Preconditioned Techniques For Large Eigenvalue Problems

Kesheng Wu

June, 1997

Abstract

This research focuses on finding a large number of eigenvalues and eigenvectors of a sparse symmetric or Hermitian matrix, for example, finding 1000 eigenpairs of a 100,000 $\times$ 100,000 matrix. These eigenvalue problems are challenging because the matrix size is too large for traditional QR based algorithms and the number of desired eigenpairs is too large for most common sparse eigenvalue algorithms. In this thesis, we approach this problem in two steps. First, we identify a sound preconditioned eigenvalue procedure for computing multiple eigenpairs. Second, we improve the basic algorithm through new preconditioning schemes and spectrum transformations.

Through careful analysis, we see that both the Arnoldi and Davidson methods have an appropriate structure for computing a large number of eigenpairs with preconditioning. We also study three variations of these two basic algorithms. Without preconditioning, these methods are mathematically equivalent but they differ in numerical stability and complexity. However, the Davidson method is much more successful when preconditioned. Despite its success, the preconditioning scheme in the Davidson method is seen as flawed because the preconditioner becomes ill-conditioned near convergence. After comparison with other methods, we find that the effectiveness of the Davidson method is due to its preconditioning step being an inexact Newton method. We proceed to explore other Newton methods for eigenvalue problems to develop preconditioning schemes without the same flaws. We found that the simplest and most effective preconditioner is to use the Conjugate Gradient method to approximately solve equations generated by the Newton methods. Also, a different strategy of enhancing the performance of the Davidson method is to alternate between the regular Davidson iteration and a polynomial method for eigenvalue problems. To use these polynomials, the user must decide which intervals of the spectrum the polynomial should suppress. We studied different schemes of selecting these intervals, and found that these hybrid methods with polynomials can be effective as well. Overall, the Davidson method with the CG preconditioner was the most successful method the eigenvalue problems we tested.

Chapter 1

Electronic Structure Simulation

Before entering into the main topic of this thesis, we use this chapter to introduces the context of our research which is also serve as motivation for our research. In this chapter, we focus on one application that is the main driving force behind this research in eigen-system solvers, the electronic structure simulation project at the University of Minnesota. We will give some background information about electronic structure simulation, and characteristics of the eigenvalue problems generated from the simulation. The eigenvalue algorithm developed in later chapters are designed to solve the eigenvalue problem of the same characteristics as these matrices.

1.1 Introduction

Many scientific and engineering applications give rise to eigenvalue problems, that is, they require the solution of the following equation

$$Ax = \lambda x,$$

where $A \in \mathbf{C}^{n\times n}$ is $n \times n$ matrix, $\lambda \in \mathbf{C}$ is an eigenvalue, and $x \in \mathbf{C}^n$ is an eigenvector [19, 51, 88, 97, 100, 95, 116, 154]. The particular eigenvalue problem we are interested in is generated from a materials science research project. The aim of the research project is to understand the dynamics of microscopic particles, specifically electronic structures of complex systems. Using quantum physics, it is possible to explain and predict certain material properties at a microscopic scale. A key step to the simulation of electronic structure is to find a steady state or a quasi-steady state. Intuitively, the process of finding this steady state is adjusting the electron distribution to minimize the total energy of the system. The total energy is a nonlinear function of the electron distribution. Finding the minimal energy and the corresponding electron distribution can be viewed as solving for the smallest eigenvalue and the corresponding eigenvector of a nonlinear eigenvalue problem. The Self-Consistent Field (SCF) iteration is the primary scheme of solving this nonlinear eigenvalue problem. At each step of the SCF iteration, a linear eigenvalue problem is generated. This is source of our matrix eigenvalue problem.

The Schrödinger equation is an non-linear eigenvalue problem. At each step of the SCF iteration, a matrix eigenvalue problem is generated, to contrast with the overall nonlinear eigenvalue problem, we will refer to this matrix eigenvalue problem as the linear eigenvalue problem in this chapter. However after this chapter, we will be concentrating on this linear eigenvalue problem, the "eigenvalue problem" will refer the linear eigenvalue problem.

In quantum physics, the electron distribution is represented by a wave-function. The governing equation is the Schrödinger equation,

$$\mathcal{H}\Psi = \mathcal{E}\Psi, \tag{1.1}$$

where $\mathcal{H}$ is the Hamiltonian operator for the system and $\mathcal{E}$ the total energy, Ψ is the wave-function [25, 63, 64]. The Hamiltonian operator describes the motion and interaction of the particles of the system. The wave-function Ψ describes where the particles are. The Schrödinger equation for any nontrivial system is complex nonlinear Partial Differential Equation (PDE). There are many numerical methods for solving a

PDE [61, 84, 128, 145]. One of the most common numerical approaches to solve the Schrödinger equation is to discretize it in a plane-wave basis [63] which is similar to spectral techniques for solving partial differential equations. This discretization scheme turns the Hamiltonian into a large dense matrix that is often too large to store in a computer's main memory. Usually the matrix is only used in the linear eigenvalue problem, one work-around to this difficulty is to use a Lanczos-type eigenvalue routine which only need to access the matrix through matrix-vector multiplications. Because the Fast Fourier Transformation (FFT) can be used to accomplish the most computation intensive operation in the matrix-vector multiplication, the matrix-vector multiplication is relatively inexpensive with this alternative.

With the plane-wave basis, the problem is solved in the Fourier Space. A different approach is to solve the problem in real space by discretizing the Schrödinger equation with the finite difference scheme [42, 128]. For localized systems, such as a cluster of atoms, high-order finite difference scheme has shown to be more efficient than plane-wave techniques in finding solutions of same accuracy [21, 22, 58, 59]. The matrices resulting from both finite difference methods and plane-wave techniques are large if the number of particles in the system is large. In addition, the size of the matrix is also affected by the desired accuracy of the solution, characteristics of the atoms involved, and the physical quantities to be computed. For complex systems involving hundreds of atoms, the matrix size could be on the order of millions.

Complex systems may also require one to solve many more linear eigenvalue problems before an acceptable solution for the nonlinear system is reached. If we solve the Schrödinger equation (1.1) directly only the smallest eigenvalue is needed. However, if we try to do so, the potential function V would be too complex to compute. In next section we will show a scheme of simplifying this V. This scheme makes V tractable, at the same time it requires computation of a large number of eigenvalues from the linear eigenvalue problem. The number of eigenvalues required is proportional to the number of atoms in the system. Most of the eigenvalue methods traditionally used for this computation are not very effective for finding a large number of eigenvalues. This is an additional challenge for an effective simulation.

1.2 *Ab Initio* pseudopotential simulation

The electronic structure of a condensed matter system, e.g., cluster, liquid or solid, is described by a quantum wave-function Ψ which can be obtained by solving the Schrödinger equation. This equation is very complex, because the Hamiltonian operator $\mathcal{H}$ describes motions of all particles in the system and the interactions among all of them. Complete analytical solution is only possible for the simplest atoms. Significant simplification is required to compute any large system. Most theories of condensed matter systems make the following three fundamental approximations to make them manageable.

Born-Oppenheimer approximation Born-Oppenheimer approximation neglects the kinetic energy of the nuclei. This is a good approximation because of two main reasons. First, the mass of nuclei is much larger than the mass of electrons in the system, typically more 1000 times larger, nuclei move very slowly compared to electrons. Second, we are not interested in the average motion of the whole system, in other word, we will solve the system in the nuclei frame of reference. Because of this approximation, the wave function we use only involves the electrons. Thus, the original electron-nuclear problem now becomes a pure electron problem. Under this circumstance, the wave-function describes the distribution of electrons only. Let $\mathbf{r}$ denote a point in space, the density of electron distribution at $\mathbf{r}$ is defined by

$$\rho(\mathbf{r}) = \Psi(\mathbf{r})^H \Psi(\mathbf{r}),$$

where the superscript H denotes complex conjugate.

Local density approximation To explain the concept of Local Density Approximation (LDA), we will first describe a more general theory, the density functional theory. The density functional theory transforms a many-electron problem into an one-electron problem. The simplified Schrödinger equation (1.1) in this case is called the Kohn-Sham equation,

$$\left\{-\frac{1}{2}\nabla^2 + V_{tot}[\mathbf{r};\rho]\right\}\psi_i(\mathbf{r}) = \epsilon_i\psi_i(\mathbf{r}) \tag{1.2}$$

where ∇^2 is the usual Laplacian operator, $V_{tot}[\mathbf{r};\rho]$ denotes the total potential experienced by an electron at location $\mathbf{r}$, ϵ_i is the energy of the ith state, and ψ_i is the corresponding one-electron wave-function. For our purposes, we can simply take (ϵ_i, ψ_i) as the ith smallest eigenvalue and the corresponding eigenvector. Because of the Pauli exclusion principle, each state can be occupied by two electrons at most. The electrons will fill the state with the lowest energy if the temperature is at absolute zero, about $-273.15°C$. In this case, if the number of electrons is q, we need to find $q/2$ eigenvalues. If the temperature is higher than absolute zero, or if there there are degenerate states, i.e., eigenvalues are equal to each other, some states may be partially occupied, then more eigenvalues needs to be computed. In terms of the one-electron wave-function, the electron distribution is defined as

$$\rho(\mathbf{r}) = \sum_{\text{occupied states}} |\psi_i(\mathbf{r})|^2. \tag{1.3}$$

The total potential, V_{tot}, is the sum of three terms, (1) the potential due to all nuclei and the core electrons, V_{ion}, (2) the Coulomb potential from all other valence electrons, also known as the Hartree-Fork potential, V_H, and (3) the exchange-correlation potential, V_{xc},

$$V_{tot}[\mathbf{r};\rho(\mathbf{r})] = V_{ion} + V_H + V_{xc}. \tag{1.4}$$

The *Local Density Approximation* (LDA) further simplifies the computation of the exchange correlation potential, V_{xc}. It states that if ρ changes slowly with respect to $\mathbf{r}$, the exchange-correlation potential can be approximated by assuming the electron density ρ to be locally uniform. The potential V_{xc} is obtained by Monte Carlo simulations in this simulation [25, 63].

Pseudopotential approximation Because most core electrons do not change their state during normal circumstance, thus it is possible to treat a nuclear and the surrounding core electrons as one entity, i.e., an ion. This is the basis of the pseudopotential approximation. The core of this approximation is to replace the complex potential function of this ion with a smoothly varying potential without changing its effect on the chemically active valence electrons. This approximation impacts our computation in two ways. First, it reduces the number of electrons that needs to be treated as unknowns, i.e., reduce the number of eigenpairs needed. Second, it simplifies the electronic potential due to ions, V_{ion}. Much research has been devoted to the construction of pseudopotential for different atoms. In our simulations, we use the potential developed in [148] for the silicon atoms.

For convenience, the ionic potential is broken into two parts, the local term and the non-local term [22]:

$$V_{ion} = \sum_a V_{loc}(|r_a|)\psi + \sum_{a,\ lm} \frac{\Delta V_l(\mathbf{r}_a)}{<\Delta V^a_{lm}>} u_{lm}(\mathbf{r}_a) \int u_{lm}(\mathbf{r}_a)\Delta V_l(\mathbf{r}_a)\psi d^3r, \tag{1.5}$$

where $<\Delta V^a_{lm}>$ is the normalization factor,

$$<\Delta V^a_{lm}> = \int u_{lm}(\mathbf{r}_a)\Delta V_l(\mathbf{r}_a)u_{lm}(\mathbf{r}_a)d^3r.$$

Here, the superscript a denotes an atom in position $\mathbf{R}_a$, $\mathbf{r}_a = \mathbf{r} - \mathbf{R}_a$, u_{lm} is the atomic pseudopotential wave-function associated with angular momentum quantum number $(l,\ m)$, $V_l(\mathbf{r})$ is an l-dependent ionic pseudopotential generated from u_{lm}, $\Delta V_l(\mathbf{r}) = V_l(\mathbf{r}) - V_{loc}(\mathbf{r})$ is the difference between the l component of the ionic pseudopotential and the local ionic potential. If we set $U_{alm}(\mathbf{r}_a) = \Delta V_l(\mathbf{r}_a)u_{lm}(\mathbf{r}_a)$ and consider the discretization of the equation (1.5) on a uniform grid, the non-local term becomes a sum of rank-one updates:

$$V_{ion} = \sum_a V_{loc}\psi + \sum_{a,l,m} t_{alm} U_{a,l,m} U^T_{a,l,m}\psi, \tag{1.6}$$

where t_{alm} are normalization coefficients and all other variables are the discretized matrix and vector forms of the operators and functions.

The Kohn-Sham equation is a nonlinear equation. A part of its nonlinearity comes from the Hartree-Fork potential V_H, which depends on the charge density $\rho(\mathbf{r})$. The charge density function in turn depends on

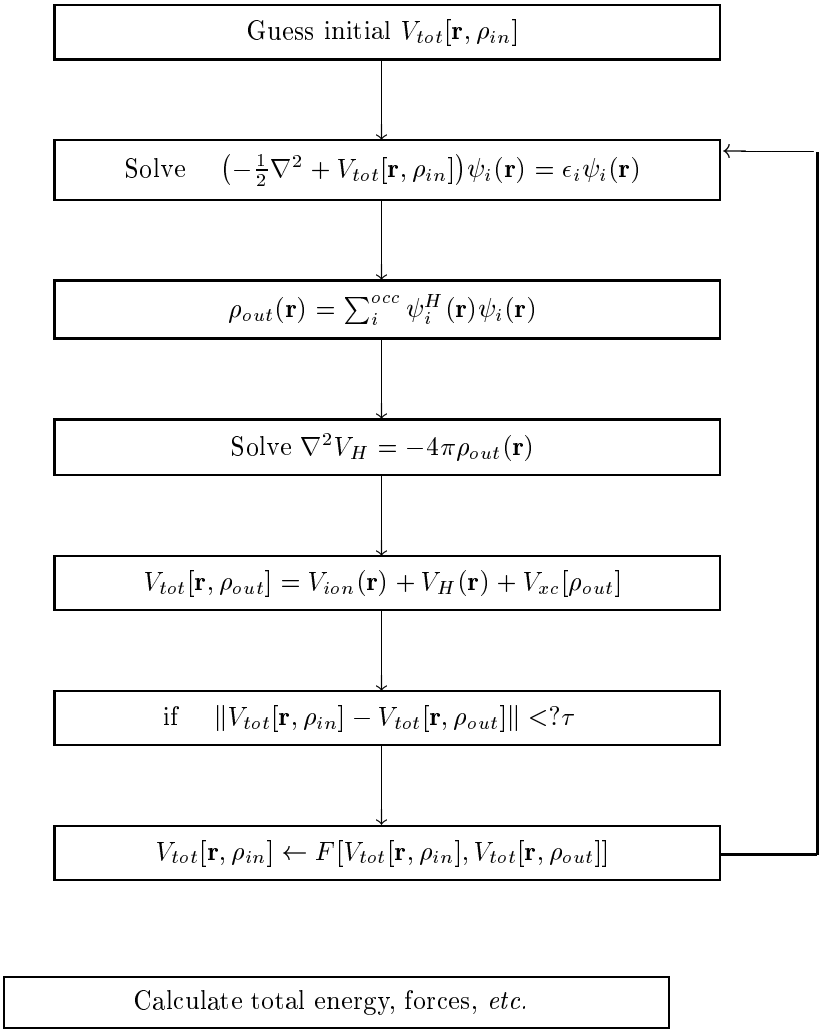

Figure 1.1: Flowchart of the self-consistent iteration for Kohn-Sham equation.

the solution of the Kohn-Sham equation via equation (1.3). Once the charge density $\rho(\mathbf{r})$ is known, the Hartree-Fork potential can be found by solving the Poisson equation:

$$\nabla^2 V_H = -4\pi\rho(\mathbf{r}). \tag{1.7}$$

The discrete form of this equation is easily solved by either the Conjugate Gradient method [119], or by a fast Poisson solver [49, 85]. Both potential V_H and V_{xc} have a local character and are represented by diagonal matrices when discretized using finite difference scheme on a uniform grid.

