(\|Fy\|^2),$$

for some constant C_2. From the theory in Chapter 2, if the relative error

$$\frac{\|\tilde{r} - r\|}{\|r\|} \leq \epsilon$$

for some constant $\epsilon \leq 1$, the residual Arnoldi method can converge. Here we assume $O(\|Fy\|^2)$ is small enough to be ignored. A sufficient condition for the convergence is $\|Fy\| \leq \epsilon C_2^{-1}\|r\|$. Since this condition has to hold for every iteration, if it takes k iterations for the residual Arnoldi method to converge to the precision τ, the condition can be more specifically rewritten

$$\|F_k y_k\| \leq \epsilon C_2^{-1}\tau, \tag{3.15}$$

where $F_k = (f_1, f_2, \cdots, f_k)$ and $y_k = (\eta_1, \eta_2, \cdots, \eta_k)^T$ is the actual primitive Ritz vector in the kth iteration. If we evenly divide the error $\epsilon C_2^{-1}\tau$ into k iterations, the bound for the error

$$\|f_i\| \leq \frac{\epsilon\tau}{C_2|\eta_i|k} \tag{3.16}$$

is sufficient to make (3.15) hold.

The above equation does not give a practical formula, since the number of iterations k to reach the precision τ is unknown, and y_k is also unavailable. Let $\theta_1, \theta_2, \cdots$ be the tolerable errors of matrix-vector multiplication for iteration $1, 2, \cdots$. As reported in [18], the convergence of inexact Krylov method is very sensitive to θ_i. A large θ_i could prolong the entire process or even cause stagnation. But small θ_i do not benefit the process much. Here we give a heuristic bound for $\|f_i\|$

$$\theta_i = \max\{\tau, \frac{\delta\tau}{\|r_i\|m}\}, \tag{3.17}$$

where δ is some constant. Comparing the above bound to (3.16), we estimate η_i by the norm of the residual of current candidate, $\|r_i\|$; and replace k by the maximum dimension of the subspace m which is used in the subspace restarting algorithm. Constant ϵ/C_2 is estimated by δ. In the next chapter, we will see how well this formula works.

When more than one eigenpair is desired, the situation becomes even more complex, because the errors valid for one eigenpair approximation may cause stagnation for other eigenpairs. Fortunately, this problem can be resolved by the residual Arnoldi method, since it can make the approximation converge as long as the corresponding eigenvectors of the perturbed matrix can still have converging Ritz vectors. Therefore, the strategy is simple: we only consider one Ritz pair at a time, the candidate. We just estimate the tolerable error for current candidate. Although these errors may cause other approximations to stagnate, the stagnation is temporary. When candidates are switched, the new one will start the convergence. If the number of desired eigenpairs increases, the parameter δ should be set smaller to safeguard the convergence.

3.6.2 Successive inner-outer residual Arnoldi process

The successive inner-outer Krylov (or Lanczos) method is based on three observations [18].

1. If we compute matrix-vector multiplication to precision τ every time in Krylov subspace methods, the achievable accuracy of computed eigenpairs will be roughly equal to τ.

2. Suppose we are given two different precisions requirements τ_1 and τ_2 with $\tau_1 > \tau_2$. If we run the Arnoldi process first with precision τ_1 and then with precision τ_2, then the convergence curve of the approximations corresponding to τ_2 will be almost identical to those corresponding to τ_1 before they reach τ_1.

3. The approximations in Krylov methods converge superlinearly.

The successive inner-outer Krylov (or Lanczos) method divides a run into a sequence of stages. Each stage computes using an increasing precision. For example, if the final precision required is 10^{-12}, then we could divide the process into four stages, and compute with precision 10^{-3}, 10^{-6}, 10^{-9} and 10^{-12}. In each stage, the subspace is restarted with the best eigenvector approximation as an initial vector. In the early stage, we could compute the matrix-vector multiplication to a low accuracy, since the required precision is low. Then, we increase the precision requirement of matrix-vector multiplication as we proceed to next stage. Since the approximations in Krylov methods converge superlinearly, one can expect the latter stages take fewer iterations to complete. Therefore, even though the latter stage needs more time to compute matrix-vector multiplication more precisely, the total time is reduced.

The matrices used in each stage can be also viewed as a parameterized matrix,

$$\tilde{A}(\tau_i) = A + E(\tau_i), \tag{3.18}$$

where $E(\tau_i)$ is a random matrix with norm no greater than τ_i.

The drawback of this process is the convergence is local, which means this process can only compute one eigenpair at a time. When the subspace is restarted, even though it may contain good approximations for many eigenpairs, only one approximation is kept in the restarting subspace. The reason is that the Krylov subspace for $\tilde{A}(\tau_i)$ is not a Krylov subspace for $\tilde{A}(\tau_{i+1})$. And if we give up the property of Krylov subspace, which means we recompute the Rayleigh quotient for $\tilde{A}(\tau_{i+1})$, and start the Arnoldi process from the last column of the subspace, the convergence property in the second observation above will not hold. The convergence curve will look like a staircase, in which the superlinear convergence property is also lost.

This problem can be solved easily in RAPACK because the RA method use an arbitrary subspace to start.

3.6.3 The complex shift-invert method

In [28], Parlett and Saad showed that either the real part or the imaginary part of a complex shift-inverted matrix can be used for subspace expansion. Let $\sigma = \alpha + i\beta$ be a complex shift value. Then the shift-inverted matrix $S = (A - \sigma I)^{-1}$ is complex. Let $S = S_r + iS_i$, where S_r and S_i are real matrices. Parlett and Saad proved that

$$\begin{aligned} S_r &= (A - \alpha I)(A - \bar{\sigma} I)^{-1}(A - \sigma I)^{-1} \\ S_i &= \beta(A - \bar{\sigma} I)^{-1}(A - \sigma I)^{-1}. \end{aligned}$$

Therefore, one can use either the real part or the imaginary part in the subspace expansion.

In RAPACK, this algorithm can be integrated into the RA mode. Specifically, whenever a matrix-vector multiplication request is made, the user program solves a complex linear system, and returns only the real part or the imaginary part as the answer. In the RA mode, RAPACK only requests matrix vector multiplication for a real vector and accepts a real vector as result. It does not care how the vector is generated.

The algorithm is briefly described as follows

1. Solve the complex linear system $(A - \sigma I)v = u$ when a matrix vector multiplication is requested in the RA mode.
2. Return the real part (or the imaginary part) of v.

One must consistently use the real or the imaginary part throughout the process. Otherwise the Rayleigh quotient, which is a cumulated result of a series of matrix-vector multiplication, cannot be computed correctly, since the matrix itself is not consistent.

The only difficulty with this method is to recover the original eigenvalue approximations from the current approximations. Assume the imaginary part is used in subspace expansion and (μ, z) is an eigenpair approximation of the matrix S_i where $\sigma = \alpha + i\beta$ is the shift value. The relation of μ and the corresponding eigenvalue λ of A is

$$\mu = \frac{\beta}{(\lambda - \sigma)(\lambda - \bar{\sigma})},$$

or equivalently

$$\lambda = \alpha \pm \sqrt{\frac{\beta}{\mu} - \beta^2}.$$

The two formula for λ are not conjugate. Only one of them is the correct eigenvalue approximation, and there is no simple way to tell. There are two possibilities. The first one is to select the solution that is close to the shift. It is simple, but can fail. The second is to compute the eigenvalue approximations as the Rayleigh quotient of the approximated eigenvector $\lambda = z^* A z$, which is more expensive but more reliable.

Chapter 4. Experiments

This chapter contains several experiments designed to compare RAPACK with other packages. Section 4.1 introduces the package Eigentest, which generates some of the test matrices we will use to evaluate RAPACK. Section 4.2 compares the performance of RAPACK and ARPACK [22]. Section 4.3 evaluates the effect of using an arbitrary starting subspace for solving eigenvalue problems. Here we will compare RAPACK and SRRIT [3]. Section 4.4 contrasts the SIRA mode in RAPACK with the inexact Krylov method, since both methods can tolerate errors in the computation. Section 4.5 conducts experiments on different strategies that can be used with a complex shift in shift-invert enhancement for real matrices.

4.1 Eigentest

Eigentest is a package that generates matrices for testing large eigenproblems, and provides subroutines for related operations [21]. The test matrices have good scalability because their storage and the time complexity of the operations increase linearly with the order. Eigentest also allows users to manipulate the condition of individual eigenpairs or a group of eigenpairs of the test matrices. Currently, Eigentest is implemented in Fortran 77, Fortran 95, C, and Matlab.

The test matrices generated by Eigentest are called eigenmats. They are generated in factored form, which we will now describe. First, an eigenmat A of order n is produced from the factorization,

$$A = XLX^{-1},$$

where L is a (block) diagonal matrix containing eigenvalues of A, and X contains its eigenvectors. The matrix X is in turn a product of two matrices, Y and Z,

$$X = YZ,$$

which are used to control the condition of eigenpairs. Specifically, matrix Y is generated from a singular value decomposition,

$$Y = U\Sigma V^*, \tag{4.1}$$

in which Σ is diagonal with nonnegative singular values of Y as its diagonal elements, and U and V are Householder transformations,

$$U = I - 2uu^* \text{ and } V = I - 2vv^*,$$

where u and v can be either unit vectors or zero vectors, and I represents the identity matrix.