Now that we have shown how to compute most of the quantities of the Kohn-Sham equation, we can describe the Self Consistent Field (SCF) iteration in more detail. The SCF iteration can be viewed as an inexact Newton method for accelerating the fixed-point iteration $V_{tot}^{new} = \mathcal{M}(V_{tot})$, in which $\mathcal{M}(V_{tot})$ represents the process of computing a new potential from the old potential. A flowchart of the SCF iteration is depicted in figure 1.1. The iteration starts by solving a linear eigenvalue problem based on the current approximate potential V_{tot}. This gives a new series of eigenpairs (ϵ_i, ψ_i). The eigenvalues are used to determine which states are occupied and the eigenvectors of the occupied states are used to compute the new electron density $\rho_{out}(\mathbf{r})$. This new electron density is in turn used to find a new Hartree-Fork potential V_H and a new exchange correlation potential V_{xc}. If the difference between the newly computed total potential and the previous one is larger than than the prescribed tolerance τ, a new initial guess for V_{tot} is computed through extrapolation and the iteration is repeated. At the end of the self-consistent iteration, a

	Matrix size	Eigenvalues
$Si_{47}H_{60}$	29279	162
$Si_{71}H_{84}$	37121	262
$Si_{87}H_{76}$	41923	275
$Si_{99}H_{100}$	50181	318
$Si_{147}H_{100}$	59757	420
$Si_{275}H_{172}$	98745	780

Table 1.1: Matrix size of an simulation series.

Order of Accuracy	α_0	α_1	α_2	α_3	α_4	α_5	α_6
2	-2	1					
4	-5/2	4/3	-1/12				
6	-49/18	3/2	-3/20	1/90			
8	-205/72	8/5	-1/5	8/315	-1/560		
10	-5269/1800	5/3	-5/21	5/126	-5/1008	1/3150	
12	-5369/1800	12/7	-15/56	10/189	-1/112	2/1925	-1/16632

Table 1.2: Weights for computing second order derivative on uniform grid.

steady state is reached, i.e., a local energy minimum for the current configuration of nuclei is found.

If the system is time-dependent, e.g., the nuclei are in motion, the problem is discretized in time first to produce a series of quasi-static states which can be solved using the same process shown in figure 1.1.

1.3 Characteristics of the eigenvalue problems

Having simplified the many-body Schrödinger system into an one-electron problem, now we proceed to discretize the Kohn-Sham equation, equation (1.2), into a matrix eigenvalue problems. Traditionally, the Kohn-Sham equation is discretized in Fourier space [25, 63, 147]. In the past few years we have started using high-order finite difference schemes [20, 21, 22, 58, 59]. A typical problem solved in our simulations might consist of a few hundreds of atoms, using 12th order finite difference of the Kohn-Sham equation on a $60 \times 60 \times 60$ uniform grid. Table 1.1 shows the matrix sizes and the numbers of eigenvalues computed for a series of Silicon cluster simulations. Because the simulation software excludes the region where the electron density is negligible, the matrix size is usually less than the numbers of the total grid points for the problems solved.

In matrix form, the Hamiltonian is the sum of a Laplacian matrix, three diagonal matrices and a matrix representing the nonlocal contributions. Two of three diagonal matrices come from discretizing V_{xc} and V_H. The third one is due to the local part of the ionic potential V_{ion}. The nonlocal matrix is from the nonlocal ionic potential which can be expressed as sum of a sequence of simple rank-one updates over all atoms and quantum numbers, see equation (1.6). Typically, a small number of steps is required for the SCF iteration to converge. The matrix is symmetric for simple isolated systems, it can be complex Hermitian when any periodicity is presented in the system, e.g., crystal, or surface of crystal.

The finite difference scheme used is developed by Fornberg and colleagues [42, 43, 44]. It is used to discretize the Laplacian operator in equation (1.2) on a uniform grid. In one-dimensional case, the second-order derivative can be expressed as

$$\left.\frac{\mathrm{d}^2 u}{\mathrm{d}x^2}\right|_i = \frac{1}{h^2}\left(\alpha_0 u_i + \sum_{j=1} \alpha_j (u_{i-j} + u_{i+j})\right),$$

where h is the distance between grid points, u is an arbitrary function. The coefficients α_i are shown in

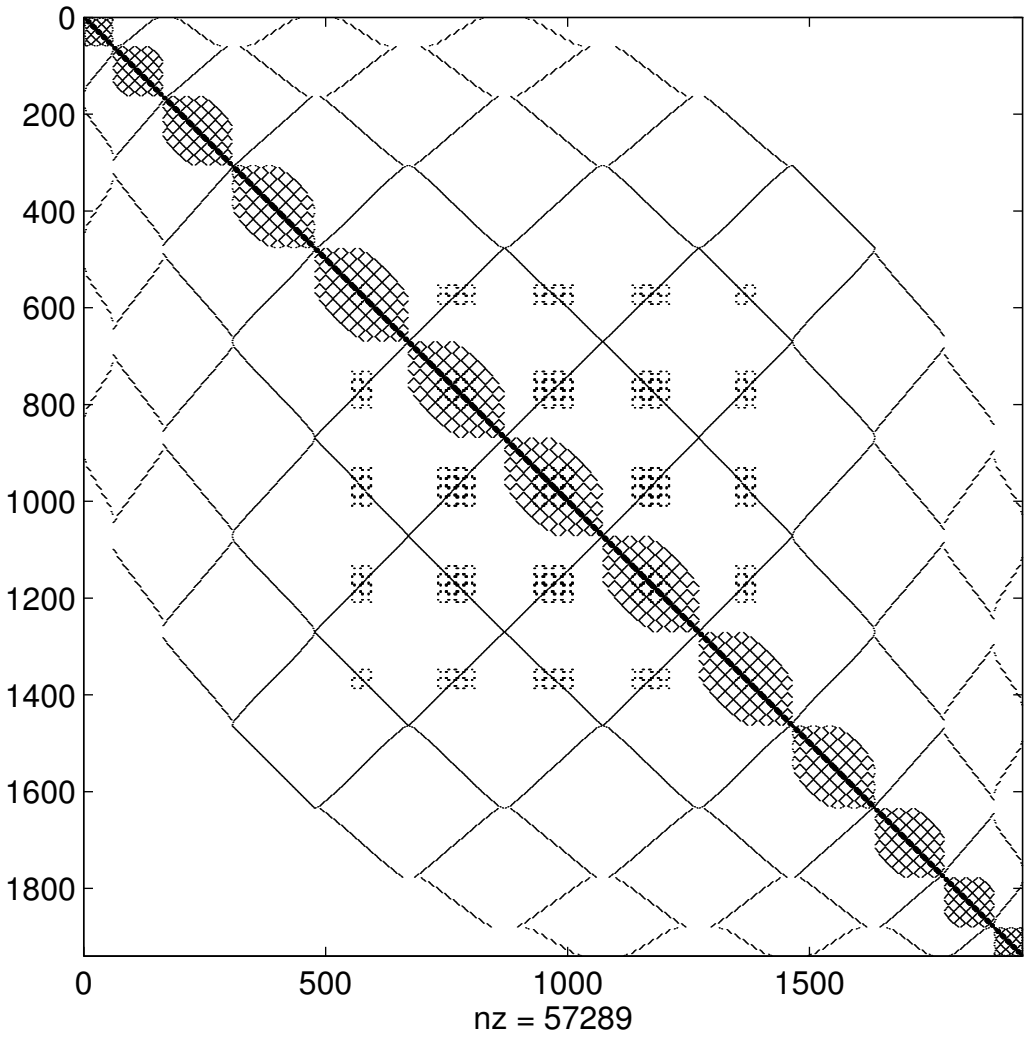

Figure 1.2: The nonzero pattern of a matrix for Si_2 cluster.

table 1.2 [21, 42]. Depending on the accuracy desired, we can choose the number of neighbors to involved in the above equation. To extend to 3-D case, we simply approximate the derivative of each direction separately. For a case where 12th order accuracy is needed, 36 neighboring nodes are used to form the discrete form of the Laplacian operator, 6 from each of $\pm x$, $\pm y$ and $\pm z$ directions.

In most test cases, we use a slightly modified form of the natural ordering for the grid points. This ordering scheme differs from the natural ordering in that it alternates between counting along $+x$ direction and $-x$ direction. It reduces the chance of memory bank conflict on some computers because it creates more variances in the memory access patterns of operations such as matrix-vector multiplication.

In the finite-difference matrix, the discrete form of the Laplacian operator dominate the non-zero pattern. If the 12th order centered finite difference is used, about 37 nonzero elements are generated for each row of the matrix. The non-local interaction generated from V_{ion} involves about 100 or so grid points around a silicon atom. This adds about 100 nonzero elements for selected rows of the matrix. Our finite difference scheme produces Hermitian or symmetric matrices. Figure 1.2 shows the nonzero pattern of a matrix for a two-silicon cluster. This matrix plotted in figure 1.2 is only 1939×1939. The net-like pattern is due to the modification to the natural ordering. In the center of the graph, the non-local interactions appears as two squares. The grid points with same z coordinate are number closer to each other which form a structure that is similar to the overall structure of the matrix. The size of these smaller structure decrease away from the

center of the matrix because more grid points are unused on the plan with same z value due to the exclusion of low electronic density regions.

A linear eigenvalue problem is produced at each self-consistent iteration. The matrix for the eigenvalue problem can be attributed to four sources, the Laplacian operator, V_{ion}, V_H, and V_{xc}. From one SCF iteration to the next one, the Laplacian part of the matrix will not change as long as the grid size is the same. The matrix generated from V_{ion} does not change if ions do not move. Using the centered finite difference discretization, the potential V_H and V_{xc} are transformed into two diagonal matrices. At each self-consistent iteration, V_H and V_{xc} varies with ρ, i.e., ψ_i. Thus, the difference between matrices from two successive iterations is only on the diagonal of the matrices. As the nonlinear problem converges, we expect the change in V_H and V_{xc} to decrease. Therefore the solution of previous linear eigenvalue problem will be a good initial guess for the next eigenvalue problem.

To reduce storage, the Laplacian part of the matrices is stored in stencil form, i.e., one row of coefficients in table 1.2 is stored as the matrix elements. A 2-D array `ja` is used to maintain the index information of the matrix where the `j`th neighbor of `i`th grid point is `ja(i,j)`. The potential is broken into components and stored in two parts. All three diagonal matrices are summed together in one vector. The nonlocal potential is stored as a series of short vectors to be used in the rank-one updates.

In summary, the eigenvalue problems we would like to address here are large, sparse and Hermitian. For realistic applications, a large number of eigenvalues and eigenvectors are desired. Good initial guess are available in most cases.

Chapter 2

Krylov Eigenvalue Solvers

This chapter explores the similarities and differences between the Arnoldi method and the Davidson method. Based on their similarities, a generic eigenvalue solver is extracted. Several different variations of this generic algorithm are studied by varying the input vector given to the preconditioner. Without preconditioning, these variants are mathematically equivalent to each other. However in practice, there are quite different due to difference in numerical properties. With preconditioning they behavior drastically different. Through this study, we will identify a numerically stable variation that works well with simple preconditioners.

2.1 Introduction

An eigenvalue λ and an eigenvector x of a matrix A is defined by the following equation

$$Ax = \lambda x.$$

The eigenvalue problems of interest here have Hermitian matrices whose eigenvalues are always real and their corresponding eigenvectors always exist. In addition, these matrices are also sparse which means there are only a small number of nonzero entries compared to total entries in the matrix. For this type of matrices, they are usually not stored in a 2-D $n \times n$ array, where n is the matrix size. Thus, they only require a fraction of the storage compared to a full matrix of the same size. The trade-off is that accessing a matrix element is often not as fast or as convenient as a matrix stored in a 2-D array.

In this thesis, we will address only symmetric eigenvalue problem since it is straightforward to extend everything discussed to Hermitian case. There are many ways of extracting eigenvalues and eigenvectors from a symmetric matrix [19, 27, 51, 95, 116, 154]. In this section we review the basic techniques for large sparse symmetric eigenvalue problems to identify a suitable starting point for our study.

Eigenvalue solvers may come very different forms, however most of them can be either classified as a QR-type method or a projection method. QR-type methods are very robust in finding eigenvalues and corresponding eigenvectors [51, 153, 154] and mature implementations of QR algorithm are widely available [4, 52, 127]. They generally require $O(n^2)$ storage and $O(n^3)$ operations. If the matrix size is relatively small, say less than a few hundreds, and all eigenpairs are needed, a QR based method is the best choice for the task. The entire set of n eigenvalues is referred to as the spectrum of the matrix. In most practical eigenvalue problems, the matrix size is large, and only part of the spectrum is needed, a QR method is usually not an effective choice.

Contrast to a QR method, projection methods are designed to extracts part of the spectrum effectively. Often, it only needs to access the matrix A through matrix-vector multiplication. For sparse matrices where the matrix is not stored in the usual $n \times n$ array, only need to perform matrix-vector multiplication with the matrix could significantly reduce the complexity of the eigenvalue program. A projection method for eigenvalue problems has two major components, one to generate a basis, the other to generate approximate solution by projecting the original eigenvalue problem into the basis. Most of variations on different eigenvalue solvers are in the basis generation part which will be reviewed in next section. Here we will focus on the projection techniques to generate approximate eigenvalues and eigenvectors.