The matrix Z is a block diagonal matrix,

$$Z = \begin{pmatrix} Z_1 & & & \\ & Z_2 & & \\ & & \ddots & \\ & & & Z_k \end{pmatrix}.$$

Each block Z_i is constructed as in (4.1). Each Z_i corresponds to one or more blocks of L and is used to control the condition of individual groups of eigenvalues.

The operations supported in Eigentest includes $(A - sI)B$, $(A - sI)^T B$, $(A - sI)^{-1}B$, and $(A - sI)^{-T}B$, where s is a shift, B is a matrix (or a vector if B has only one column). These routines are usually invoked by the subspace methods based on matrix vector multiplication. Eigentest also provides a function to extract a specific eigenvector, both left and right, and to compute the condition of its eigenvalue.

4.2 ARPACK and RAPACK

ARPACK is a numerical package for solving large eigenvalue problems, which implements the Implicitly Restarted Arnoldi Method (IRAM) [36]. The package is well designed and its typical performance is quite satisfactory. It provides functions not only for solving the standard eigenproblem, but also for solving generalized eigenproblem, and computing singular value decompositions. In addition, the package includes all subroutines designed to handle symmetric, nonsymmetric and complex. The recent development of

PARPACK and ARPACK++ [23] makes it an extensive and popular package.

To simplify the comparison, only two computational modes, mode 1 and mode 3, of ARPACK, corresponding to the RA mode and the SIRA mode of RAPACK, are treated. The following subsections present each experimental setting and results separately.

4.2.1 Mode 1 and the RA mode

There are many similarities between the mode 1 of ARPACK and the RA mode of RAPACK. First, both methods are designed to compute eigenpairs whose eigenvalues are on the periphery of the spectrum. Second, both methods requires only matrix vector multiplication. In fact, restarting strategies aside, if they use the same starting vector, they generate the same subspaces, at least mathematically [Chapter 2].

The experiment uses a real nonsymmetric eigenmat A of order 10000, whose first 100 eigenvalues are $1, 0.95, 0.95^2, \cdots, 0.95^{99}$, and the rest are randomly distributed in $(0.25, 0.75)$. The eigenvectors of A are determined by matrices Y and Z, as described in the previous section. The matrix Y is randomly generated with singular values uniformly distributed in $(0, 1)$. The matrix Z is consisted of three diagonal blocks: the first block and the third block, corresponding to the ten largest eigenvalues and the ten smallest eigenvalues respectively, have condition number 10^5. The second block is an identity matrix. This choice will tend to increase the ill-conditioning of the eigenvalues on the periphery of the spectrum.

The problem is to find the six largest eigenvalues of A. We measure two aspects of the calculation. One is the elapsed time in seconds, denoted ETIME, and the other is the total number of matrix-vector multiplications, denoted TMVM [18]. For both packages, we let the maximum dimension of subspaces be 20, and the convergence criterion be 10^{-13}, which means the norms of the residuals of computed eigenpairs must be less than 10^{-13}. Both packages start with the same initial vector, which is a randomly generated unit vector.

Table 4.1 displays the experimental results for both methods. It is not surprising that RAPACK spends more time than ARPACK; in each subspace expansion RAPACK needs extra time to compute Ritz vectors and residuals. For real-world problems, the cost of residual computation is usually much less than the cost of matrix-vector multiplication, and both packages would

	Mode 1 of ARPACK	Mode RA of RAPACK
ETime	4.6860	8.4242
TMVM	113	138

Table 4.1: Mode 1 of ARPACK and mode RA of RAPACK.

achieve similar performance.

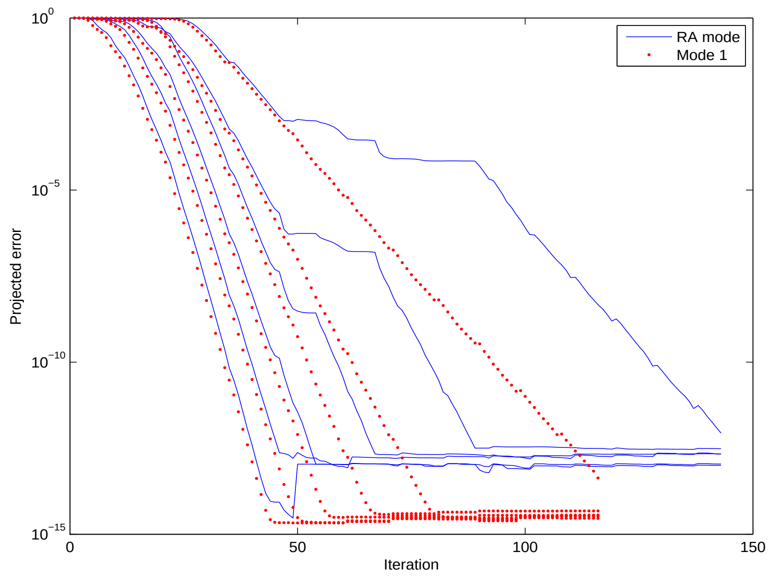

Figure 4.1: The projection error of approximated subspace.

The more interesting problem is that RAPACK requires more matrix-vector multiplication than ARPACK, since theoretically, both methods generate the same subspace. Of course, more than one factor influences this phenomenon, since the entire process involves restarting, deflation and other computations. Figure 4.1 illustrates this phenomenon, in which we measure $\|(I - U_kU_k^*)x\|$, where U_k is an orthonormal basis for the current subspace and x is the target eigenvector. The six solid lines represents the projection errors in the subspaces generated by ARPACK, while the dotted lines are for the subspaces generated by RAPACK. As can be seen, the convergence of the first three eigenvectors is almost identical. But the remaining curves stagnate once the third one reaches a level of around 10^{-13}. After the candidate is switched, convergence resumes. Therefore, there are some staircase phenomena on the convergence of latter approximations.

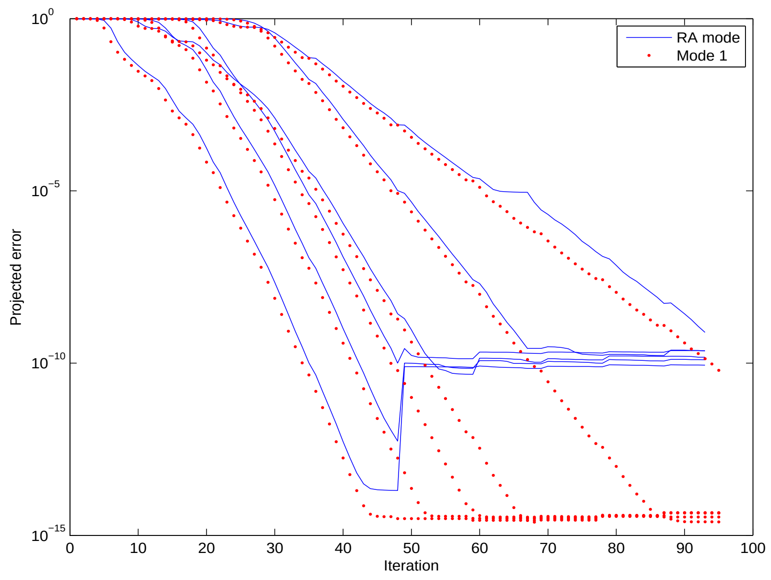

Figure 4.2: The projection error of approximated subspace.

If we lower our precision requirement to 10^{-10}, then the projection error of desired eigenpair in the generated subspaces are as in figure 4.2. Before iteration 50, approximations in two subspaces have similar convergence. After the deflation of the third eigenpair approximation, all the projection errors that are originally lower than 10^{-10} rise to 10^{-10}. However, the staircase phenomena disappear. Every approximation converges smoothly.

The reason for this problem is fundamental. In ARPACK, the subspace is expanded by Au_k, where u_k is the last vector of U_k. The Arnoldi relation

$$AU_{k-1} = U_{k-1}H_{k-1} + \beta_{k-1}u_k e_{k-1}^*$$

indicates that u_k is in the direction parallel to all the residuals of eigenvector approximations. The error only comes from orthogonalization process. For the RAPACK, the subspace is expanded by the residual. As the candidate residual diminishes, the error from residual computation becomes relatively large, which destroys the Arnoldi relation. When the accumulated error exceeds the tolerable range of an approximation, it will stagnate, as shown in Figure 4.1. However, when the stagnated approximation becomes a candidate, most errors will not affect the convergence, as we have seen in Chapter

2. Hence, the convergence can be restarted. If the accumulated error is never large enough to affect the convergence of desired approximations, the residual Arnoldi method will have almost the same performance as the Arnoldi process, as the case in Figure 4.2.

4.2.2 Mode 3 and the SIRA mode

ARPACK and RAPACK differ significantly when shift-invert enhancement is applied. In ARPACK, linear systems must be solved to the required precision of desired eigenpairs. Direct methods, such as LU or Cholesky[38, 16, 33, 9], give satisfactory accuracy. But they are expensive when the order of the matrix is large. On the other hand, RAPACK allows imprecise solutions of linear systems, which can substantially reduce the cost when iterative methods for linear systems are used.

In this subsection, the mode 3 of ARPACK and the SIRA mode of RAPACK are compared. The test case is the same as the previous subsection, in which the maximum dimension of subspace is 20, and the dimension of restarting subspace is 6. The difference is we request the 6 smallest eigenvalues this time. The linear systems are solved by restarted GMRES [32], which is implemented in the library for the book: *Template for the Solution of Linear Systems* [4].[2] The restarting subspace for the linear solver is set to 40. The shift is zero. The only different setting is the precision requirement for linear system solving. In ARPACK, it is set 10^{-13}, and it is 10^{-3} in RAPACK.

	Mode 3 of ARPACK	Mode SIRA of RAPACK
Etime	378	168
TMVM	11842	4606
OUTER-ITER	68	144
AVG-INNER-ITER	174	33

Table 4.2: Mode 3 of ARPACK and mode SIRA of RAPACK.

Table 4.2 lists the experimental results. The table gives the elapsed time (ETIME) in seconds and the total number of matrix vector multiplication

[2]The GMRES code was modified for this experiment, because the original program has several serious bugs.

(TMVM). The outer iteration[3] (OUTER-ITER) represents the total number of reverse communications invoked by ARPACK or RAPACK to perform the linear system solving. The average inner iteration (AVG-INNER-ITER) is the average number of matrix vector multiplications spent in solving each linear system. For ARPACK,

$$\text{TMVM} = \text{OUTER-ITER} \times \text{AVG-INNER-ITER}.$$

For RAPACK,

$$\text{TMVM} = \text{OUTER-ITER} + \text{OUTER-ITER} \times \text{AVG-INNER-ITER},$$

because in each outer iteration, RAPACK also needs one matrix vector multiplication.

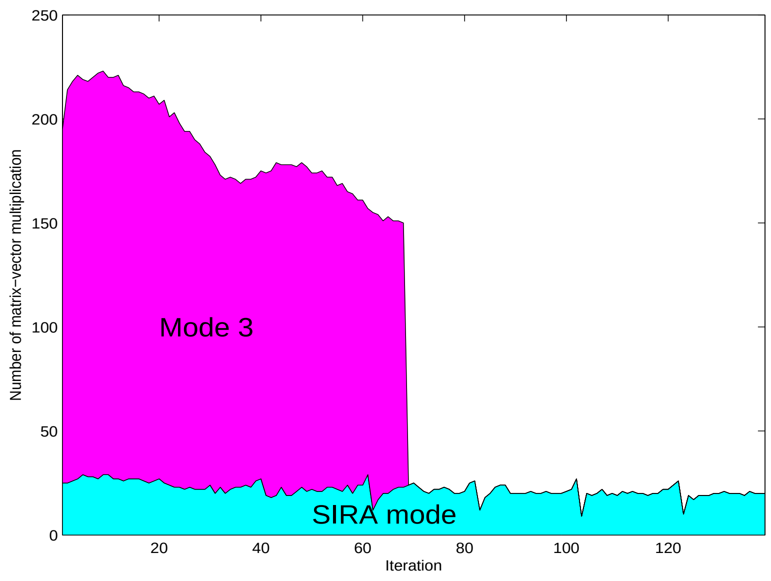

Figure 4.3: The number of matrix-vector multiplications for mode 3 and the SIRA mode.

The number of matrix-vector multiplication required by each iteration is displayed in Figure 4.3. The area represents the total number of matrix-vector multiplications. Although SIRA mode does need more iterations to

[3]RAPACK and ARPACK have different definition of iterations. One iteration defined in ARPACK is the process between implicit restarting. The total number of iterations in ARPACK corresponds to the number of subspace restartings in RAPACK. Here we use the definition of RAPACK.

computed the required eigenpairs, the cost for each iteration is relatively cheap. This result shows the superiority of RAPACK when shift-invert enhancement is applied. On the average, the required matrix-vector multiplications in one iteration of RAPACK is less than one-fifth that of ARPACK.

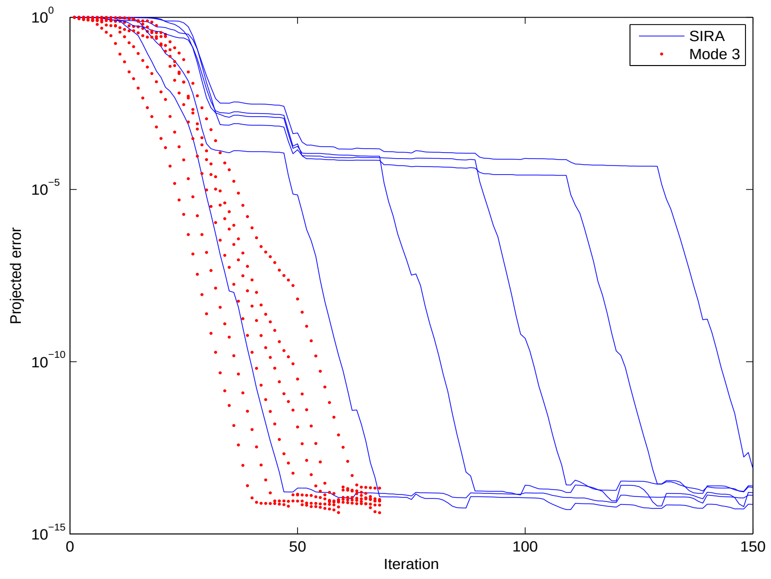

Figure 4.4: The convergence of mode 3 and the SIRA mode.

Figure 4.4 shows the convergence of both methods, which is also measured by the projected error $\|(I - U_k U_k^*)x\|$. The solid line represents the convergence of mode 3, and the dotted line is for the SIRA mode. Since we explicitly allow 10^{-3} errors in the SIRA mode, approximations that are not targeted stop converging around the level of error, until they are selected to be candidate.

4.3 SRRIT and RAPACK

As shown before, the RA mode of RAPACK is not as good as the existing package. However, its ability to use a subspace, not necessary Krylov, that contains certain eigenvector approximations as its initial basis is not usually found in Krylov based methods. To our knowledge, only SRRIT [3, 37] has similar ability. In this section, we will compare these two packages.

The test problem is the same as before: a 10000×10000 Eigenmat where the first 100 eigenvalues are $1, 0.95, 0.95^2, \cdots, 0.95^{99}$ with the rest uniformly distributed in $(0.25, 0.75)$. As previously, we still compute the 6 smallest eigenvalues of it. The difference is that, in this experiment, we use both packages to implement the successive inner-outer strategy [18]. As briefly described in Chapter 3, this method divides the process into stages, and the successive stages have increasing precision requirements. Except for the first stage, in which the subspace is created afresh, each package will use the subspace computed in the previous stage as an initial subspace for the current stage. The ratio of the precision requirement between two consecutive stages is called the step ratio. In this experiment, we use constant step ratios. To make the experiment interesting, we let the final precision requirement be 10^{-12}, so that we could compare the effect of various step ratios: 10^{-2}, 10^{-3}, 10^{-4}, 10^{-6} and 10^{-12}.

The setting of each package is described as follows. For RAPACK, the RA mode is used, but when the request for a matrix vector multiplication arrives, we solve the shifted linear system, $(A - \sigma I)x = b$, to the precision of current stage. In this experiment, the linear systems are not solved by GMRES. Instead, the corresponding function in Eigentest is used, because the purpose of this experiment is to compare the ability to reuse existing subspaces. The remaining settings of RAPACK, such as the maximum subspace size, are the same as before. For SRRIT, we also use the shift-inverted matrix. The dimension of the subspace is 20.

Figure 4.5 displays one of the experimental results of RAPACK, in which the step ratio is 10^{-3}. The entire process contains four stages with precision requirement 10^{-3}, 10^{-6}, 10^{-9} and 10^{-12}. The y-axis in the figure denotes the norm of residuals, and the x-axis is for the number of iterations. The line segments in each group stands for the convergence of six targeted eigenpairs in the computed order. As can be seen, these line segment can be partitioned into four groups, each of which represents a stage. The last four lines in the first group are shorter than in the others, because the subspace in the first stage is a Krylov subspace of the original matrix, for which approximations can converge simultaneously. This global convergence property does not apply to the approximations of other stages.