There are two common methods to find eigenvalue approximations on a given basis, they are the Rayleigh-Ritz projection [51, 154] and the harmonic Ritz projection [78]. Given a m-dimensional basis $V \in \mathbf{R}^{n \times m}$, both projection techniques seek approximate solutions of the form

$$\begin{cases} x &= Vy, \\ \lambda &= x^T Ax / x^T x. \end{cases} \tag{2.1}$$

where λ is an approximate eigenvalue, x is an approximate eigenvector, and y is chosen by the two projection techniques according to their criteria. An eigenvalue and an eigenvector together is often referred to as an eigenpair. To reduce the number of symbols used, the exact value of a quantity will use the same symbol as the approximate value. The difference is that a superscript '$*$' will be attached to the exact one, for example (λ^*, x^*) denotes an exact eigenpair. The approximate eigenvalue in the above equation is defined as a ratio known as the Rayleigh quotient. An eigenvalue approximation may not be a Rayleigh quotient. However, due optimality of Rayleigh quotient [95], it is often the favorite choice. The eigenvalue and eigenvector approximation computed from the Rayleigh-Ritz projection are also known as the Ritz value and the Ritz vector.

Usually V is an orthonormal basis. In which case, $\lambda = x^T Ax = y^T V_m^T A V_m y = y^T H_m y$. This means that the Ritz values are the eigenvalues of $H_m \equiv V_m^T A V_m$. Because the size of H is usually small, a QR method can efficiently find its eigenvalues and eigenvectors. In addition, to get a normalized x, we only need to choose a normalized y. Since y is much smaller in size than x, it is a lot easier to normalize y. These are good motivations to maintain V_m orthonormal. However in general, the basis V does not need to be orthonormal. The residual of the above approximation is

$$r \equiv Ax - \lambda x = AVy - \lambda Vy.$$

The Rayleigh-Ritz technique projects the original eigenvalue problem $Ax = \lambda x$ by making the residual of the approximate solution orthogonal to the basis V, i.e.,

$$V^T (AVy - \lambda Vy) = 0. \tag{2.2}$$

Let $H = V^T AV$, and V be an orthonormal basis, i.e., $V^T V \equiv I$, then λ and y can be found by solving the following projected eigenvalue problem

$$Hy = \lambda y. \tag{2.3}$$

This projected eigenvalue problem is usually solved by using a QR method from LAPACK [4] or EISPACK [52, 127]. An eigenvector y of equation (2.3) is chosen to generate an approximation to the original eigenvalue problem based on equation (2.1). For example, if the smallest eigenvalue of A is sought, the eigenvector corresponding to the smallest eigenvalue of H is taken. In this case, the eigenvalue approximation λ can be easily computed. It is simply the eigenvalue of H corresponding to the chosen y. This is another advantage of using orthonormal basis. The following algorithm describes a version of the Rayleigh-Ritz procedure we will use later.

ALGORITHM **2.1 Rayleigh-Ritz projection** *to find an approximate solution to the smallest eigenvalue λ and corresponding eigenvector x.*

Given an orthonormal basis V,

1. *$H = V^T AV$,*
2. *find the smallest eigenvalue of H and corresponding eigenvector y,*
3. *the approximate solution to the smallest eigenvalue of A is the smallest eigenvalue of H, the corresponding Ritz vector is $x = Vy$. The residual of the approximation is $r = Ax - \lambda x = AVy - \lambda Vy$.*

The harmonic Ritz projection is a relatively new technique which follows the following orthogonality condition on the residual [78, 91]

$$(AV)^T (AVy - \lambda Vy) = 0. \tag{2.4}$$

By defining $W \equiv AV$, this equation can be rewritten as

$$WA^{-1}Wy = \frac{1}{\lambda}W^TWy,$$

which represents a Rayleigh-Ritz projection of A^{-1} on the basis W. To find a vector y for equation (2.1), the following projected eigenvalue problem is solved,

$$Hy = \eta Gy \tag{2.5}$$

where $H = V^TAV$ and $G = (AV)^T(AV)$. Usually, the matrix A is also shifted to favor the eigenvalue sought. For example, if the eigenvalue wanted is close to λ, the above definition of H and G are redefined as follows,

$$H = V^T(A - \lambda I)V \qquad G = V^T(A - \lambda I)^T(A - \lambda I)V.$$

The vector y needed for equation (2.1) is chosen from the eigenvectors of the above generalized eigenvalue problem. Once y is chosen, we can compute the approximation to the original eigenvalue problem using equation (2.1).

The harmonic Ritz projection algorithm shown below is one of the schemes shown in [78]. There are different realization of the harmonic Ritz projection, the one chosen here has orthonormal basis V which make it very easy for it to share the same basis generation procedures with the Rayleigh-Ritz projection algorithm described before.

ALGORITHM **2.2 Harmonic Ritz projection** *for finding an approximation of the smallest eigenvalue and corresponding eigenvector of a symmetric matrix A.*

Given an orthonormal basis $V \in \mathbf{R}^{n\times m}$, $W \equiv AV$,

1. $H = W^TV, \quad G = W^TW$.
2. *Solve the generalized eigenvalue problem $HY = LGY$ to find $Y, L \in \mathbf{R}^{m\times m}$. For convenience, columns of Y are normalized.*
3. *Let λ be the minimum of* $\mathrm{diag}(Y^THY)$, *y be the corresponding vector such that $\lambda = y^THy$,*
4. *λ is the approximation to the smallest eigenvalue, $x = Vy$ is the corresponding eigenvector approximation, the residual $r = Wy - \lambda x$ is the associated residual vector.*

Both the Rayleigh-Ritz procedure and the harmonic Ritz procedure can be used to find eigenvalues other than the minimum eigenvalue by selecting y according to different criteria. The descriptions here are geared toward the eigenvalue problem on hand. It should be a trivial change to adapt them for different needs.

The Rayleigh-Ritz projection is widely used. It is observed that Rayleigh-Ritz procedure is effective in approximating the extreme eigenvalues [95] and the harmonic Ritz projection is effective in extracting eigenvalue in the interior of the spectrum [78]. Since we are seeking extreme eigenvalues, we are naturally inclined toward using the Rayleigh-Ritz projection. However, a large number of eigenvalues is desired, some of them could be considered interior eigenvalues where harmonic Ritz projection is more suited. The focus of this chapter is on the basis generation schemes, Rayleigh-Ritz projection is used through out.

For some nonsymmetric matrices, the projected eigenvalue problems, equations (2.3) and (2.5), could be defective. In which case, the number of eigenvectors will be less than the size of the matrix. Because the Schur vectors can always be computed, if the matrix A is not symmetric, using a Schur vector as y is a more reliable choice [130, 131]. Since the matrix of interest is Hermitian, we will only use eigenvectors of (2.3) and (2.5) as y for equation (2.1).

2.2 Projection methods for eigenvalue problems

All the projection methods for eigenvalue problems can be described by the following abstract algorithm.

ALGORITHM **2.3** *A generic algorithm for solving eigenvalue problem, $AX = X\Lambda$, where $A \in \mathbf{R}^{n\times n}$, X may have arbitrary number of columns.*

1. *Given an initial basis V_i, expand its size to form $V_m \in \mathbf{R}^{n\times m}$.*
2. *Apply one of the projection schemes on V_m to generate an approximate solution.*
3. *If the residual norm of the approximation is smaller than a preset tolerance, stop, else collapse V_m to V_i and go back to step 1.*

The aim of section is to review existing schemes for generating a new basis. Different methods use different initial basis size i and maximum basis size m. More importantly, they differ on how to extend V_i to V_m.

Subspace iteration This type of methods are mainly used for finding dominant eigenvalues, and can be considered extensions of the power method. Some examples include the power method, inverse iteration, the Rayleigh-Quotient iteration [93, 95], simultaneous iterations [104, 105], the trace minimization method [56, 122], and the preconditioned subspace method [65]. Software based on this approach includes RITZIT for symmetric matrices [105], and SRRIT for nonsymmetric matrices [12]. One very distinct feature of this family of methods is that V_i and V_m are the same size. These methods are generally considered reliable because they converge to the desired solutions. However they often take many more matrix-vector multiplications than the Krylov methods we will discuss next.

In iterative sparse linear system solvers, in order to speed up the convergence, an auxiliary linear system is solved. Solving this auxiliary linear system is called *preconditioning* [10, 119]. Similar approaches have also been applied to speed up the convergence of eigenvalue solvers. Preconditioning techniques for Krylov eigenvalue solvers are more well published in the United States [19, 116] than preconditioning for the subspace iteration techniques [16, 65]. This is a slight disadvantage for the subspace iteration methods. In summary, after a brief review of this type of methods, we have decided to concentrate our attention on other methods to be discussed.

Krylov methods Given a matrix A and a vector v, Krylov subspace is the span of $\{v, Av, A^2v, \ldots\}$. Methods in this family generate Krylov subspace bases and then extract eigenvalue and eigenvector approximations from the bases. The power series, $v, Av, A^2v, \ldots$ does not make a good basis because the vectors late in the sequence could become very close to each other. One effective process for generating an orthonormal basis of a Krylov subspace is the Arnoldi algorithm described below [5, 116]

ALGORITHM **2.4 The Arnoldi method** *for generating an orthonormal basis. Given an initial vector v and maximum basis size m, let $v_1, v_2, \ldots,$ denote the columns of the basis V, $v_1 = v/\|v\|$, and $V_j = [v_1, v_2, \ldots, v_j]$ be the first j columns of V, for $j = 1, \ldots, m$, do the following,*

1. $z = Av_j$.
2. $h_{ij} = v_i^T w_j$, $i = 1, \ldots, j$; $z = z - V_j[h_{1j}, \ldots, h_{jj}]^T$.
   $h_{j+1,j} = \|z\|$, $v_{j+1} = z/\|z\|$.

This algorithm generates a basis satisfying

$$AV_m = V_m H + \beta v_{m+1} e_m, \tag{2.6}$$

where $V_m = [v_1, \ldots, v_m]$, e_m is the mth column of the identity matrix, $\beta = h_{m+1,m}$ and $H = (a_{ij})$, $i, j = 1, 2, \ldots, m$. A crucial characteristic of this equation is that the matrix H is upper Hessenberg. Any basis that satisfies this equation equation will be referred to as an Arnoldi basis. Note that this Hessenberg matrix H is also the same matrix H needed for both projection techniques, see equation (2.3) and (2.5). This is one reason why the Arnoldi algorithm is a favorite for building basis for projection methods.

For symmetric matrices, the Hessenberg matrix H becomes a tridiagonal matrix. Step 2 of the Arnoldi algorithm can be simplified since $h_{1j} = \ldots = h_{j-2,j} = 0$.. The well known Lanczos algorithm can be viewed as this simplified algorithm [50, 69]. Step 2 in algorithm 2.4 can be express as

$$h_{j-1,j} = h_{j,j-1}, h_{jj} = v_j^T z, z = z - h_{j-1,j} v_{j-1} - h_{jj} v_j . h_{j+1,j} = \|z\|, v_{j+1} = z/\|z\|.$$

Eigenvalue solvers based on this scheme are very effective in finding a few extreme eigenvalues of a matrix. Extensive studies have been done on the Lanczos algorithm for symmetric eigenvalue problems [27, 28, 67, 95, 96, 109, 154]. The books by Parlett [95] and by Cullum and Willoughby [27, 28] are good references on applying it to symmetric eigenvalue problems. A number of eigenvalue solvers based Lanczos algorithm are available from NETLIB mathematical software repository[1].

When the matrix is not symmetric, the Lanczos algorithm can be generalized in two ways: the nonsymmetric Lanczos algorithm and the Arnoldi algorithm. The nonsymmetric Lanczos algorithm still tridiagonalizes the matrix, but it requires two sets of bi-orthogonal bases [13, 11, 51, 54, 55, 116, 154]. Software packages based on nonsymmetric Lanczos algorithm are also available from NETLIB and other archives.

The Arnoldi algorithm can be used for both symmetric and nonsymmetric eigenvalue problems [5, 110]. In the symmetric case, the difference between the Lanczos algorithm and the Arnoldi algorithm is that the later one maintains an orthonormal basis by explicitly orthogonalizing the new vector against all existing vectors while the Lanczos algorithm only orthogonalizes the new vector against two previous ones. Because of this, it is easier for the Arnoldi method to maintain an orthonormal basis than the Lanczos method. Due to round-off error in digital computer, one pass through the orthogonalization step of the two method may not produce a result that is orthogonal to the existing basis. This is the loss of orthogonality problem. It is simpler to detect this problem in the Arnoldi algorithm than in the Lanczos algorithm [29]. Much research has been done on detecting and correcting the loss of orthogonality problem in Lanczos eigenvalue methods [53, 94, 96]. Many schemes have been developed as the result of the research, but they are usually quite complex. For simplicity, we prefer the Arnoldi method over the Lanczos method.

Preconditioning can be easily introduced into the Krylov eigenvalue solvers [81, 116]. Most of the preconditioners used for solving linear systems can be directly translated into preconditioned Krylov eigenvalue solvers [8, 9, 66, 76, 114, 118, 152]. Usually, preconditioning is easier to apply to the Arnoldi method than the Lanczos method. Very simply, to perform preconditioning, one can simply change step 1 of algorithm 2.4 to

$$z = M^{-1} A v_j,$$

where M is the preconditioner which approximates A in some way [116], For example, $M = \text{diag}(A) - \lambda I$ [30].

In the class of Lanczos methods, Ruhe's variation of implementation of block Lanczos method was very instrumental in our development of block eigenvalue solvers [102]. It shows that the initial block size don't have to the subsequent block size.

Given a maximum basis size, the computational complexity of the Arnoldi algorithm is about $2m^2 n$. Computing the eigenvalues and eigenvectors of H using a QR algorithm generally required $O(m^3)$ operations. For large m, these facts mean that the Arnoldi method for find eigenvalues could be very expensive. For this reason, practical Arnoldi eigenvalue solvers use relatively small m, e.g., 20 - 30. However, choice of m should be made with consideration of many factors, for example, computer memory available, number of eigenvalues sought, size of the matrix, and so on. If m is large, less restart is needed to reach convergence but building a large basis is expensive. If m is small, it is cheap to build a basis V_m, but more restart will be needed. Finding an optimal choice is an open research question [120]. Traditionally, at restart, the current Ritz vector is used as the starting vector to build another Krylov basis. This is referred to as explicit restarting to contrast with what is described next.

Implicitly Restarted Arnoldi method This is a fairly new method due to Sorensen [131] which has quickly caught the attention of many researchers [73, 83, 135]. A software package based on this technique is available [129]. It is named Implicit Restart Arnoldi (IRA) because it restarts in such a way that the starting vector for the new basis is never explicitly computed. There are two special characteristics that

[1] The URL is `http://www.netlib.org/`.

deserve some special attention. First, the Implicitly Restarted Arnoldi (IRA) method can be interpreted as an incomplete QR method [130, 131]. Thus it links the QR algorithm with the Arnoldi algorithm. A great deal of research has been done on improving the convergence rate of QR methods. Now these techniques is available to be transplanted to the Arnoldi algorithm.

Another important characteristic of IRA is in the restarting technique. IRA restarts by keeping a significant portion of the old Arnoldi basis. Even though the idea of saving more than one Ritz vector was discussed in [31], the recent interests in restarting techniques were directly sparked by the development of IRA method [70, 70, 72, 83, 135]. IRA method successfully demonstrated the potential of keeping more vectors at restart [70]. In the context of Davidson method, Stathopoulos and colleagues call this strategy of restarting *thick restart* [134]. Morgan showed an analytical argument for preserving more Ritz vectors for the Arnoldi method [83]. The approximate eigenvectors saved in the basis deflate the relevant part of the spectrum and effectively increase the gap between the wanted eigenvalues and the unwanted ones [83]. This characteristics was further exploited in [135] to produce a dynamic restarting scheme.

Davidson method The original Davidson eigenvalue method was proposed in 1975 [30]. By 1989, when Davidson published his survey, the method had gone through a series of significant improvements [31]. Many novel techniques used in the Davidson method and its variants have greatly influenced the development of other methods. It pioneered an early form of preconditioning which is still commonly used today, and it is the first to allow restarting with more vectors than the desired number of eigenvectors which is an important feature of IRA.

The Davidson method was initially designed to find a few smallest eigenvalues of the Schrödinger equation from quantum Chemistry applications [30]. Now it is a common technique for finding a few extreme eigenvalues of large matrices generated from fields ranging from numerous science and engineering fields [31, 116]. Publications available on the Davidson method are extensive [14, 26, 31, 32, 40, 41, 77, 79, 80, 86, 87, 120, 125, 136]. The original Davidson method was proposed for symmetric eigenvalue problems, but later developments have generalized it to nonsymmetric eigenvalue problems and generalized eigenvalue problems [31, 77, 79, 120]. Another area of development is the block versions of the Davidson method [32, 79]. More importantly, there has been considerable new developments in preconditioning and restarting techniques [90, 125, 133]. The scheme proposed in [90] modifies the original Davidson preconditioning scheme to make the result of preconditioning orthogonal the current eigenvector approximation. The Jacobi-Davidson method by van der Vorst and colleagues is another way of realizing the same intension starting from a different point of view [14, 41, 125]. A significant contribution from [133] is the use of biased estimate for the shift in preconditioning which can address a number of issues facing the original Davidson preconditioning scheme.

Similar to the Arnoldi method, the Davidson method also needs restart. So far only explicit restart is available for the Davidson method. Without restart, the Davidson method would eventually converge for every matrix as m approaches n. Crouzeix, Philippe, and Sadkane showed that the restarted version of the Davidson method also converges [26]. There is a strong connection between the Davidson method and Newton method for eigenvalue problem, the Davidson preconditioning step is regarded as solving an equation from the Newton method for eigenvalue problems. Due to this connection, it is believed that given a good preconditioner, the Davidson method can converge quadratically.

In short, two classes of methods appear to be good candidates for our needs, the Krylov method and the Davidson method. Next we consider one from each class for further evaluation. The Arnoldi method and the original Davidson method are selected because they are simple to implement and easy to use. Many steps in the Arnoldi method and the Davidson method are the same. For this reason, we will only give one complete eigenvalue algorithm including all steps mentioned in algorithm 2.3. We have shown the Arnoldi method for generating an orthonormal basis, and the Rayleigh-Ritz projection algorithm. So we will show a complete Davidson algorithm to demonstrate how the components work together, see [30] for the original form of the algorithm.

The Davidson eigenvalue method is the easiest to understand when described as an algorithm for finding one eigenvalue and restart with the current Ritz vector. This is what is shown in algorithm 2.5. As before V_j denote the first j columns of V. For convenience, we use V_0 to mean null, or literally first 0 column of V.

ALGORITHM **2.5 Davidson's algorithm** *for finding the smallest eigenvalue of a symmetric matrix A.*

1. **Start**. *Choose an initial guess for the eigenvector, x. Let $z = x$.*

2. **Iterate** *to build an orthonormal basis $V_m = [v_1, v_2, \ldots, v_m]$. Let $j = 1$.*

   (a) $z = z - V_{j-1}V_{j-1}^T z$, $v_j = z/\|z\|$.

   (b) $w_j = Av_j$.

   (c) $h_{ij} = v_i^T w_j$, *for* $i = 1, \ldots, j$.

   (d) if $(j < m)$ *then find a new vector z to augment V_j,*

   i. compute the residual vector r is current best approximate solution.

   ii. $z = M^{-1}r$,

   iii. Let $j = j + 1$, *goto 2.a.*

   else continue to step 3.

3. **Rayleigh-Ritz procedure** *to find the best approximate solution (λ, x) and the corresponding residual r using the basis V_m.*

4. **Convergence test**. *If $\|r\|$ is smaller than the preset limit, τ, stop, else let $z = x$, go to step 2.*

In above algorithm, the parameter m is preset and it is chosen to be a small number such as 20 or 30. Conceptually, the three steps of algorithm 2.3 are the steps 2, 3, and 4 of algorithm 2.5, where step 2 builds an orthonormal basis and step 3 performs the Rayleigh-Ritz projection on the basis to extract the wanted eigenvalue and eigenvector. More specifically, step 2.a describes a classic Gram-Schmidt orthogonalization procedure. It orthogonalizes z against existing basis vectors and append the resulting normalized vector to the basis to form a new one. This step is responsible for keeping the basis orthonormal. Step 2.b performs matrix-vector multiplication. Step 2.c computes a new column of $H = V^T AV$. Steps 2.b and 2.c are placed here to progressively compute H. If we simply let $z = w_j$, it is easy to verify that step 2 is a slightly rearranged form of the Arnoldi algorithm. The essential difference between the Davidson method and the Arnoldi method is in how they generates a new vector to augment the basis. To change algorithm 2.5 into a preconditioned Arnoldi method, we could simply skip step 2.d.i and replace step 2.d.ii with $z = M^{-1}w_j$.

2.3 Alternative input vectors

Based on the observed commonalities and differences between the Arnoldi method and the Davidson method, a generic algorithm for generating a basis of a given size can be written as follows.

ALGORITHM **2.6 A algorithm for generating an orthonormal basis** *V_m from b_0 input vectors. The basis is also required to be orthogonal to an orthonormal set X.*

0. *Place the b_0 input vectors at the first b_0 columns of V. Let $j = 0$ and $b = b_0$.*

1. *Orthonormalization.*

   $[v_{j+1}, \ldots, v_{j+b}] = (I - V_j V_j^T - XX^T)[v_{j+1}, \ldots, v_{j+b}]$, *where* $V_j = [v_1, \ldots, v_j]$.

   Perform QR decomposition on the new vectors, $QR = [v_{j+1}, \ldots, v_{j+b}]$, and replace $[v_{j+1}, \ldots, v_{j+b}]$ with first b columns of Q.

2. $j = j + b$; $b = b_1$.

3. *If $j < m$, continue, else stop.*

4. *Compute b vectors, s_1, ..., s_b to give to the preconditioner.*

5. *Apply the preconditioner,* $[v_{j+1}, \ldots, v_{j+b}] = M^{-1}[s_1, \ldots, s_b]$.

6. *Goto step 1.*

This algorithm combines a number of techniques used in more complex eigenvalue solvers. It is can be made to either reproduce algorithm 2.4 or the step 2 of algorithm 2.5. If $b_0 = b_1 = 1$ and $s_1 = Av_j$ in step 4, then it is the Arnoldi algorithm. If $b_0 = b_1 = 1$ and $s_1 = r$, then it is the generates the same basis as the Davidson algorithm 2.5. In addition, we have incorporated a few changes to make it more flexible. First it can take an arbitrary number of input vectors. This feature originates from the Davidson method [31], has been successfully used by many researchers [73, 83, 131, 135]. The augmented Krylov subspace concept is also very similar to this idea [106]. Our experiences also show this to be a worthwhile feature to have. For convenience of discussion we have split step 2.d of algorithm 2.5 to make it into four separate steps. Steps 2.b and 2.c of algorithm 2.5 are not shown in this algorithm because they are primarily used to produce the projected matrix H_m for the Rayleigh-Ritz procedure. In practice we may compute H_m progressively either for efficiency concern or to use the intermediate H_j in the step 4 of algorithm 2.6. A good example of this is the Davidson method where it is necessary to compute the residual vectors for preconditioning, the matrix H_j has to be available at every step.

Step 1 of the above algorithm is the orthonormalization step. It can be viewed as a progressive QR decomposition which orthonormalizes new vectors as they are generated. The basis is also required to be orthogonal to an additional set of vectors here. Usually this additional set of vectors are converged eigenvectors. This is part of the process necessary for locking converged eigenvectors out of the working basis. What is shown here is only intended to be a mathematical formula, some attention to details are required to guarantee the quality of basis produced. This issue and many other implementation issues are addressed in next chapter. Although it is possible to set b_1 to any number in step 2 of algorithm 2.6, we have found it to be more effective if b_1 is set to 1 in most cases. We use $b_1 = 1$ through out the thesis unless specifically stated otherwise. Both b_0 and b_1 may be referred to as the block size [102, 116], in our discussion, we will always call b_1 the block size and b_0 the initial basis size. Since b_1 is usually 1, we will drop subscript 1 in s_1 is step 4 and 5 in algorithm 2.5.

In the Arnoldi method, if the current block size b is less than the value in the proceeding step, there is a choice to be made on what to include in s. A simple scheme we use is to always choose the first b new Av_j's. This can be viewed as a different implementation of the augmented Krylov subspace described in [18, 106]

In the jth step of the Davidson method, it is possible to compute j residual vectors. The process of choosing b out of j residual vectors is commonly referred to as targeting. A simple scheme we will use here is to choose b residual vectors corresponding to the first b wanted eigenvalues for most of our experiments. No matter what is the size of the block, there is only one eigenvalue we consider as the targeted eigenvalue at any time. This relates to the choice of shift used in the preconditioner.

In the frame work of this generic basis generation algorithm, we can vary what is generated in the step 4 to significantly affect what basis is generated. The main focus of this chapter is to vary the input vector generated in this step of the algorithm. Next we will given each one of them one at a time.

Orthogonalized Arnoldi scheme One difference between the Davidson method and the Arnoldi method is that the Davidson method needs to compute the current residual vector at every step. If we neglect the operations required to solve the small eigenvalue problem, and assume that both V_j and $W_j \equiv AV_j$ are saved in memory, computing one residual vector needs about $2jn$ floating-point multiplications and $2jn$ additions. This is a significant amount of floating-point operations. We would like to reduce this operation count if possible. Notice that in many cases the Davidson method works better than the Arnoldi method, the question becomes what properties of the residual vector is important to the performance of the Davidson method. The Davidson method uses Rayleigh-Ritz projection to extract eigenvalue approximation. From the orthogonality condition, equation (2.2), we know that the residual of the approximate solution is orthogonal to the current basis. If the current basis satisfies the Arnoldi recurrence relation $AV_j = V_j H_j + \beta u_{j+1} e_j^T$, then the residual of the Rayleigh-Ritz approximation is

$$r = AV_j y - \lambda V_j y = V_j H_j y - \lambda V_j y + \beta u_{j+1} e_j^T y.$$

Since $H_j y = \lambda y$, see equation (2.3),

$$r = \beta e_j^T y u_{j+1}.$$

According to the Arnoldi recurrence, $w_j \equiv Av_j = \sum_{i=1}^{j} h_{ij}v_i + \beta u_{j+1}$, if we orthogonalize w_j at step 4 of algorithm 2.6, the vector given to the preconditioner will be

$$s = \beta u_{j+1},$$

which is parallel to the residual of the Davidson algorithm. Therefore the two schemes would generate the bases spanning the same space.

There is a fairly inexpensive way of modifying the Arnoldi method to make Av_j orthogonal to the basis vectors.

$$(I - V_j V_j^T)Av_j = Av_j - V_j(V_j^T Av_j)$$

Since $V_j^T Av_j$ is already computed in the process of computing $H_j = V_j^T AV_j$, orthogonalizing Av_j against V_j is only half as expensive as computing a residual vector. Let $h_j \in \mathbf{R}^j$ denote $V_j^T Av_j$, the preconditioning vector of this scheme is

$$s = Av_j - V_j h_j. \tag{2.7}$$

In theory, this vector is orthogonal to the basis V_j. If no preconditioning is used, this would be the redundant because it will be explicitly orthogonalized against V_j again in step 1 of next iteration, see algorithm 2.6.

Modified Arnoldi method This scheme can be considered a trivial implementation of the approximate Cayley transformation scheme presented in [75]. If the current targeted eigenvalue is λ, this scheme compute the s as follows

$$s = (A - \lambda I)v_j. \tag{2.8}$$

This is a simple modification to the Arnoldi method. Since Av_j is computed while computing H, only $2n$ additional floating-point operations are needed to compute s.

The advantages of this scheme is articulated in [75]. One of the main points is that the preconditioned iteration matrix $M^{-1}(A - \lambda I)$ has one eigenvector that is close to the solution. The eigenvalue scheme presented in [75] is a new way of combining the Arnoldi method and the Davidson method.

Harmonic Davidson scheme The Davidson method computes the residual vector based on Rayleigh-Ritz projection. The Rayleigh-Ritz procedure is shown to have many optimality conditions, see for example [95]. However it is not the only way to extract eigenvalue approximation from a basis V_m. Notably the harmonic Ritz values has attracted much attention in research community [78, 82, 91, 125]. The harmonic Ritz value techniques proposed in [78] can be immediately applied to the Davidson method by replacing the Rayleigh-Ritz procedure with the harmonic Ritz procedure. According to the norm used to maintain orthogonality among the basis vectors, three ways of implementing the harmonic Ritz procedure was proposed in [78], (1) use 2-norm, (2) use $A^T A$-norm, i.e., maintain W_m 2-norm orthonormal, or (3) use A-norm, i.e., maintain V_m and W_m bi-orthogonal. In [78], an example of harmonic Lanczos method which maintained V_m 2-norm orthonormal was shown. In [125], the authors proposed a harmonic Davidson method which maintains V_m and W_m bi-orthogonal.

The scheme we will use here is a mixture of both the Rayleigh-Ritz technique and the harmonic Ritz technique. In particular, we use harmonic Ritz values to generate new input vectors to the preconditioner, the vector s in step 4 of algorithm 2.6, but use the Rayleigh-Ritz procedure to generate the approximate solution when the basis is full after algorithm 2.6. Because we keep the basis V orthonormal in Davidson's method, we use the harmonic Ritz procedure proposed in [78].

This scheme is more expensive compared to the Davidson scheme mainly because it needs both H_m and G_m. Compared to the Davidson method, this scheme need $2jn$ extra FLOP to update G_j into G_{j+1}.

2.4 Equivalence properties

So far we have described five different ways of generating orthonormal basis V_m, (1) the Arnoldi method, (2) the Davidson method, (3) the orthogonalized Arnoldi method, (4) the modified Arnoldi method, and (5) the harmonic Davidson method. This section compares the different schemes without preconditioning, i.e., $M = I$ in step 5 of algorithm 2.6.