Table 4.3 summarizes all the experimental results for RAPACK and SRRIT. Each column represents the result of a specific step ratio. The items inside table indicate the number of iteration taken to finish a stage. For example, in the first table, which is for the result of RAPACK, the data in

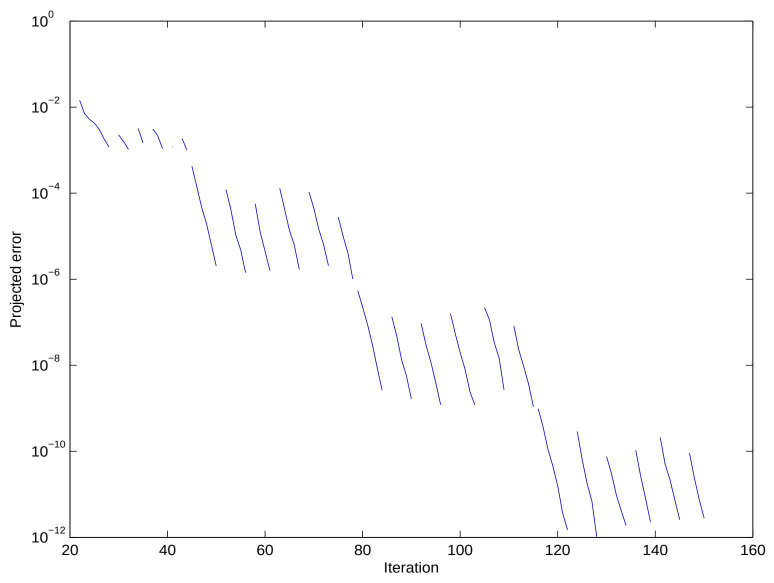

Figure 4.5: Experimental result of RA mode with the successive inner-outer process. (step=10^{-3})

the second column (step = 10^{-3}), 39, 40, 40 34, represent the number of iterations spent in the four stages. One can compare this result to Figure 4.5.

The data in Table 4.3 verifies that both methods can take advantage of reusing the existing subspace. On the average, RAPACK outperforms SRRIT for this particular problem, because it requires much fewer iterations in each stage. It should be noted that the number of iterations does not reflect the total effort in the computation, since the cost of one iteration in the latter stage would be more expensive than that of the early stage.

4.4 Inexact Krylov method and the SIRA mode

In this section, we want to compare the SIRA mode with another subspace method that is also capable of tolerating errors, the inexact Krylov method. For the SIRA mode, the tolerable error is a constant. For the inexact Krylov method, the allowed error is increasing as approximations are converging. Note that, unlike the SIRA mode in RAPACK, the inexact Krylov method can tolerate errors not just from linear system solving, but also from matrix

No of Stages	6	4	3	2	1
Step	10^{-2}	10^{-3}	10^{-4}	10^{-6}	10^{-12}
10^{-2}	27				
10^{-3}		39			
10^{-4}	31		32		
10^{-6}	31	40		39	
10^{-8}	31		53		
10^{-9}		40			
10^{-10}	31				
10^{-12}	32	34	52	68	60
total	183	153	140	107	60

(a) Experimental result of RAPACK

No of Stages	6	4	3	2	1
Step	10^{-2}	10^{-3}	10^{-4}	10^{-6}	10^{-12}
10^{-2}	275				
10^{-3}		305			
10^{-4}	120		360		
10^{-6}	120	120		432	
10^{-8}	120		200		
10^{-9}		152			
10^{-10}	120				
10^{-12}	120	213	217	320	784
total	875	783	777	752	784

(b) Experimental result of SRRIT

Table 4.3: Comparison of RAPACK and SRRIT for successive inner-outer process with various number of stages.

vector multiplication, which appears in many applications [43, 45].

The problem setting is identical to the one in previous section, in which the matrix A is a 10000×10000 real nonsymmetric matrix with exponential declining eigenvalues $1, 0.95, 0.95^2, \ldots, 0.95^{99}$, and others are just randomly generated in $(0.25, 0.75)$. The targeted spectrum consists of the six smallest eigenvalues. Both methods employ shift-invert enhancement with shift $= 0$. The required precision for computed eigenpairs is 10^{-13}.

The inexact Krylov method can be easily implemented in RAPACK. The reason is that RAPACK uses reverse communication to obtain the results of matrix related operations, which gives user program the flexibility to control the quality of matrix-vector multiplication. In addition, RAPACK provides the residual information which can be used to set the current precision.

Here, the inexact Krylov method is implemented in the RA mode. The precision requirement θ for solving linear system, as defined in (2.4), is given as

$$\theta = \max\{\tau, \frac{\epsilon\tau}{m\|r\|}\}, \tag{4.2}$$

where τ is the desired precision of computed eigenpair, $\|r\|$ is the norm of current residual, m is the maximum dimension of the subspace, and $\epsilon < 1$ is a fudge factor. We take $\tau = 10^{-12}$, $m = 50$ and $\epsilon = 10^{-3}$. Hence

$$\theta = \max\{10^{-12}, \frac{10^{-3} \times 10^{-12}}{50\|r\|}\} = \max\{10^{-12}, \frac{2 \times 10^{-17}}{\|r\|}\}.$$

	Inexact method	Mode 3	SIRA mode
Etime	80	106	48
TMVM	5240	7083	2829
OUTER-ITER	43	50	89
AVG-INNER-ITER	122	142	32

Table 4.4: Inexact Krylov method, mode 3 of ARPACK, and mode SIRA of RAPACK.

Table 4.4 presents the experimental results. The inexact Krylov method works nicely. It converges to six smallest eigenvalues within 37 outer iterations, and the total number of matrix vector multiplication is 5240. Although it is not as good as the SIRA mode, it still outperforms the mode 3 of ARPACK. Therefore, if some fast matrix-vector multiplication methods, whose cost is proportional to required precision, are applicable, the inexact Krylov is definitely preferable.

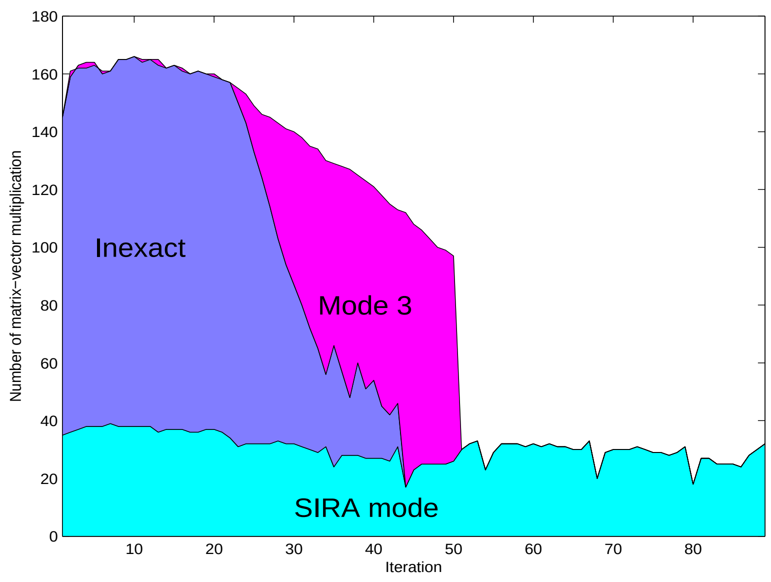

Figure 4.6: The precision requirements for the inexact Krylov method, the SIRA mode and the mode 3 of ARPACK.

Figure 4.6 exhibits the number of matrix-vector multiplications required by the inexact Krylov method, as well as those by the SIRA mode and by the mode 3 of ARPACK. In the beginning, the inexact Krylov method demands a similar number of matrix-vector multiplications as ARPACK. As the candidate approximation converges, the precision requirement is relaxed, and the cost of solving linear systems drops. Note this inexactness does not affect the convergence much. Figure 4.7 presents the projection errors of the six eigenvectors with the subspaces generated by the inexact Krylov method and by the mode 3. For an eigenvector x and an orthonormal basis U of a subspace, the projection error is $\|x - UU^*x\|$. In the figure, the solid lines are for the inexact Krylov method and the dotted lines are for mode 3. The approximations for both methods converge smoothly. The inexact Krylov method just uses less computation to achieve the same performance.

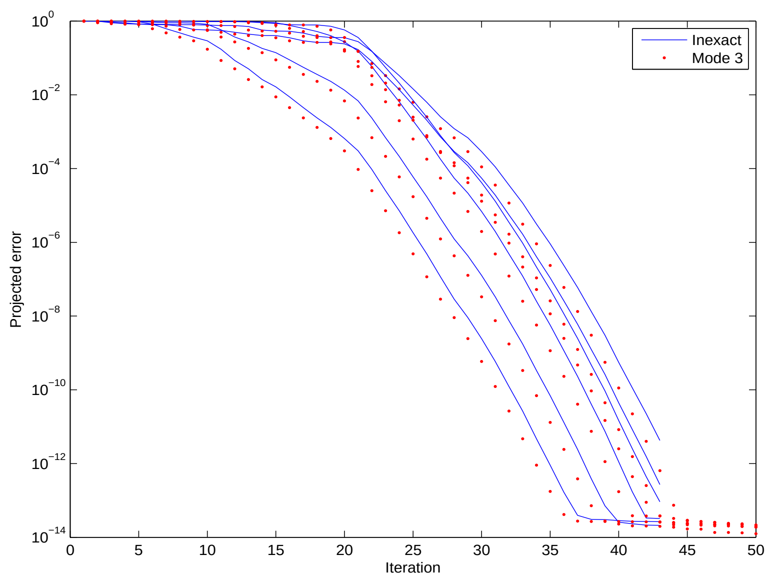

Figure 4.7: The convergence of the inexact Krylov method and the mode 3 of ARPACK.