Starting with same initial vectors, all other aspects being equal, the basis V_m built by the Arnoldi method and the orthogonalized Arnoldi method are identical because the extra orthogonalization step in the orthogonalized Arnoldi method does not alter the resulting basis vector at any step. This is true in theory, in practice because of round-off error, the orthogonalized Arnoldi is slightly more stable because of the extra orthogonalization step. The modified Arnoldi method also produces exact the same basis at the original Arnoldi method. To compare the other two methods, we need the help of the following lemma.

Lemma 2.1 *Assume V_j is an Arnoldi basis, see equation (2.6), the residual of an arbitrary Ritz pair (λ, x) generated according to equation (2.1) is in the space spanned by $[V_j, u_{j+1}]$.*

Proof. Given an arbitrary normal vector $y \in \mathbf{R}^j$, the Ritz pair generated by equation (2.1) has residual $r = AV_jy - \lambda V_jy$. According to the assumption, $AV_j = V_jH_j + \beta u_{j+1}e_j^T$, the following is true,

$$AV_j \in \text{span}\{V_j, u_{j+1}\}.$$

Thus r is a linear combination of vectors in $[V_j, u_{j+1}]$, i.e., $r \in \text{span}\{V_j, u_{j+1}\}$. □

The implication of the above lemma is that if V_j is an Arnoldi basis, any vector of the form $AV_jy - \lambda V_jy$ can be used at step 4 of algorithm 2.6, the new basis is another Arnoldi basis, and $v_{j+1} = u_{j+1}$. The above lemma can be applied to both the Davidson method and the harmonic Davidson method because y can be any unit vector.

It is fairly straightforward to extend the above lemma to a block version of the Arnoldi basis. We extend equation (2.6) to the block form as follows

$$AV_j = V_jH_j + U[e_{j-b+1}, \ldots, e_j]^T B, \tag{2.9}$$

where H_j is a b-Hessenberg matrix that has b non-zero subdiagonal elements, $B \in \mathbf{R}^{b\times b}$, $U \in \mathbf{R}^{n\times b}$ and e_i is the ith column of the identity matrix. The above lemma can be trivially extended as follows.

Lemma 2.2 *Assume V_j is a block Arnoldi basis, see equation (2.9), the residual of an arbitrary Ritz pair (λ, x) is in the space spanned by $[V_j, U]$.*

Lemma 2.3 *If at any step j, the basis V_j created by the orthogonalized Arnoldi method, the modified Arnoldi method, the Davidson method or the harmonic Davidson method span the same space as the basis of the Arnoldi method, and $[V_j, s_1, \ldots, s_b]$ is not degenerate, then the basis V_{j+b} will also span the same space.*

Proof. It is clear that V_{j+b} produced by the Arnoldi method, the orthogonalized Arnoldi method and the modified Arnoldi method will be in the span of $[V_j, U]$ if they are not degenerate. All the residual vectors of the Davidson method and the harmonic Davidson method are in span$[V_j, U]$. If the original basis and b residual vectors do not form a degenerate set, they must form a basis of span$[V_j, U]$. In conclusion, the bases produced by the five methods span the same space. □

If the number of initial guesses is the same as the block size b, than the basis satisfies equation (2.9). If the same initial guesses are used in the five methods mentioned, the bases produced by them should span the same space till the end of algorithm 2.6. If these bases are given to the same projection procedure to find the same eigenpairs, the resulting Ritz vectors should be identical for all methods. If the new set of vectors after restart also satisfies equation (2.9), then the five methods will be identical through-out. For symmetric matrices, if we choose to save more than b Ritz vectors to restart, the new vectors satisfying equation (2.9).

The results stated here combine and generalize some of the previous equivalence theorems. Previously, the restarted Arnoldi method with arbitrary b_0 was shown to be mathematically equivalent to the implicitly restarted Arnoldi method [83], and the thick restarted Davidson method was shown to be equivalent to the implicitly restarted Arnoldi method in [134]. Combining these two, we can conclude that the restarted Arnoldi method is equivalent to the restarted Davidson method if they start with one initial vector at the beginning, restart with the same Ritz vectors, and add only one vector to their bases at each step. Because the strong connection between the implicitly restarted Arnoldi (IRA) method and the thick restarted Arnoldi for symmetric matrices, the above lemmas also extend to IRA method.

method	extra FLOP
Arnoldi	-
Modified Arnoldi	2n
Orthogonalized Arnoldi	2jn
Davidson	4jn
Harmonic Davidson	6jn

Table 2.1: Complexity of the methods compared to the Arnoldi method.

2.5 Practical differences

We have implemented the five schemes for finding eigenvalues and eigenvectors using the frame work of algorithm 2.3. In our implementation, we save both V and W in memory to avoid extra matrix-vector multiplications when computing residual vectors. Comparing the five methods, the Arnoldi method is the cheapest per step. Table 2.1 shows how many more floating-point operations used by other four methods compared to the Arnoldi method. The comparison is for the step that extends V_j to V_{j+1}. Because the orthogonalization procedure, step 1 of algorithm 2.6, is implemented as an iterative procedure [29], the actual number of floating-point operations of the orthogonalization step depends on the quality of the input vectors, primarily how close is the new vector to be a linear combination of the existing ones. Therefore the actual differences may differ from what is shown in the table. More details on the orthogonalization procedure will be discussed later. Without preconditioning, the five methods discussed are equivalent to each other in theory. Here we have just shown two differences among them. First, they have different complexities. Second, quality of the basis generated by them are different. When preconditioned, they are even more different.

For convenience of discussion, let's assume $b_1 = 1$ in algorithm 2.6, the vector s generate at this step can always be expressed as

$$s = \sum_{i=1}^{j} \alpha_i v_i + \beta u. \tag{2.10}$$

where $\alpha_i = s^T v_i$, $\beta = \|s - \sum \alpha_i v_i\|$, $u = (s - \sum \alpha_i v_i)/\beta$. If β happens to be zero, u can be any normal vector orthogonal to V_j. Since the orthogonalization step will normalize the vector, the ratio of β compared to $\|(\alpha_i)\|$ is an important quantity. Because the residual of the Rayleigh-Ritz procedure is orthogonal to the basis V_j, this ratio could be considered as infinite for the Davidson method. In theory the orthogonalized Arnoldi should also have infinite ratio for $\beta/\|(\alpha_i)\|$. But this ratio is reduced to finite number if Av_j is close to be linearly dependent on V_j, because equation (2.7) is not enough to generate a vector that is exactly orthogonal to V_j. The other three methods, the Arnoldi method the modified Arnoldi method and the harmonic Davidson method could have much smaller ratios. This indicates that even if u in equation (2.10) is the same for all five methods, $M^{-1}s$ could be very different. Most often the vector u is not the same either because the bases do not satisfy the Arnoldi recurrence, or because round-off errors have accumulated to be significant. Due to these differences, even if V_j is the same for all five methods mentioned above, they will most likely produce different v_{j+1}. Usually the Davidson method uses a varying preconditioner such as $M = \mathrm{diag}(A) - \lambda I$ where λ is updated at every step, which makes it even harder to quantify the differences between the methods.

Even if the preconditioner inverts $(A - \lambda I)$ exactly, unless V_j is an invariant subspace of A, for different value of $\beta/\|(\alpha_i)\|$, the result of preconditioning is different. More importantly, the new basis built by the five different methods will be different. This means that even with a perfect preconditioner, they will produce different results.

The most effective way of showing how the methods behavior is showing numerical examples. A series of numerical results will be shown in the remaining of this section. The test matrices used are non-diagonal symmetric matrices. To reduce bias in the data, we have used all symmetric matrices in the collection. They came from a variety of sources, such as structure analysis, etc., see table 2.2 and 2.3. Most of the matrices are from real applications. The performance of the eigenvalue solvers on these problems will give

NAME	N	NNZ	comment
1138BUS	1138	2596	admittance matrix, power system
494BUS	494	1080	admittance matrix, power system
662BUS	662	1568	admittance matrix, power system
685BUS	685	1967	admittance matrix, power system
GR3030	900	4322	nine point matrix on a 30 X 30 grid
LUNDA	147	1298	A of Lund eigenvalue problem
LUNDB	147	1294	B of Lund eigenvalue problem
NOS1	237	627	biharmonic operator on beam
NOS2	957	2547	biharmonic operator on beam
NOS3	960	8402	biharmonic operator on plate
NOS4	100	347	beam structure
NOS5	468	2820	building structure
NOS6	675	1965	Poisson's equation in L shape, mixed BC
NOS7	729	2673	Poisson's equation in unit cube
PLAT1919	1919	17159	ocean model
PLAT362	362	3074	Atlantic ocean model
ZENIOS	2873	15032	air-traffic control model

Table 2.2: Non-diagonal symmetric matrices from HB collection (Part I).

us some indications as how well they will perform in general. The maximum basis size is chosen to be 25, a total of about 50 vectors are stored in memory. The results show in the table are from finding 5 smallest eigenvalues and the corresponding eigenvectors. In the application we are interested in, many more eigenpairs are needed. Here we are only performing a small test on the eigenvalue solvers. For simplicity the initial vector to build the first basis is always a vector of all ones, $[1, 1, \ldots, 1]^T$. At restart, 12 Ritz vectors corresponding to the smallest Ritz values are saved. There are 45 non-diagonal symmetric matrices in the Harwell-Boeing collection. To save space, only the cases where five eigenvalues were found within 5000 matrix-vector multiplications. The entry with a dash ("-") indicates that the eigenvalue solver did not find 5 eigenvalues within 5000 matrix-vector multiplications.

Most of the test matrices are very sparse, typically having only about 5 to 10 nonzero entries per row. This about results were obtained on a SPARC-10 workstation runs at about 50MHz. All the Harwell-Boeing matrices shown in the table can fit into the main memory of the computer, which is 64 Megabytes. Because the typical matrix-vector operation is less expensive than an average orthogonalization in terms of floating-point operations, the complexity of the whole eigenvalue solver is dominated by the orthogonalization and projection steps. Table 2.1 clearly shows that the Arnoldi method is cheaper than others. When the number of the matrix-vector multiplication is about the same, the Arnoldi method should use the least amount of time.

The first set of test results is shown in table 2.4. We have run the five methods on all 45 test problems. Only four methods are shown in the table because the number of matrix-vector multiplications (MATVEC) used by the modified Arnoldi method is exactly the same as that of the Arnoldi method. For each of the four methods shown, two columns are displayed for each one of them. The one headed by "MATVEC" is the number of matrix-vector multiplications used and the one headed with "time" shows the total execution time in seconds. From this table, we see that the Arnoldi method does use less time if about the same number of matrix-vector multiplications is used. Comparing the number of matrix-vector multiplications used by the different methods, the Davidson method clearly uses less than any other.

Generally, we can observe two trends from the data in table 2.4,

- The Davidson method generally use less iterations to converge than others;
- The Arnoldi method uses less time than others when same number of matrix-vector multiplication is used.

NAME	N	NNZ	comment
BCSSTK01	48	224	stiffness matrix
BCSSTK02	66	2211	stiffness matrix, small oil rig
BCSSTK03	112	376	stiffness matrix
BCSSTK04	132	1890	stiffness matrix, oil rig
BCSSTK05	153	1288	stiffness matrix, transmission tower
BCSSTK06	420	4140	stiffness matrix
BCSSTK07	420	4140	stiffness matrix
BCSSTK08	1074	7017	stiffness matrix, frame building (TV studio)
BCSSTK09	1083	9760	stiffness matrix, square plate clamped
BCSSTK10	1086	11578	stiffness matrix, buckling of hot washer
BCSSTK11	1473	17857	stiffness matrix, ore car (lumped masses)
BCSSTK12	1473	17857	stiffness matrix, ore car (consistent masses)
BCSSTK13	2003	42943	stiffness matrix, fluid flow
BCSSTK14	1806	32630	stiffness matrix, roof of omni coliseum, Atlanta
BCSSTK15	3948	60882	stiffness matrix, module of an offshore platform
BCSSTK16	4884	147631	stiffness matrix, corp. of engineers dam
BCSSTK17	10974	219812	stiffness matrix, elevated pressure vessel
BCSSTK18	11948	80519	stiffness matrix, R.E.Ginna nuclear power station
BCSSTK19	817	3835	stiffness matrix, part of a suspension bridge
BCSSTK20	485	1810	stiffness matrix, frame within a suspension bridge
BCSSTK21	3600	15100	stiffness matrix, clamped square plate
BCSSTK22	138	417	stiffness matrix, textile loom frame
BCSSTK23	3134	24156	stiffness matrix, 3D building
BCSSTK24	3562	81736	stiffness matrix, winter sports arena
BCSSTK25	15439	133840	stiffness matrix, 76 story skyscraper
BCSSTK26	1922	16129	stiffness matrix, reactor containment floor
BCSSTK27	1224	28675	stiffness matrix buckling problem
BCSSTK28	4410	111717	solid element model (MSC NASTRAN)
BCSSTM07	420	3836	mass matrix
BCSSTM10	1086	11589	mass matrix, buckling of hot washer
BCSSTM12	1473	10566	mass matrix, ore car (consistent masses)
BCSSTM13	2003	11973	mass matrix, fluid flow generalized eigenvalues
BCSSTM27	1224	28675	mass matrix, buckling problem

Table 2.3: Non-diagonal symmetric matrices from HB collection (Part II).

The orthogonalized Arnoldi method was built as the hybrid of the Arnoldi method and the Davidson method. In a couple of cases, it used less matrix-vector multiplications than all others, e.g., 662BUS and BCSSTK22. However the advantage of this method is not significant. The harmonic Davidson method is too expensive per step to be competitive in this experiment.

Table 2.5 shows the results of using diagonal preconditioner, $M = \text{diag}(A) - \lambda I$. In this table, all five methods are shown, but only the time for the Davidson method is shown. The value λ in the preconditioner is the latest available eigenvalue approximation. In the Davidson method and the harmonic Davidson method the eigenvalue is updated as every step. In the Arnoldi method and the orthogonalized Arnoldi method, it is only updated after the bases are full. We have made two modifications to it. Here is how M is defined in our program,

$$m_{ii} = \begin{cases} |a_{ii} - \lambda|, & |a_{ii} - \lambda| > \epsilon \\ \epsilon, & |a_{ii} - \lambda| \leq \epsilon \end{cases} \tag{2.11}$$

Table 2.5 only shows the 28 cases where all five wanted eigenpairs were found by one of the methods. The Davidson method solve more problems than any other method in this case. Even in the cases where other method also found all 5 eigenpairs, it always uses less iteration and less time.