4.5 Complex shift-invert for real eigenproblem

Occasionally, a complex shift is required in shift-invert enhancement for solving real eigenproblems. In this section, two methods that are able to fulfill this requirement are considered. Let σ be a complex shift value. The first method, proposed by Parlett and Saad [28], solves the linear system

$$(A - \sigma I)v = u,$$

where u is the latest Krylov vector, and uses the real part or the imaginary part of the solution v in subspace expansion. The second method is the SIRA mode of RAPACK, which solves the linear system

$$(A - \sigma I)v = r,$$

with the residual r as right hand side, and uses both real part and imaginary part of v in subspace expansion. Parlett and Saad's algorithm has been implemented in ARPACK as mode 3 and mode 4. Mode 3 uses the real part of the complex solution in subspace expansion, and mode 4 uses the imaginary part. Although they can be easily realized in RAPACK also [Chapter 3], the performance will not be as good as that in ARPACK. In this experiment, we will compare the mode 3, mode 4 of ARPACK with the SIRA mode of RAPACK.

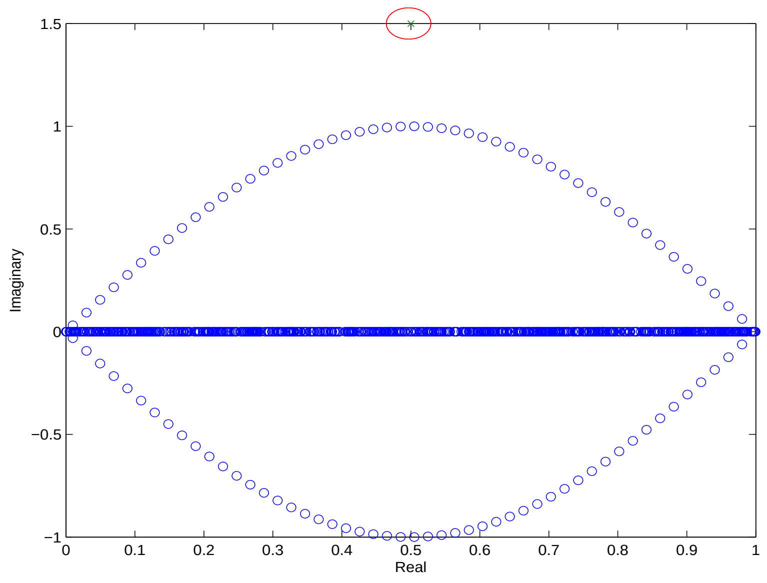

Figure 4.8: The spectrum of the problem.

The testing case used in the experiment is a 1000×1000 real nonsymmetric matrix whose first 100 eigenvalues are of the form $\lambda \pm i \sin(\lambda\pi)$. The value λ is sampled in $(0, 1)$, $\lambda_i = (2i - 1)/101$ for $i = 1, \cdots, 50$. The reset eigenvalues are real, randomly distributed in $(0, 1)$. Figure 4.8 shows the spectrum of the problem. The eigenvalues we are interested in are those near the point $0.5 + 1.0i$, which cannot be enhanced by any real shift. Therefore, a complex shift $0.5 + 1.5i$ is used, as marked the cross point in the figure.

The experimental settings are as follows. For each computation mode, the maximum dimension of subspace is 40. The desired precision of computed eigenpair is 10^{-11}. The complex GMRES algorithm will be used to solve the linear systems. For mode 3 and mode 4 of ARPACK, the linear systems are asked to be solved to the accuracy 10^{-11}; and for SIRA mode of RAPACK, it is 10^{-3}. The maximum dimension of the subspace used in GMRES, ARPACK and RAPACK is 40.

Table 4.5 compares the performance of these three methods. Basically, the mode 3 and mode 4 of ARPACK give similar results. Again, the SIRA

Methods	Mode 3	Mode 4	SIRA mode
TMVM	66003	66707	34870
OUTER-ITER	242	242	469
AVG-INNER-ITER	273	275	74

Table 4.5: Result of complex shift invert methods.

mode of RAPACK shows better performance, in which the total number of matrix-vector multiplications is only half of the other methods, although it takes more iterations to finish the task.

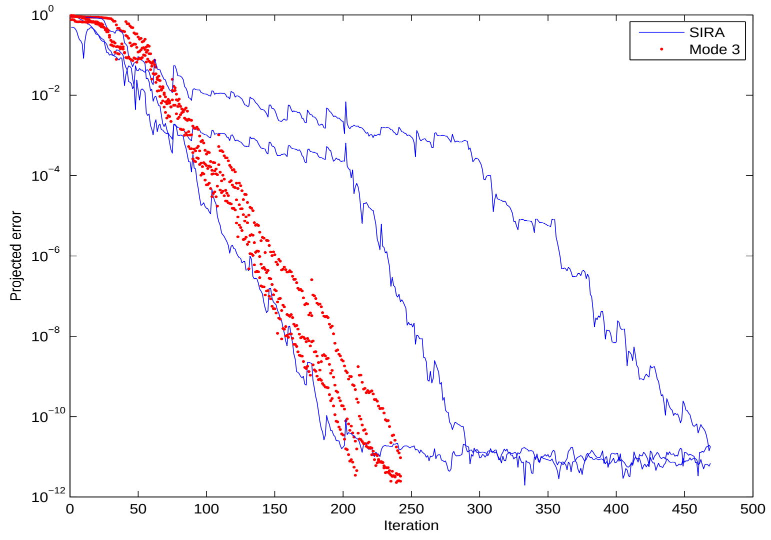

Figure 4.9: Convergence of mode 3 and SIRA mode.

Figure 4.9 compares the projection errors of mode 3 (dotted lines) and the SIRA mode (solid lines). It shows the first approximations for both methods have similar convergence. The second and the third approximations of mode 3 converges simultaneously with the first approximation, while those of the SIRA mode start their convergence after becoming candidates. Figure 4.10 displays the numbers of matrix-vector multiplications required by mode 3 and the SIRA mode for each iteration.

One thing should be mentioned is these three methods do not compute the same results. Table 4.6 lists computed eigenvalues for these three methods.

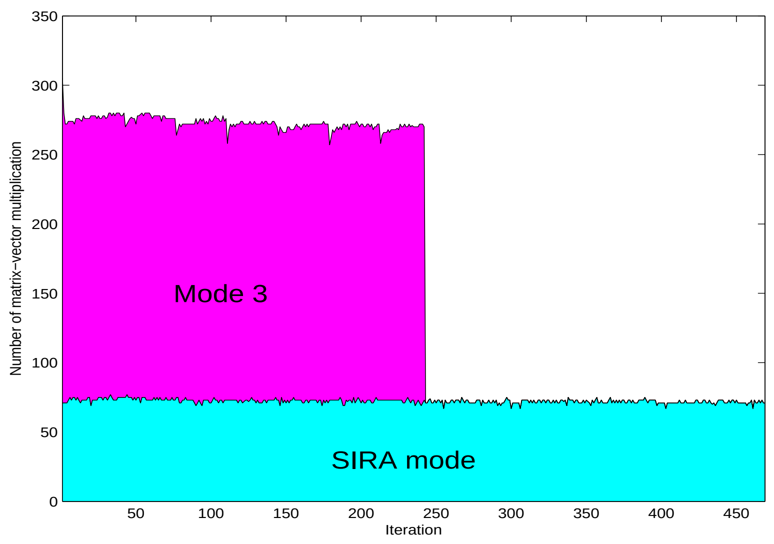

Figure 4.10: Number of matrix vector multiplications required by mode 3 and SIRA mode.

As it shows, only SIRA mode computes the correct answers, which are three eigenvalues closest to $0.5 + i$. Mode 3 and mode 4 only find partial solutions. This phenomenon requires further investigation.

4.5.1 Discussion

There are many tradeoffs when the SIRA mode is used. Although we usually specify 10^{-3} as the required precision for solving linear systems, it may not be the optimal value for all the cases. Sometimes, a smaller precision could reduce the number of outer iterations. But it also means more inner-iterations

Eigenvalue	Mode 3	Mode 4	SIRA mode
$0.465 \pm 0.994i$		$\surd$	
$0.485 \pm 0.999i$		$\surd$	$\surd$
$0.505 \pm 0.999i$	$\surd$	$\surd$	$\surd$
$0.525 \pm 0.997i$	$\surd$		$\surd$
$0.545 \pm 0.990i$	$\surd$		

Table 4.6: Computed eigenvalues of complex shift invert methods.

are required to solve linear systems. How to choose a proper precision for linear system solving so that the overall cost could be minimized is still a research problem.

Another tradeoff is the choice of shift. In the above case, the best shift is $0.5 + 1.0i$, for which both methods can compute six desired eigenvalues within 50 outer iterations. However, with such a close shift, the linear system is almost unsolvable by iterative methods like GMRES. In general, a shift close to desired targets can accelerate the outer process, but it is an obstacle for the inner procedure.

Chapter 5. Conclusions

This chapter provides a brief summary of the results in this thesis and proposes some possible directions for the future research. The summary is given in section 5.1, and the future works, including new algorithms and extensions of RAPACK, are presented in section 5.2.

5.1 Summary

As mentioned in Chapter 1, the residual Arnoldi method has some unique numerical properties that can be used not only to design more efficient algorithms but also help to have a better understanding of Krylov subspace methods. Here we enumerate some important properties.

1. The residual Arnoldi method generates a Krylov subspace when no errors are introduced.
2. When errors are introduced in the computation, the candidate approximation will converge, while the other eigenpair approximations stagnate.
3. The residual Arnoldi method can start with a non-Krylov subspace that contains eigenvector approximations.
4. The shift-inverted enhancement can be effectively applied to the residual Arnoldi method with low cost.

These properties are analyzed by the study of the residuals. We first derived the residual Arnoldi relation, which characterizes the subspace generated by the residual Arnoldi method with and without error. From this relation, we extracted the the backward error of the generated subspace. Although the backward error E_k itself may not be small, its product with the eigenvector $E_k x$ is converging along with the residuals. Using this property, we could establish the convergence theory of the residual Arnoldi method under some reasonable assumptions. A similar approach applied to the theory of the residual Arnoldi method with shift-invert enhancement.

The residual Arnoldi method has been implemented in RAPACK, a numerical package for solving real nonsymmetric eigenproblems. RAPACK contains two computational modes, RA and SIRA, which implement the residual Arnoldi method and the residual Arnoldi method with shift-invert enhancement respectively. RAPACK uses reverse communication to work with various matrix formats. It also implements the Krylov–Schur restarting algorithm to manage memory so that only moderate memory is required. The time complexity for the RA mode is $kO(nm^2) + kf(n)$, where n is the order of the original matrix, m is the maximum dimension of the generated subspace, k is the total number of iterations, and $f(n)$ is the time complexity for matrix-vector multiplication. The time complexity for the SIRA mode is similar, $kO(nm^2) + kf(n) + kg(n, \epsilon)$, where $g(n, \epsilon)$ is the time complexity for solving linear systems to the precision ϵ. The storage requirement for both modes are $n \times (2m + 11) + 3(m + 1) \times (m + 1)$.

The efficiency and versatility of RAPACK has been evaluated and tested through experiments. In the experiments, RAPACK was compared with two existing eigensolvers, ARPACK [22] and SRRIT [3]. RAPACK outperforms ARPACK significantly when shift-invert enhancement is applied, and achieves similar performance for other cases. The comparison with SRRIT is part of an experiment that reuses existing subspace to refine the solutions. The results indicate the superiority of using RAPACK when few eigenpairs are sought. In addition, we also implemented the inexact Krylov method in the RA mode of RAPACK to compare with the SIRA mode of RAPACK. Experimental result shows the SIRA mode is still faster than the inexact Krylov method, but the inexact Krylov method has its advantage when the matrix-vector products can be computed inexactly. The last category of experiments compares the algorithms that use complex shift in shift-invert enhancement in solving real nonsymmetric eigenproblems. The mode 3 and mode 4 of ARPACK is compared with the SIRA mode of RAPACK. Again, the SIRA mode shows its advantage of allowing inexactness in solving linear systems.

5.2 Ongoing and future research

5.2.1 The harmonic residual Arnoldi method and the refined residual Arnoldi method

Sometimes, the Rayleigh-Ritz method may fail to compute accurate eigenvector approximations [41], especially when the corresponding eigenvalues are interior and clustered. Several methods, such as the harmonic Ritz method [26] and the refined Ritz method [19], can improve the quality of the eigenvector approximations.

The harmonic Ritz method extracts the approximations from the eigenpairs of a general eigenvalue problem that protects the eigenvectors near a shift against impersonators. Let A be the matrix, U be the generated subspace, and κ be a shift value. The harmonic Ritz method uses the solution (δ, w) of the general eigenvalue problem

$$U^*(A-\kappa I)^*(A-\kappa I)Uw = \delta U^*(A-\kappa I)^*Uw, \tag{5.1}$$

to extract an eigenpair approximation, $(\kappa+\delta, Uw)$, of A.

The refined Ritz vector is computed from the following optimization process,

$$\min_{z\in\operatorname{span}\{U\},\|z\|=1} \|Az-\mu z\|, \tag{5.2}$$

where μ is an eigenvalue approximation. Computationally, the solution x has the form Uz where z is the right singular vector corresponding to the smallest singular value of the matrix $AU-\mu U$.

We test the combination of these two methods with the residual Arnoldi method through the following experiment. As before, we use a 100×100 matrix A which has eigenvalues $1, 0.95, \cdots, 0.95^{99}$ and randomly generated eigenvectors in the experiment. The 30th eigenvalue, 0.2259, is targeted for each method. The harmonic Ritz method uses 0.222 as the shift value. For comparison, the residual Arnoldi method is also included in the study. The trials for each method have two cases: one is without error and another with error 10^{-5} in each iteration. All experiments employ the same initial vector.

Figure 5.1 summarized the experimental results. The top two are for the residual Arnoldi method. The solid line is for the targeted eigenvalue 0.2259; and the dashed line is for the next one 0.2146. The convergence is measured by the projection error, $\|x-UU^*x\|$, in which x is the actual eigenvector and

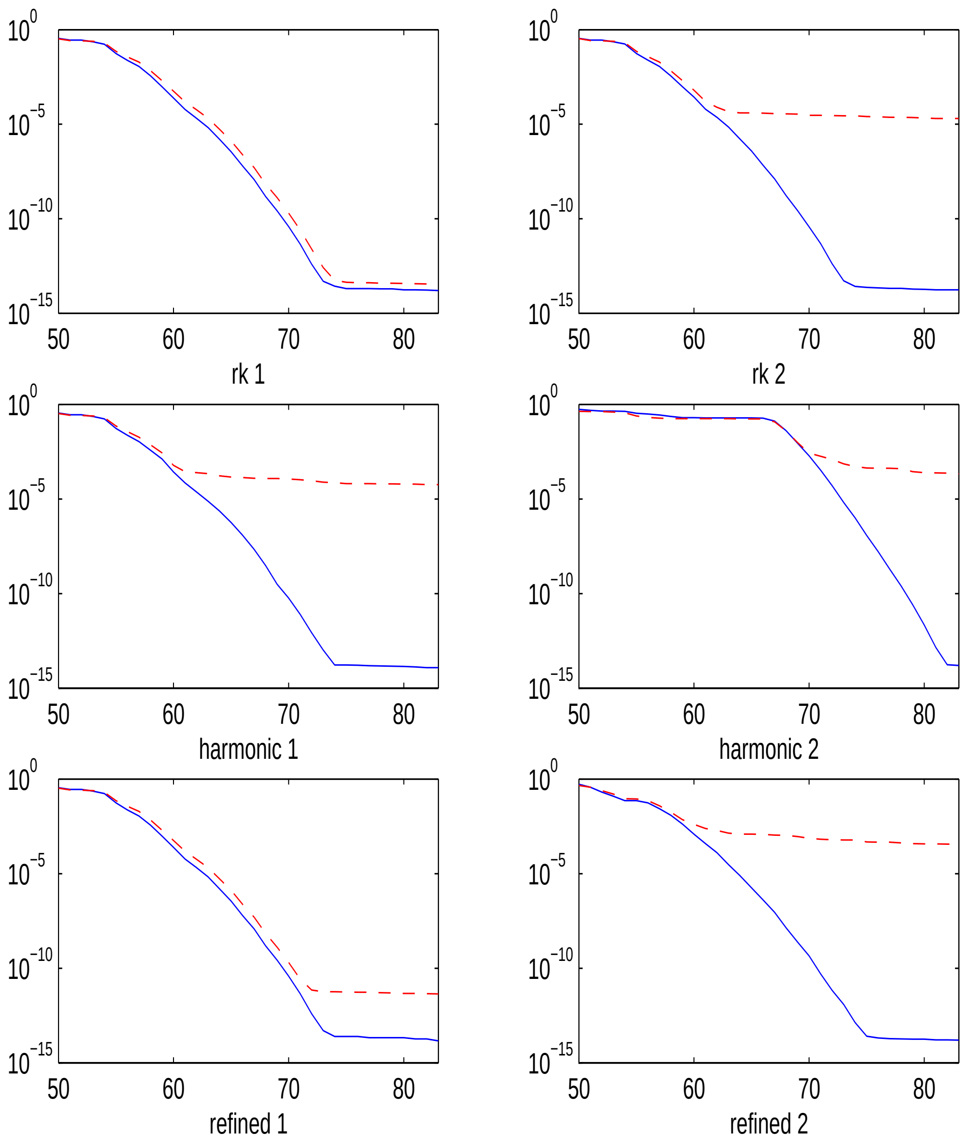

Figure 5.1: The residual Arnoldi method, the harmonic residual Krylov method and the refined residual Arnoldi method. The x-axis represent the number of iterations, and y-axis is for projected error.