	Arnoldi		Orthogonalized		Davidson		Harmonic	
	MATVEC	time	MATVEC	time	MATVEC	time	MATVEC	time
662BUS	3070	44.0	2876	44.6	2990	75.9	3591	127.5
685BUS	1615	24.8	1628	25.9	1591	41.0	1627	57.1
BCSSTK01	400	0.9	376	0.9	359	3.2	418	5.7
BCSSTK02	182	0.6	169	0.6	154	1.6	194	3.0
BCSSTK04	4648	17.9	4975	21.0	3908	42.6	>5000	-
BCSSTK05	612	2.6	612	2.9	605	7.4	612	11.4
BCSSTK09	730	18.5	679	19.7	336	14.4	660	39.5
BCSSTK16	794	144.0	718	147.1	718	177.5	848	252.9
BCSSTK22	2173	8.2	2083	8.7	2146	24.3	2290	40.1
BCSSTM10	183	4.6	183	5.2	183	7.9	183	10.2
BCSSTM27	2629	99.7	2526	105.2	2425	141.3	2799	208.3
GR3030	286	5.4	157	3.6	56	1.8	211	10.4
LUNDA	1406	5.8	1419	6.3	1305	14.7	1315	24.5
LUNDB	2290	9.2	2200	9.9	2190	25.4	2252	41.0
NOS3	441	9.0	380	9.4	364	14.0	380	19.9
NOS4	158	0.5	146	0.5	125	1.3	146	2.3
NOS5	1171	11.7	1133	13.0	1133	23.3	1133	33.1
ZENIOS	118	8.7	132	11.7	98	9.8	206	26.6

Table 2.4: Solving non-diagonal systems without preconditioning.

	Arnoldi	Orthogonalized	Modified	Davidson		Harmonic
	MATVEC	MATVEC	MATVEC	MATVEC	time	MATVEC
494BUS	>5000	>5000	>5000	3310	72.3	>5000
662BUS	>5000	>5000	>5000	872	23.5	>5000
685BUS	>5000	>5000	>5000	773	21.3	>5000
BCSSTK01	2864	653	2566	123	1.1	965
BCSSTK02	2586	1303	2638	192	2.0	925
BCSSTK03	>5000	>5000	>5000	1804	20.1	>5000
BCSSTK04	>5000	3681	>5000	185	2.2	2274
BCSSTK05	>5000	>5000	>5000	414	5.4	>5000
BCSSTK06	>5000	>5000	>5000	1926	37.7	>5000
BCSSTK07	>5000	>5000	>5000	1926	37.6	>5000
BCSSTK08	>5000	>5000	>5000	919	39.6	>5000
BCSSTK09	4369	>5000	4350	780	35.1	2655
BCSSTK15	>5000	>5000	>5000	4644	862.7	>5000
BCSSTK16	>5000	>5000	>5000	808	206.4	3421
BCSSTK21	>5000	>5000	>5000	1514	233.2	>5000
BCSSTM07	>5000	>5000	>5000	357	7.1	>5000
BCSSTM10	1849	1862	4682	258	11.6	3513
BCSSTM13	>5000	>5000	>5000	313	25.1	2342
BCSSTM27	>5000	>5000	>5000	2587	152.5	>5000
GR3030	259	223	260	56	1.9	205
LUNDA	>5000	1940	>5000	225	2.7	1933
LUNDB	>5000	4007	>5000	356	4.3	>5000
NOS3	2236	4915	2315	612	24.5	2173
NOS4	1766	1290	1751	216	2.5	1602
NOS5	4768	>5000	4976	801	17.0	3049
NOS6	>5000	>5000	>5000	2504	68.1	>5000
NOS7	>5000	>5000	>5000	182	5.3	>5000
ZENIOS	130	120	131	85	9.1	131

Table 2.5: Solving the non-diagonal systems with diagonal preconditioning.

	Arnoldi		Orthogonalized	
	MATVEC	time	MATVEC	time
494BUS	>5000	-	118	66.7
BCSSTK01	3045	28.3	705	6.5
BCSSTK02	2612	26.9	1355	14.3
BCSSTK04	>5000	-	3864	44.3
BCSSTK07	>5000	-	198	13.1
BCSSTK09	4756	169.2	>5000	-
BCSSTM07	>5000	-	133	15.3
BCSSTM10	1810	66.1	1927	76.4
GR3030	195	5.0	235	7.4
LUNDA	>5000	-	1979	23.3
LUNDB	>5000	-	4084	48.5
NOS3	2262	71.1	4942	170.4
NOS4	1868	20.5	1303	15.1
NOS6	>5000	-	1680	41.9
ZENIOS	158	13.0	119	11.6

Table 2.6: Solving non-diagonal systems with diagonal preconditioning using better shifts.

Comparing table 2.4 and 2.5, we see that the Davidson method is the only method benefited from the diagonal preconditioner. All others converged on less problems with this preconditioner than without preconditioning.

We have observed some characteristics among the five eigenvalue routines when preconditioner is used. Now we will try to satisfy ourselves that these characteristics are not due to some peculiarity of the diagonal preconditioner used. First thing we noticed is that we did not update the shift to the preconditioner in the Arnoldi method and the orthogonalized Arnoldi method, we have modified them to compute the eigenvalue approximation at every step. Table 2.6 contains the results of this experiment. The orthogonalized Arnoldi method can now converge on 14 systems rather this 10 in table 2.5. However this is still only half of the number of systems solved by the Davidson method. More about shifting strategy will be discussed later, here we note that changing shifting strategy does not seem to make the orthogonalized Arnoldi comparable with the Davidson method.

Table 2.7 shows the results of using SOR as preconditioner to find the smallest eigenvalues of the Harwell-Boeing matrices. More systems reached convergence with SOR preconditioner than with the diagonal preconditioner. A total of 35 systems reached convergence in this case. However, again we see the same trend that is present in the diagonal preconditioning case, that is, the method that is most successful in taking advantage of the SOR preconditioner is the Davidson method. The Arnoldi method is lagging behind others in number of systems converged.

Both the diagonal scaling and SOR preconditioner are often considered weak preconditioners for linear systems. Next we use two incomplete LU factorizations preconditioners. They are often regarded as more robust for linear systems. The results is shown in tables 2.8 and 2.9. The first ILU preconditioner is ILU0 which preserves the sparsity pattern of the matrix in the LU factor [48, 76, 115]. The second ILU preconditioner is ILUTP which is an incomplete LU factorization with column pivoting, threshold based dropping and fill based dropping strategies [118]. Because the matrix $(A - \lambda I)$ is often indefinite, ILUTP is more stable than ILUT. Both ILU0 and ILUTP are from SPARSKIT [115]. The ILUTP preconditioner has three parameters, the level of fill, the drop tolerance and the pivoting threshold. In our experiment, the drop tolerance is 10^{-4} and pivoting threshold is 0.1, the level of fill is set to a value such that each row of L and U factor could have one more non-zero element than the average number of nonzero elements in the lower and upper triangular part of the original matrix. Our intend is to find out that given the same preconditioner which one of the five methods performs better, so we simply allowed the program to perform a new factorization every time a different shift is generated. This is not the most time-efficient way of using these two preconditioners, however it does not affect our ability to compare how the five different eigenvalue solvers behave. As before, we see that the Davidson method reaches converge on more problems than others.

	Arnoldi	Orthogonalized	Modified	Davidson		Harmonic
	MATVEC	MATVEC	MATVEC	MATVEC	time	MATVEC
1138BUS	>5000	>5000	>5000	4241	240.8	>5000
494BUS	>5000	>5000	>5000	4413	126.3	>5000
662BUS	>5000	>5000	>5000	393	13.6	3785
685BUS	>5000	>5000	>5000	353	12.8	3784
BCSSTK01	>5000	471	>5000	61	0.5	913
BCSSTK02	3838	796	3513	93	1.6	1002
BCSSTK03	>5000	>5000	>5000	393	4.6	>5000
BCSSTK04	>5000	1082	>5000	86	1.7	1291
BCSSTK05	>5000	2330	>5000	145	2.2	2037
BCSSTK06	>5000	>5000	>5000	392	14.2	>5000
BCSSTK07	>5000	>5000	>5000	392	14.1	>5000
BCSSTK08	>5000	>5000	>5000	320	28.1	>5000
BCSSTK09	>5000	>5000	>5000	390	26.5	2733
BCSSTK10	>5000	>5000	>5000	1482	99.8	>5000
BCSSTK14	>5000	>5000	>5000	1808	283.1	>5000
BCSSTK15	>5000	>5000	>5000	846	363.5	>5000
BCSSTK16	>5000	>5000	>5000	234	251.4	3111
BCSSTK21	>5000	>5000	>5000	626	119.5	>5000
BCSSTK22	>5000	>5000	>5000	>5000	-	2057
BCSSTK27	>5000	>5000	>5000	1502	143.3	>5000
BCSSTM07	>5000	2369	>5000	159	4.7	3344
BCSSTM10	2161	913	>5000	95	6.6	>5000
BCSSTM12	>5000	>5000	>5000	2910	237.0	>5000
BCSSTM13	>5000	2980	>5000	113	15.6	2640
BCSSTM27	>5000	>5000	>5000	481	48.3	>5000
GR3030	1224	575	1095	113	5.5	662
LUNDA	>5000	757	>5000	104	1.7	1251
LUNDB	>5000	1381	>5000	115	1.7	1147
NOS1	>5000	>5000	>5000	4316	66.8	>5000
NOS3	>5000	2070	>5000	188	12.9	2928
NOS4	2629	809	2811	94	1.0	551
NOS5	4990	2901	4926	310	9.2	1927
NOS6	>5000	>5000	>5000	354	12.4	>5000
NOS7	>5000	1173	>5000	63	2.8	3946
ZENIOS	1386	1092	3899	104	13.4	1142

Table 2.7: Solving non-diagonal systems with SOR preconditioning.

Even on the problems where more than one method reached convergence, the Davidson method is often the one that takes the least amount of time and the least number of matrix-vector multiplications.

Table 2.10 sums up the total number of eigenvalue problems converged under each method with or without preconditioning. Without preconditioning, the five methods converged on the same problems. With preconditioning, the Davidson method converges on more problems than any other method. Table 2.11 counts the number of cases where the method converged in the least amount of time. In the unpreconditioned case, the Arnoldi method uses the least amount of time to reach convergence on this set of problems. While in the preconditioned cases, the Davidson method invariably reaches convergence faster than others.

All preconditioners used up to now are imperfect, i.e., they solve an approximate form of $(A - \lambda I)z = s$. What if we have this equation solved exactly? In the linear system case, if the preconditioner is exactly the matrix itself, the linear system is solved in one step. In the eigen-system case, this is not true. We have observed some performance difference between the five eigenvalue solvers, the particular question we would like to find an answer to is "will the eigenvalue solvers become identical if the preconditioner is exact?". To answer this, we decide to apply the diagonal preconditioner on a series of diagonal matrices. The diagonal matrices we use are from the Harwell-Boeing collection, see table 2.12. For our purpose here, the important

	Arnoldi	Orthogonalized	Modified	Davidson		Harmonic
	MATVEC	MATVEC	MATVEC	MATVEC	time	MATVEC
1138BUS	>5000	>5000	>5000	798	37.3	>5000
494BUS	>5000	>5000	>5000	624	14.6	>5000
662BUS	>5000	>5000	>5000	271	8.1	2387
685BUS	>5000	>5000	>5000	1355	73.6	>5000
BCSSTK01	>5000	>5000	>5000	65	0.7	645
BCSSTK02	25	106	159	19	1.1	93
BCSSTK03	>5000	>5000	>5000	106	1.3	3018
BCSSTK04	>5000	>5000	>5000	93	2.2	1277
BCSSTK05	>5000	1420	>5000	107	1.9	2447
BCSSTK06	>5000	>5000	>5000	164	7.4	4056
BCSSTK07	>5000	>5000	>5000	164	8.1	4056
BCSSTK08	>5000	3083	>5000	134	18.0	>5000
BCSSTK20	>5000	>5000	>5000	>5000	-	4199
BCSSTK22	>5000	>5000	>5000	>5000	-	909
BCSSTK23	>5000	>5000	>5000	>5000	-	713
BCSSTK24	>5000	>5000	>5000	1682	3367.7	>5000
BCSSTK26	>5000	>5000	>5000	4318	551.2	>5000
BCSSTK27	>5000	>5000	>5000	110	20.7	>5000
BCSSTM07	>5000	>5000	>5000	97	3.0	1531
BCSSTM10	2317	2720	>5000	99	7.5	2317
BCSSTM12	>5000	4045	>5000	110	12.7	>5000
BCSSTM13	>5000	2772	>5000	49	8.2	698
BCSSTM27	>5000	>5000	>5000	93	14.0	>5000
GR3030	2837	653	2200	89	3.8	800
LUNDA	>5000	731	>5000	64	1.2	1035
LUNDB	>5000	>5000	>5000	75	1.3	888
NOS3	>5000	1771	>5000	130	9.7	1771
NOS4	3864	106	3656	3864	46.7	1212
NOS5	>5000	2135	>5000	140	4.4	2135
NOS6	>5000	692	>5000	320	10.0	>5000
NOS7	>5000	>5000	>5000	58	2.5	4510
ZENIOS	587	726	>5000	148	22.6	713

Table 2.8: Solving non-diagonal systems with ILU0 preconditioning.

thing is that they are diagonal matrices. Table 2.13 shows the results of unpreconditioned case and table 2.14 shows the results of preconditioned case. In this test, the preconditioner improved the performance of every method, however the Davidson method still uses less matrix-vector multiplications and/or time than other four methods in most cases.

Since the Arnoldi method we implement is always lagging behind the Davidson method, it is nature to suspect that our implementation might be flawed in some way. To show that we have made a reasonable sound implementation, we compared our results against a few public domain eigenvalue packages. The package that is closest to our implementation of the Arnoldi method is the ARPACK [129]. In table 2.15 we show the number of matrix-vector multiplication and time used to reach convergence by both IRA(25) and IRA(50). Note that the basis size we use in all our tests is 25 which is the same as IRA(25). However we need 50 vectors to save both V and W, thus the total workspace size is very close to IRA(50). Table 2.15 only shows the matrices who reached convergence on all 5 eigenvalues within 5000 matrix-vector multiplications. Compared with table 2.4, the Arnoldi method we used converged on slightly more problems, and it is very competitive in both the number of matrix-vector multiplications and the time used. We know that the five methods will behave differently with perfect preconditioner. This tests shows that the Davidson preconditioning scheme is better than others with perfect preconditioner too.

	Arnoldi	Orthogonalized	Modified	Davidson		Harmonic
	MATVEC	MATVEC	MATVEC	MATVEC	time	MATVEC
1138BUS	>5000	>5000	>5000	850	50.9	>5000
494BUS	>5000	>5000	>5000	1090	31.8	>5000
662BUS	>5000	>5000	>5000	326	11.8	>5000
685BUS	>5000	>5000	>5000	282	11.0	>5000
BCSSTK01	>5000	796	>5000	73	0.7	3925
BCSSTK02	523	237	3786	25	0.9	>5000
BCSSTK04	>5000	1069	>5000	79	3.0	4993
BCSSTK05	>5000	1134	>5000	94	2.0	>5000
BCSSTK08	>5000	>5000	>5000	554	75.5	>5000
BCSSTK09	>5000	2109	>5000	161	17.8	>5000
BCSSTK14	>5000	>5000	>5000	204	202.2	>5000
BCSSTK15	>5000	>5000	>5000	1499	3665.4	>5000
BCSSTK16	>5000	4293	>5000	102	507.5	1550
BCSSTK21	>5000	>5000	>5000	791	237.1	>5000
BCSSTK22	>5000	>5000	>5000	117	1.6	2641
BCSSTM07	>5000	3214	>5000	77	3.9	>5000
BCSSTM10	744	718	>5000	97	15.8	>5000
BCSSTM13	>5000	>5000	>5000	156	26.4	>5000
BCSSTM27	>5000	>5000	>5000	59	28.9	>5000
GR3030	2922	822	2570	137	7.4	>5000
LUNDA	>5000	952	>5000	88	2.3	>5000
LUNDB	>5000	>5000	>5000	82	1.7	>5000
NOS1	>5000	3474	>5000	5003	86.7	>5000
NOS3	>5000	1160	>5000	113	12.7	>5000
NOS4	>5000	731	>5000	85	1.0	>5000
NOS5	>5000	1147	>5000	94	5.2	>5000
NOS6	>5000	926	>5000	918	38.7	>5000
NOS7	>5000	1082	>5000	63	3.9	>5000
ZENIOS	942	1285	>5000	141	22.2	2295

Table 2.9: Solving non-diagonal systems with ILUTP preconditioning.

	Arnoldi	Orthogonalized Arnoldi	Modified Arnoldi	Davidson	Harmonic Davidson
(NONE)	18	18	18	18	17
diagonal	9	10	9	28	13
SOR	6	15	5	34	19
ILU0	5	13	3	29	22
ILUTP	4	15	2	27	19

Table 2.10: Number of eigenvalue problems solved by each method.

	Arnoldi	Orthogonalized Arnoldi	Modified Arnoldi	Davidson	Harmonic Davidson
(NONE)	16	2	-	0	0
diagonal	0	0	0	28	0
SOR	0	0	0	34	0
ILU0	1	1	0	27	0
ILUTP	0	2	0	26	0

Table 2.11: Number of cases where the method reached convergence in the least amount of time.

NAME	N	comment
BCSSTM01	48	mass matrix
BCSSTM02	66	mass matrix, small oil rig
BCSSTM03	112	mass matrix
BCSSTM04	132	mass matrix, oil rig
BCSSTM05	153	mass matrix, transmission tower
BCSSTM06	420	mass matrix
BCSSTM08	1074	mass matrix, frame building (TV studio)
BCSSTM09	1083	mass matrix, square plate clamped
BCSSTM11	1473	mass matrix, ORE car (lumped masses)
BCSSTM19	817	mass matrix, part of a suspension bridge
BCSSTM20	485	mass matrix, frame within a suspension bridge
BCSSTM21	3600	mass matrix, clamped square plate
BCSSTM22	138	mass matrix, textile loom frame
BCSSTM23	3134	mass matrix, 3D building
BCSSTM24	3562	mass matrix, winter sports arena
BCSSTM25	15439	mass matrix, 76 story skyscraper
BCSSTM26	1922	mass matrix, reactor containment floor

Table 2.12: Diagonal RSA matrices from HB collection.

	Arnoldi		Orthogonalized		Davidson		Harmonic	
	MATVEC	time	MATVEC	time	MATVEC	time	MATVEC	time
BCSSTM01	25	0.08	25	0.08	11	0.07	25	0.24
BCSSTM02	25	0.08	25	0.08	17	0.12	25	0.28
BCSSTM03	95	0.34	81	0.30	72	0.74	82	1.23
BCSSTM04	25	0.11	81	0.31	36	0.32	25	0.32
BCSSTM05	25	0.13	45	0.19	25	0.27	25	0.34
BCSSTM06	911	9.02	1391	14.07	713	12.82	1231	31.86
BCSSTM08	1171	31.17	1374	35.26	1171	45.53	728	40.20
BCSSTM09	25	0.60	25	0.63	10	0.25	19	0.66
BCSSTM11	43	1.44	43	1.43	29	1.21	28	1.45
BCSSTM21	25	2.43	25	2.30	9	0.62	25	3.45
BCSSTM22	145	0.60	132	0.55	121	1.47	134	2.08

Table 2.13: Solving diagonal systems without preconditioning.

2.6 Summary

From comparing the Arnoldi method and the Davidson method, we observed that many eigenvalue solvers can fit into the frame work of a generic projection eigenvalue algorithm, see algorithm 2.3. From this generic algorithm we describe three variations. The five methods are mathematically equivalent to each other without precondition. If no preconditioner is available, the most efficient method among the five is the Arnoldi method because it is cheaper than others per iteration. However, compared to others, the bases it created is more likely to loss orthogonality which could lead to more matrix-vector multiplications being used to be compute the same solution. Even though it might take a few extra steps, if the matrix-vector multiplication is relatively cheap, the Arnoldi method may still take less time than others.

When a preconditioner is applied, the five methods behave quite differently. A good preconditioner can enhance the performance of all five methods we tested. However, in the tests we have conducted, the Davidson method can take advantage of preconditioning much better than other four methods. This trend is consistent in all preconditioners we have tried, diagonal scaling, SOR, ILU0, and ILUTP. Even when the preconditioner is exactly $(A - \lambda I)$, the five methods still produce different bases. As we have seen in the case of diagonal matrices with diagonal preconditioner, the Davidson method can still outperform the others

	Arnoldi	Orthogonalized	Modified	Davidson		Harmonic
	MATVEC	MATVEC	MATVEC	MATVEC	time	MATVEC
BCSSTM01	25	25	25	9	0.06	25
BCSSTM02	25	46	25	20	0.12	53
BCSSTM03	38	95	55	20	0.15	42
BCSSTM04	25	57	25	27	0.22	25
BCSSTM05	25	25	50	14	0.11	25
BCSSTM06	107	160	54	22	0.33	211
BCSSTM08	171	172	54	24	0.79	44
BCSSTM09	38	25	25	8	0.21	25
BCSSTM11	40	56	1607	20	0.84	38
BCSSTM19	120	172	55	25	0.63	1068
BCSSTM20	133	172	25	28	0.49	719
BCSSTM21	25	38	25	12	0.86	25
BCSSTM22	146	159	54	23	0.19	122
BCSSTM23	120	133	55	24	2.37	740
BCSSTM24	66	77	25	28	3.09	501
BCSSTM25	145	118	>5000	31	16.96	>5000
BCSSTM26	159	198	81	24	1.38	956

Table 2.14: Solving diagonal systems with diagonal scaling.

	IRA(25)		IRA(50)	
Name	MATVEC	time(sec)	MATVEC	time (sec)
662BUS	3729	46.8	2533	58.6
685BUS	2327	31.9	1379	33.6
BCSSTK02	233	1.1	179	1.1
BCSSTK05	801	3.5	665	4.5
BCSSTK09	1028	27.3	971	40.5
BCSSTK16	1601	352.1	1183	333.7
BCSSTK22	>5000	-	1754	10.2
BCSSTM10	329	9.5	314	13.6
BCSSTM27	3665	159.3	2307	138.7
GR3030	274	5.6	263	8.6
LUNDA	4285	17.4	2005	12.8
LUNDB	>5000	-	4342	27.4
NOS3	735	17.2	703	25.9
NOS4	233	0.9	182	1.1
NOS5	2879	28.8	1875	30.0
ZENIOS	82	6.0	94	9.9

Table 2.15: Solving the non-diagonal systems using ARPACK.

with exact preconditioners.

Through this study we have shown that preconditioning the projection methods for eigenvalue problems is an effective way of improving the performance of the methods. The tests with diagonal matrices showed that if a good preconditioner is available, the performance of all five methods can be significantly improved. However, even if with a week preconditioner, such as the diagonal scaling or SOR, the Davidson method can reach convergence on more problems. On those problem that can be solved without preconditioning, the Davidson method with preconditioning can often reduce the time and the number of matrix-vector multiplications used to compute the solutions.

Chapter 3

Preconditioning Schemes

From study of the five preconditioned projection methods for eigenvalue problems in the previous chapter, we see that preconditioning can significantly help reduce the number of matrix-vector multiplication and time of solving an eigenvalue problem. The Davidson method is especially effective in taking advantage of preconditioner. It is fairly simple to implement and very effective in many cases and works well with simple preconditioners. In many applications where the Davidson method is used, when a simple preconditioner is not sufficient, it is often possible to construct an effective preconditioner based on the characteristics of the problem. However there are also cases where aggressive preconditioners do not work well with the Davidson method [125, 126]. This and other issues discussed in section 3.1 have generated a great deal of interest in seeking new preconditioning schemes for eigenvalue algorithms. The Davidson preconditioning scheme can be viewed as an inexact Newton method for eigenvalue problems. In this chapter we extend on this theme and develop a number of preconditioning schemes that approximate Newton-type algorithms for eigenvalue problems. Similar to the Rayleigh Quotient Iteration (RQI), these Newton methods converge cubically to a simple eigenvalue of a symmetric matrix. In addition to their high convergence rate, the preconditioning schemes developed from them can avoid some of serious difficulties in the Davidson preconditioning scheme.

Section 3.1 of this chapter reviews the current state of the Davidson preconditioning, including its connection to Newton method and its weaknesses. Section 3.2 describes a number of Newton-type recurrences for eigenvalue problems and their relation to the Rayleigh Quotient Iteration. Section 3.3 introduces strategies of using these Newton-type methods as preconditioners and practical issues related to their implementations. Section 3.4 present a few numerical examples. A short summary is given in section 3.5.

3.1 Davidson preconditioning

When Davidson described his method in [30], he did not use the word *preconditioning*. Later on, many researchers noticed the resemblance between the technique used by Davidson and preconditioning techniques for solving linear systems [31, 80]. The diagonal scaling step of the Davidson method is call preconditioning. Since the diagonal scaling can be regarded as an approximation to $(A-\lambda I)^{-1}$. We refer to any technique that approximates $(A-\lambda I)^{-1}$ as a Davidson preconditioning scheme. Many techniques for preconditioning linear systems are now applied to the Davidson method to improve the convergence of eigenvalues. Preconditioning for eigenvalue problem is gradually being accepted as a practical tool. Theoretical study of this subject has just begun [16, 71, 77, 81, 133, 155].

Preconditioning for iterative linear system solver is a well studied research topic [9, 10, 119]. In order to speed up the convergence of an iterative procedure for linear system $Ax = b$, the procedure is usually applied on one of the following three preconditioned systems instead,

$$\begin{aligned} M^{-1}Ax &= M^{-1}b, \\ AM^{-1}(Mx) &= b, \\ M_L^{-1}AM_R^{-1}(M_Rx) &= M_L^{-1}b, \end{aligned}$$

where the first form is commonly referred to as left-preconditioning, the second right-preconditioning, and the third split preconditioning. One way of viewing preconditioning is that preconditioning attempts to make $(M^{-1}A)$, (AM^{-1}) or $(M_L^{-1}AM_R^{-1}$ close to the identity matrix because a linear system with the identity matrix is trivial to solve. Since the spectrum of $(M^{-1}A)$, (AM^{-1}) and $(M_L^{-1}AM_R^{-1}$ can differ from that of A arbitrarily, preconditioning for eigenvalue problem has to take on a different form. There are different ways of viewing preconditioning, for example, preconditioning to reduce condition number and preconditioning to increase stability, however, they are not directly relevant to eigenvalue problems either.

An eigenvalue algorithm like the Lanczos method or the Davidson method can be viewed to have two distinct components: one to build a basis and one to perform projection on the basis, see algorithm 2.3. In this setting, preconditioning is applied to the basis generation step of the algorithm, see algorithm 2.6. As shown in the previous chapter, the unpreconditioned Davidson method appends the current residual vector to the basis. The Davidson preconditioning scheme applies a preconditioner to the residual vectors in an attempt to improve the quality of the basis. The original Davidson method solves the following linear system for preconditioning

$$(M - \lambda I)z = r,$$

where M is the diagonal part of the matrix A, λ is the approximate eigenvalue, $r \equiv (A - \lambda I)x$ is the residual and x is the approximate eigenvector. Solving this linear system is "the preconditioning step", or simply "preconditioning". The matrix $(M - \lambda I)$ is called the preconditioner. The preconditioner for linear systems approximates A. The Davidson preconditioner approximates $(A - \lambda I)$. Because their similarities, any preconditioner for linear systems can also be adopted for eigen-systems [32, 77, 80]. However, using an approximation of $(A - \lambda I)$ as preconditioner has the following pitfalls most of which are pitfalls of the underlying Newton method for eigenvalue problems [36].

1. **Linear dependent basis** When M is very close to A, one would expect an good preconditioner based on experiences with the linear system solvers. However the above equation suggests that $z \approx x$, appending z to the current basis will cause the new basis to be almost linearly dependent. Usually, this means we have to perform extra reorthogonalization to maintain the orthogonality of the basis. If the difference between z and x is very small, the new basis vector generated will be dominated by round-off error which might impede convergence.

2. **Ill-conditioned iteration matrix** The matrix $(A - \lambda I)$ is ill-conditioned when the approximate eigenvalue λ is close to an actual eigenvalue. In [95], it was shown that the error from this ill-conditioned system does not hurt the convergence of inverse iterations. One might be tempted to adopt the result to the preconditioner. However, the argument used there can not be directly applied because the preconditioner is usually solved with an iterative solver or an approximate factorization, which only approximately solves $(A - \lambda I)z = r$. In addition, a linear system with an ill-conditioned matrix is generally harder to solve for an iterative method.

3. **Misconvergence** When the initial Ritz value is far from the desired eigenvalue, the Davidson method may converge to an eigenvalue that is close to the initial Ritz value rather than the desired one. There is no practical procedure that can guarantee the Lanczos method or the Davidson method will converge to the desired solution with or without preconditioning. However, this problem is much more serious with preconditioning than without.

4. **Indefinite iteration matrix** Usually λ is inside the spectrum of A, if M is a good approximation to A, then $(M - \lambda I)$ is indefinite. An indefinite system is harder to solve for a common preconditioner such as an iterative linear system solver or an incomplete factorization. An indefinite iteration matrix can cause an iterative solver to convergence slowly. An incomplete LU factorization procedure may fail on an indefinite matrix for it may encounter zero pivot.

5. **Complex shift** When the matrix is nonsymmetric, λ may be complex, the above preconditioned linear system is complex. This could demand either added storage and/or added complexity to the preconditioning procedure.

Among these five problems, we will not encounter the 5th one because we deal with Hermitian matrices only. We will address the other four problems with a strong emphasis on the first two.