U is the generated subspace matrix. The graph, labeled rk 1, is for the case without errors; and graph rk 2 is the result when errors are introduced.

Graphs in the middle display the results for the harmonic residual Krylov method, without and with error. Without errors (harmonic 1), the candidate approximation converges as the residual Arnoldi method, but other approximation levels off around 10^{-5}. This phenomenon is similar to the case when the shift-invert enhancement is applied to a Krylov sequence [41, page 305-206]. In the other case (harmonic 2), the convergence gets worse. Not only the second eigenpair stagnates, but the convergence of the targeted approximation is also postponed.

The experimental results for the refined residual Arnoldi method are shown in the bottom two graphs. Refined 1 demonstrates the convergence in the without error. As can be seen, the second eigenvector stagnates before the targeted approximation reaches the machine precision, although not seriously. In figure refined 2, as error interfered, both approximations converge as they do in the residual Krylov method. However, one can observes that the second eigenvector does not reach the same precision level as it does in figure rk 2.

These experimental results confirm the idea that the precise eigenvector approximations may not result a more desirable subspace when their residuals are used. The Rayleigh Ritz method may generate poor eigenvector approximations, but the residuals of them works well in the subspace expansion. As for the unexpected behaviors of the harmonic method and the refined method, more studies are needed for better understanding.

5.2.2 The block residual Arnoldi method

Many subspace methods can be generalized to the block versions by initialing and expanding the subspace with a block, which is a matrix of multiple columns. For example, the block Krylov subspace method [30] expands its subspace with vectors

$$b_1, b_2, \cdots, b_p, Ab_1, Ab_2, \cdots, Ab_p, \cdots, A^{k-1}b_1, A^{k-1}b_2, \cdots, A^{k-1}b_p, \cdots,$$

where p is the block size, and $[b_1, b_2, \cdots, b_p]$ is the initial block. Several advantages of this generalization have been established in literatures. For example, block methods usually have better convergence properties [30], and they improve the robustness of computing multiple or clustered eigenvalues.

The block residual Arnoldi method follows this extension naturally. In each iteration, in stead of selecting a candidate and computing its residual, the block method chooses p candidates and uses their residual in subspace expansion. The method is outlined in Algorithm 5.1.

Algorithm 5.1 The block residual Arnoldi method.

```
1. Let $U = [b_1, b_2, \cdots, b_p]$ be a $n \times p$ orthonormal matrix.

2. Compute the Rayleigh quotient $M = U^*AU$ and its eigendecomposition.

3. Select $p$ eigenpairs of $M$, $(\mu_1, y_1), (\mu_2, y_2), \cdots (\mu_p, y_p)$, and compute the
   residuals of their corresponding Ritz pair $r_i = AUy_i - \mu_i Uy_i$ for $i = 1 \cdots p$.

4. Expand $U$ by $r_1, r_2, \cdots, r_p$.

5. Go to step 2 until all desired eigenpairs converged.
```

We have tested this algorithm and compared it with the Arnoldi process and the residual Arnoldi method. The block size p is set to be 3. Matrix in problem is of order 100, with eigenvalue $1, 1, 0.95, \cdots, 0.95^{98}$ and randomly generated eigenvectors. Note that the first two eigenvalues are multiple. For each method, two versions of experiments are evaluated, one without error and another with error 10^{-3} in each iteration.

Figure 5.2 displays the experimental results. Each subplot has four lines, representing the convergence of four dominant eigenvectors. The convergence is measured by the angle between the actual eigenvectors and the subspace. When the eigenvector x is corresponding to a simple eigenvalue, the measurement is made by $\|(I - UU^*)x\|$, where U is the orthonormal basis of the subspace. For multiple eigenvalues, the eigenvectors can be any vectors in the subspace $\text{span}\{x_1, x_2, \cdots, x_i\}$, where $x_1, x_2, \cdots, x_i$ are the original eigenvectors. The sine of the angles of $\text{span}\{U\}$ and $\text{span}\{x_1, x_2, \cdots, x_i\}$ are the singular values of the the matrix $(I - UU^*)V$ [41], where V is the orthonormal basis of $\text{span}\{x_1, x_2, \cdots, x_i\}$.

The plot on the left top represents the convergence of the Arnoldi process without error. The solid line and the dashed line are for the dominant eigenvectors; and the dashed-dotted line and the dotted line are for the third and

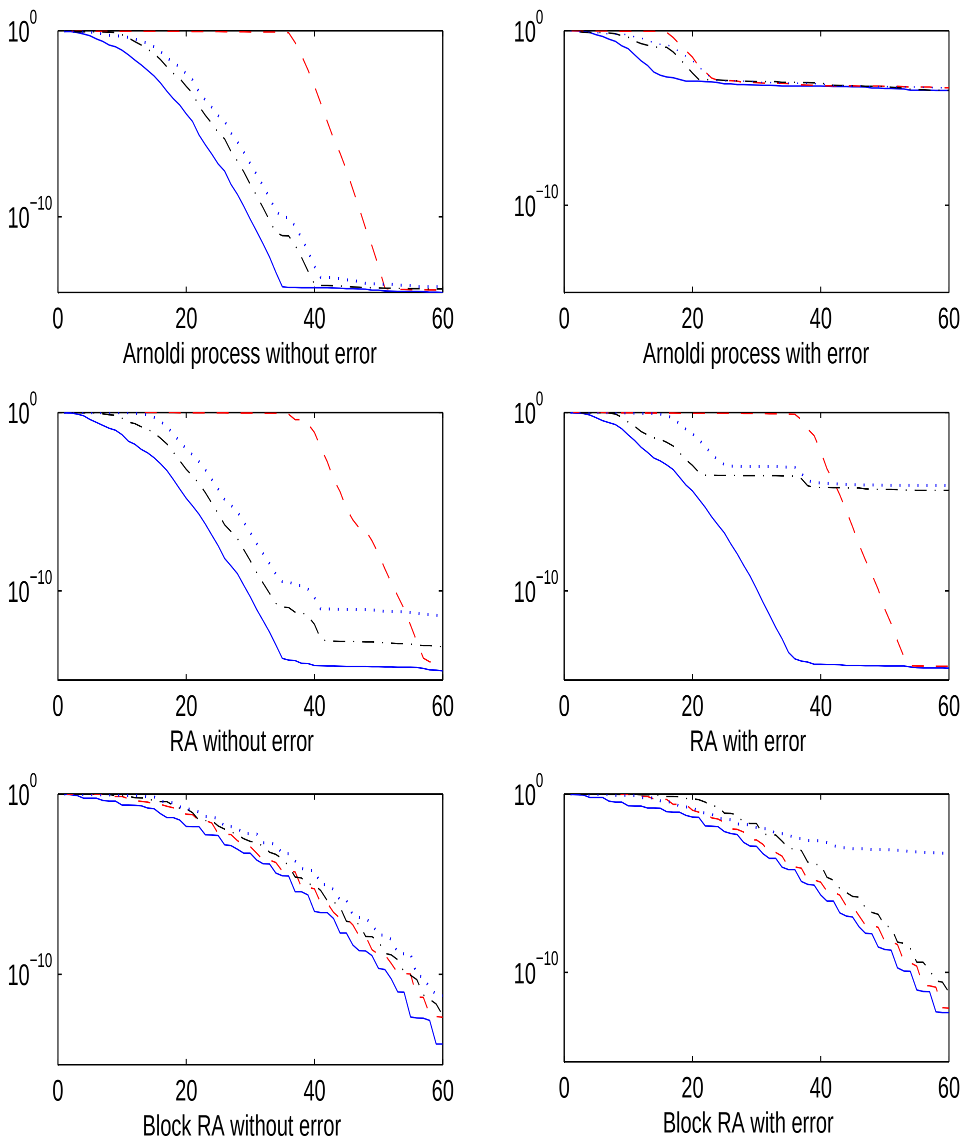

Figure 5.2: The block residual Arnoldi method. The x-axis represent the number of iterations, and y-axis is for projected error.

fourth eigenvectors. As can be seen, the second eigenvector, which also has eigenvalue 1, does not start its convergence until the dominant one reaches the machine precision. Plot on the right top is the case when error 10^{-3} are added to the subspace in each iteration. Unsurprisingly, all approximations stagnate at the level of 10^{-3}.

Plot in the middle are the results for the residual Arnoldi method without and with errors. In both cases, the dominant and subdominant eigenvector converge as they do in the Arnoldi process, and the remaining eigenvectors stagnate when errors are introduced.

Results of the block residual Arnoldi method are plotted in the bottom two graphs. Unlike the previous two methods, the convergence is slower. (The x axis is the dimension of the subspace, not the number of iterations.) However, the subdominant eigenvector starts its convergence with others. In addition, when errors are introduced, the three dominant eigenvector still converge, since their approximations are the candidates for the process (block size is 3), but fourth one stagnates.

5.2.3 Other research directions

Here are some related research directions.

1. **Inexact subspace methods for eigenvalue problems:** Recently study on eigenvalue problems is centered at algorithms that could tolerate errors, such as the Jacobi–Davidson method [35], the inexact Krylov subspace methods [8, 17, 34, 13] and inexact Rayleigh quotient iteration [15]. Our research on the perturbation theory of residual Arnoldi method could lead to better understanding of these algorithms and help the design of new ones.

2. **Extensions of RAPACK:** We propose to include suites for real symmetric and complex matrices in RAPACK. Later we may implement suites for generalized eigenvalue problems and the singular value decomposition of large matrices. The wrapper modules that provide easy usage are also considered, for example, the subroutine that user can call without reverse communication.

BIBLIOGRAPHY

[1] Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman. *Compilers: Principles, Techniques and Tools*. Addison-Wesley, 1986.

[2] Zhaojun Bai, James Demmel, Jack Dongarra, Axel Ruhe, and Henk van der Vorst. *Templates for the Solution of Algebraic Eigenvalue Problems*. SIAM, Philadelphia, 2000.

[3] Zhaojun Bai and G. W. Stewart. Algorithm 776: SRRIT: A Fortran subroutine to calculate the dominant invariant subspace of a nonsymmetric matrix. *ACM Transactions on Mathematical Software*, 23(4):494–513, 1997.

[4] R. Barrett, M. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. Van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition*. SIAM, Philadelphia, PA, 1994.

[5] C. Beattie, Mark Embree, and J. Rossi. Convergence of restarted Krylov subspaces to invariant subspaces. *SIAM Journal on Matrix Analysis and Applications*, 25(4):1074–1109, 2001.

[6] Christopher A. Beattie, Mark Embree, and D. C. Sorensen. Convergence of polynomial restart Krylov methods for eigenvalue computations. *SIAM Rev.*, 47(3):492–515, 2005.

[7] M. Bennani and T. Braconnier. Stopping criteria for eigensolvers. Technical report TR/PA/94/92, CERFACS-ERIN, 1994.

[8] A. Bouras and TITLE = V. Frayse. Technical report.

[9] Tim Davis. Research and software development in sparse matrix algorithms. http://www.cise.ufl.edu/research/sparse/.

[10] Howard C. Elman, David J. Silvester, and Andrew J. Wathen. *Finite Elements and Fast Iterative Solvers*. Oxford University Press, 2005.

[11] L. Elsner. An optimal bound for the special variation of two matrices. *Linear Algebra and Its applications*, 71:77–80, 1985.

[12] Mark Embree. The Tortoise and the Hare, Restart GMRES. *SIAM Review*, 45:259–266, 2003.

[13] Jasper Van Den Eshhof and Gerard L.G. Sleijpen. Inexact Krylov subspace methods for linear system. *SIAM Journal on Matrix Analysis and Applications*, 26(1):125–153, 2004.

[14] Diederik R. Fokkema, Gerard L. G. Sleijpen, and Henk A. Van der Vorst. Jacobi–Davidson style QR and QZ algorithms for the reduction of matrix pencils. *SIAM Journal on Scientific Computing*, 20(1):94–125, 1999.

[15] M. A. Freitag and A. Spence. Convergence of inexact inverse iteration with application to preconditioned iterative solves. *BIT Numerical Mathematics*, 43(1):1–18, 2003.

[16] G. H. Golub and C. F. Van Loan. *Matrix Computations.* John Hopkins University Press, Baltimore, MD, 3 edition, 1996.

[17] G. H. Golub and Q. Ye. Inexact inverse iterations for the generalized eigenvalue problems. *BIT*, pages 671–684, 2000.

[18] Gene H. Golub, Zhenyue Zhang, and Hongyuan Zha. Large sparse symmetric eigenvalue problems with homogeneous linear constraints: the Lanczos process with inner-outer iterations. *Linear Algebra and its Application*, 309(1-3), 2000.

[19] Zhongxiao Jia. Refined iterative algorithm based on Arnoldi's process for large unsymmetric eigenproblems. *Linear Algebra and its Application*, 1997.

[20] Zhongxiao Jia and G. W. Stewart. On the convergence of Ritz values, Ritz vectors, and refined Ritz vectors. *UMIACS TR-99-07 CMSC TR-3986*, 2001.

[21] Che Rung Lee and G. W. Stewart. Eigentest: A test matrix generator for large-scale eigenproblems. Tech report UMIACS TR-2006-07 CMSC TR-4783, University of Maryland, 2006.

[22] R. B. Lehoucq, D. C. Sorensen, and C. Yang. *ARPACK Users' Guide: Solution of Large Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods.*, 8 1997.

[23] Rich Lehoucq, Kristi Maschhoff, Danny Sorensen, and Chao Yang. *PARPACK*. http://www.caam.rice.edu/software/ARPACK/.

[24] Rich B. Lehoucq. Implicitly restarted Arnoldi methods and subspace iteration. *SIAM Journal on Matrix Analysis and Applications*, 23(2):551–562, 2001.

[25] K. Meerbergen and D. Roose. The restarted Arnoldi's method applied to iterative linear system solvers for calculation of rightmost eigenvalues. *SIAM Journal on Matrix Analysis and Applications*, 18:1–20, 1997.

[26] Ronald B. Morgan. On restarting the Arnoldi method for large nonsymmetric eigenvalue problems. *Mathematics of Computation*, 65(215):1213–1230, 1996.

[27] NIST. Matrix Market. math.nist.gov/MatrixMarket/.

[28] B.N. Parlett and Y. Saad. Complex shift and invert strategies for real matrices. *Linear Algebra and its Application*, 88/89:575–595, 1987.

[29] Y. Saad. *Iterative Methods for Sparse Linear Systems.* SIAM, Philadelphia, 2 edition.

[30] Y. Saad. On the convergence of the Lanczos and the block Lanczos methods. *SIAM Journal on Numerical Analysis*, pages 687–706, 1980.

[31] Y. Saad. *Numerical Methods for Large Eigenvalue Problems.* Manchester University Press, Oxford rd, Manchester, UK, 1992.

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

[33] Yousef Saad. *Iterative methods for sparse linear systems.* SIAM, 2nd edition, 2003.

[34] Valeria Simoncini and Daniel B. Szyld. Theory of inexact Krylov subspace methods and applications to scientific computing. *SIAM Journal of Scientific Computing*, 25(2):454–477, 2003.

[35] G. L. G. Sleijpen and H. A. van der Vorst. A Jacobi-Davidson iteration method for linear eigenvalue problems. *SIAM J. Matrix Anal. Appl.*, 17:401–425, 1996.

[36] D. C. Sorensen. Implicit application of polynomial filters in a k-step Arnoldi method. *SIAM Journal on Matrix Analysis and Applications*, 13(1):357–385, 1992.

[37] G. W. Stewart. SRRIT - a Fortran subroutine to calculate the dominant invariant subspaces of a real matrix. Tech Report TR-514, University of Maryland, College Park, 1978.

[38] G. W. Stewart. *Matrix Algorithms: Basic Decompositions*, volume 1. SIAM, Philadelphia, 1998.

[39] G. W. Stewart. A generalization of Saad's theorem on Rayleigh-Ritz approximations. *Linear Algebra and its Application*, (327):115–120, 2001.

[40] G. W. Stewart. A Krylov–Schur algorithm for large eigenproblems. *SIAM Journal on Matrix Analysis and Applications*, 23(3):601–614, 2001.

[41] G. W. Stewart. *Matrix Algorithms: Eigensystems*, volume 2. SIAM, Philadelphia, 2001.

[42] G. W. Stewart and Ji guang Sun. *Matrix Perturbation Theory.* Academic Press, 1990.

[43] X. Sun and N. P. Pitsianis. A matrix version of the fast multipole method. *SIAM Review*, 43(2):289–300, 2001.

[44] Ronald L. Rivest Thomas H. Cormen, Charles E. Leiserson and Clifford Stein. *Introduction to Algorithms.* MIT Press, 2 edition, September 2001.

[45] Antony F. Ware. Fast approximate Fourier transforms for irregularly spaced data. *SIAM Review*, 40(4):838–856, 1998.

[46] Yunkai Zhou and Yousef Saad. Block Krylov-Schur method for large symmetric eigenvalue problems. Research Report UMSI 2004/215, Supercomputing Institute, 2004.