There are several variations on the Davidson preconditioning scheme to overcome some of the difficulties mentioned above. The original preconditioning scheme in the Davidson method solves $(M - \lambda I)z = r$ where M is the diagonal part of A. One of the earliest modifications is from Olsen and colleagues which uses the following equation for preconditioning [90]

$$(M - \lambda I)z = r - \epsilon_O x. \tag{3.1}$$

This was shown to be an effective scheme if an appropriate ϵ_O is used. The original criteria for choosing ϵ_O was to make z orthogonal to x as this would avoid the first problem mentioned above. To achieve this, it is easy to show that we must have

$$\epsilon_O = \frac{x^T(M - \lambda I)^{-1}r}{x^T(M - \lambda I)^{-1}x}.$$

The idea of orthogonalizing the output of preconditioning against the Ritz vector appears to be a logical choice for avoiding the linear dependent basis problem. The Jacobi-Davidson preconditioning schemes can be thought as a different implementation to realize this intention. In order to solve equation (3.1), it is necessary to apply the preconditioner on both x and r which doubles the cost of preconditioning. Stathopoulos and co-authors later modified this scheme based on the correction equation

$$(A - (\lambda + \epsilon))d = -(A - (\lambda + \epsilon))x, \tag{3.2}$$

where $\lambda^* = \lambda + \epsilon$ and $x^* = x + d$ are the exact eigenvalue and the corresponding eigenvector [36, 133]. Based on this equation, the error in λ, ϵ, is used as ϵ_O in equation (3.1). This provides an inexpensive alternative to the original choice of ϵ_O.

For convenience of discussion, we will refer to the first modified residual as *the Olsen residual* and this later modification as *the Stathopoulos residual*. When referring to both of them without distinction, we will use the term *modified residual*. In general we will write them as,

$$s \equiv [A - (\lambda + \epsilon)I]x, \tag{3.3}$$

where the actual value of ϵ is chosen to either make it the Olsen residual or the Stathopoulos residual. Most often the later is used.

In equation (3.1), λ is used as the shift to the preconditioner M. It is a common choice. However there is no indication that it is an optimal choice. In [133], the authors proposed to use a biased estimate of the eigenvalue in the above preconditioner based on the correction equation (3.2). In order to define this biased estimate, recall that the error of a Ritz pair can be bounded as follows.

$$\epsilon \leq \begin{cases} \|r\|, & \|r\| \geq g, \\ \|r\|^2/g, & \|r\| < g, \end{cases} \tag{3.4}$$

where g is the gap between λ and the closest eigenvalue [51, 133]. When seeking the smallest eigenvalues, the biased estimate of λ^* is $\delta = \lambda - \epsilon$, where λ is the current Ritz value; when looking for the largest eigenvalues, $\delta = \lambda + \epsilon$. Using the definition of biased estimated of eigenvalue, the Stathopoulos residual can be written as $s = (A - \delta I)x$. In many cases, this biased eigenvalue estimate is closer to the exact eigenvalue than the Ritz value. Numerical tests have shown the biased shift to be more effective than λ. Tests also show that it is better in addressing the misconvergence problem in the above list. Some researchers have argued that $(M - \delta I)$ should be positive definite [81], using a biased shift in many cases will resulting a definite preconditioner. One limitation of this biased estimate idea is that it does not extend to complex eigenvalues easily.

After the above two modifications on the residual and the shift, the preconditioning equation can be rewritten as

$$(M - \delta I)z = s, \tag{3.5}$$

where δ is an overestimate of eigenvalue and s is the modified residual defined by equation (3.3). We note that this preconditioning equation is essentially an approximate form of the correction equation. Dongarra

and colleague have studied this subject from the perspective of improving accuracy of eigenvalues computed [34, 35, 36]. It was pointed out in [36] that directly solving the correction equation may also face the same difficulties we have listed above.

Recently, the Jacobi-Davidson method has attracted much attention [125]. The preconditioning step of the Jacobi-Davidson method can be written as

$$(I - xx^T)(A - \lambda I)(I - xx^T)z = r, \tag{3.6}$$

instead of equation (3.5). The matrix on the left-hand side is singular which make it unsuitable for many preconditioning techniques. However, a special advantage of this scheme is that when this linear system is solved using a Krylov iterative solver the result is orthogonal to the approximate eigenvector. In spirit, this can be regarded as a reincarnation of the Olsen scheme because it is a different way of making z orthogonal to x.

In this chapter we will only try to find one eigenpair. The subscripts to variables, λ_i, x_i, and r_i, are used to indicate that they are values produced at ith iteration. The initial values are denoted as λ_0 and x_0. For simplicity we will always assume that x_0 is a unit vector and λ_0 is the corresponding Rayleigh quotient, $\lambda_0 = x_0^T A x_0$.

3.2 Newton method for eigenvalue problems

The Newton method is a class of method that seeks a solution for $f(x) = 0$ by performing the following recurrence

$$x_{i+1} = x_i - \left(\frac{\mathrm{d}f}{\mathrm{d}x}\right)^{-1}_{x=x_i} f(x_i).$$

The derivative $\frac{\mathrm{d}f}{\mathrm{d}x}$ is known as the Jacobian or the Jacobian matrix which is usually denoted by J. for a continuous function f, if J is nonsingular and the starting point x_0 is close enough to the solution, this recurrence converges quadratically. An inexact Newton method is one that solves the equation $J(x_i - x_{i+1}) = f(x_i)$ inaccurately.

The Davidson preconditioner solves an Newton equation approximately. For this reason, given a good enough preconditioner, the Davidson method could converge quadratically [26, 31, 32, 81]. From our description in the previous section, it is clear that the Davidson preconditioning scheme can be viewed as an approximate form of the correction equation. If the exact eigenvalue λ^* is known, finding the corresponding eigenvector can be achieved by applying the following Newton recurrence to solve $r = (A - \lambda^* I)x = 0$,

$$x_{i+1} = x_i - (A - \lambda^* I)^{-1} r_i, \tag{3.7}$$

where $r_i = (A - \lambda^* I)x_i$. Note that this equation is equivalent to the correction equation (3.2). Because the Davidson preconditioning approximates the correction equation, it is an inexact form of the above Newton recurrence.

If equation (3.7) is actually solved accurately, the solution is zero, $x_{i+1} = x_i - (A - \lambda^* I)^{-1}(A - \lambda^* I)x_i = 0$. This is a different manifestation of linear dependent basis problem described on page 30. Since λ^* is an exact eigenvalue, the Jacobian matrix $(A - \lambda^* I)$ can't be inverted, so the above Newton recurrence is not well defined. To correct this, the following recursion is considered instead of equation (3.7),

$$x_{i+1} = x_i - (A - \lambda^* I)^{+} r_i, \tag{3.8}$$

where the superscript + indicates pseudoinverse [51]. By definition of pseudoinverse, $AA^+A = A$. The above equation is still equivalent to the correction equation.

$$\begin{aligned}(A - \lambda^* I)(x_i - x_{i+1}) &= (A - \lambda^* I)(A - \lambda^* I)^{+} r_i \\ &= (A - \lambda^* I)(A - \lambda^* I)^{+}(A - \lambda^* I)x_i \\ &= (A - \lambda^* I)x_i \equiv r_i.\end{aligned}$$

This Newton iteration removes the component orthogonal to x^* from x_i. If the initial guess x_0 is not orthogonal to x^*, x_1 computed by equation (3.8) is a non-trivial solution of $r = 0$, in other word, x_1 is the wanted eigenvector.

Through the above argument, we see that solving the correction equation using pseudoinverse, see equation (3.8), has significant advantage over the alternative, see equation (3.7). In practice, we don't know the exact eigenvalue, therefore $(A - \lambda I)$ is not singular, thus $(A - \lambda I)^+ = (A - \lambda I)^{-1}$. We continue to face the same problems mentioned above. Since the pseudoinverse of a matrix is different from the inverse only if the matrix is singular, we could force the matrix to be singular as what is done in the Jacobi-Davidson method. Note that the Jacobi-Davidson method was not derived this way, but this is a valid interpretation. In the Jacobi-Davidson method, a Krylov subspace method is used to solve the preconditioning equation (3.6). The Krylov subspace method approximates pseudoinverse by computing a solution of in the range of the iteration matrix [10]. One difference between equation (3.7) and equation (3.8) is that the later one generates a correction that is orthogonal to the exact eigenvector x^*, in other word, the correction is in the range of the iteration matrix. Both the Olsen preconditioning scheme and the Jacobi-Davidson preconditioning scheme are successful because they generate orthogonal corrections. An alternative strategy to address the same issue is to find a Newton method with a nonsingular Jacobian matrix. Searching for a well behaved Newton method is the main objective of this section.

Newton based schemes for eigenvalue problems have been investigated from many different points of view. Peters and Wilkinson showed a Newton method for eigenvalue problems which solves an augmented $(n + 1) \times (n + 1)$ system [98]. The eigenvalue problem is treated as a unconstrained minimization problem in this case. As shown above, the correction equation can be regarded as a Newton method for eigenvalue problems. One of the main uses of the correction equation is to improve the accuracy of computed eigenpairs. Numerous variations of the correction equation have been studied for this purpose in [34, 35, 36]. Newton method is also often used as a part of the Homotopy method for eigenvalue problems [2, 3, 89]. In this case the Newton method used is very close to what is proposed in [98]. The Jacobi-Davidson method is regarded as a form of Newton's method in the space perpendicular to x [38, 126]. Due to this unique construction, it avoids some of the pitfalls of the original Davidson preconditioning scheme. There are a number of eigenvalue methods that are based on Newton method for eigenvalue problems, for example, the trace minimization method [122] and the inflated inverse iteration [47]. Since we are interested in mimic the original Davidson preconditioning scheme, we will not directly adopt these methods as preconditioners though the idea may deserve some attention.

In this section we will revisit the augmented Newton scheme proposed in [98]. It is slightly reformulated so that the eigenvector is scaled in 2-norm. Three variations are derived from this augmented Newton method, one of which can lead back to the Olsen scheme and all three are equivalent to the Rayleigh quotient iteration in exact arithmetic. The eigenvalue problem can also be formed as an constrained optimization problem in which case the Newton method for constrained optimization proposed by Tapia can be applied [142, 143, 144]. We will describe this constrained minimization problem and discuss its properties. In addition, we will present the Jacobi-Davidson iteration as a Newton method for eigenvalue problems. Coincidentally, it is also equivalent to the Rayleigh quotient iteration.

In this section we will use the subscript i to indicate an arbitrary iteration. If a subscript is missing from x, λ, or r, it is implied that they refer to x_i, λ_i, or r_i.

3.2.1 Rayleigh quotient iteration

The Rayleigh Quotient Iteration (RQI) is a very effective method for finding one eigenvalue and its eigenvector [51, 93]. Its asymptotic convergence rate is cubic for simple eigenvalues of symmetric eigenvalue problems. Unless the eigenvalue is known, no algorithm can converge to an eigenvector with higher speed. It is also very simple to describe, this basic iteration can be summarized in one line,

$$x_{i+1} = \frac{(A - \lambda_i I)^{-1} x_i}{\|(A - \lambda_i)^{-1} x_i\|}, \quad \lambda_i \equiv x_i^T A x_i, \tag{3.9}$$

where x_0 is a unit vector.

Note that the Rayleigh quotient iteration has most of the problems we have observed for preconditioners, see page 30, because the iteration matrix used in RQI is the same as in the Davidson preconditioning. Our

i	λ_i	$\|r_i\|$	κ_i
1	0.01361736175	0.003830979028	523.5
2	0.01427847652	0.000357647692	1109
3	0.01430356647	2.468630821e-06	3.029e+04
4	0.01430356864	2.006922521e-12	3.503e+08
5	0.01430356864	9.544854661e-17	5.705e+16

Table 3.1: Ritz values and residual norms produced at each step of RQI while computing the smallest eigenvalue of PLAT362.

i	λ_i	$\|r_i\|$	κ_i
1	-90685.22644	911704.3057	762
2	2500.224244	34828.83139	2.567e+04
3	0.211851563	672.661174	2.807e+05
4	0.2109448726	0.01285108761	4.407e+11
5	0.2106508436	0.001874826371	4.598e+11
6	0.2097380738	0.001081499539	5.698e+11
7	0.2094248973	0.000100055371	2.218e+12
8	0.2094224436	6.79586972e-08	2.853e+14
9	0.2094224436	8.26389478e-09	2.134e+18

Table 3.2: Ritz values and residual norms from RQI while computing the smallest eigenvalue of EX2.

main interest here is not to study the Rayleigh quotient iteration but to use it for comparisons against the Newton methods to be discussed. Because we will often refer to its iteration matrix, we will name it J_R, $J_R \equiv A - \lambda I$. We will also regard it as the iteration matrix for the Newton iteration described in equation (3.7) since λ^* is not known before convergence.

To test the Newton methods we will describe, we apply them on a test matrix called PLAT362. It is a small symmetric positive definite matrix, see table 2.2 for the size and the origin of the matrix. All of the eigenvalues for PLAT362 are in pairs, i.e., they are doublets. The smallest one is about 3.55×10^{-12}, and the largest one is 0.774. We set the tolerance on the residual norm to be 10^{-12} so that eigenvalues can be accurately resolved. Our intention here is to seek the smallest eigenvalue and the corresponding eigenvector starting with the initial guess $(1, 1, \ldots, 1)^T$. This experiment is conducted on a SPARC[1] 10 workstation using MATLAB[2] which uses 64-bit arithmetic for floating-point operations. The unit round-off error is 2.2×10^{-16}. For our purpose, the unit round-off error can use understood as follows. If the largest eigenvalue in absolute value is λ_{max}, any eigenvalue with absolute value less than $|2.2 \times 10^{-16} \lambda_{max}|$ is indistinguishable from zero. The condition number for a Hermitian matrix can be defined as $\kappa = \frac{|\lambda_{max}|}{|\lambda_{min}|}$ where λ_{max} and λ_{min} are the largest and smallest eigenvalue in absolute value. In 64-bit arithmetic, any matrix with condition number greater than 4.5×10^{15} could be considered as practically singular. Note that finding the smallest eigenvalues of PLAT362 is relatively hard for the Davidson method.

In a number of cases, we will also use a second test matrix. This second test matrix is EX2 which is a finite element matrix for a fully coupled Navier-Stokes equation. It is generated from solving the second example problem in the FIDAP package[3]. It is a symmetric matrix with only simple eigenvalues ranging from -7×10^8 to 3×10^6. The matrix size is 441×441. The largest negative eigenvalue is -48501, the smallest positive eigenvalue is 0.068. It is chosen because its unusual spectrum where there are 28 will separated negative eigenvalues and about 160 eigenvalues between 0 and 1. The conditioner number is about 10^{10}. Again we use the vector $(1, 1, \ldots, 1)^T$ as the initial guess. For this test matrix we iterate till the residual norm can no longer be reduced.

[1] SPARC is a trade of Sun microsystem.
[2] MATLAB is a trademark of MathWorks.
[3] FIDAP is a trademark of Fluid Dynamics International.

Table 3.1 show the λ_i, $\|r_i\|$ from solving PLAT362 using RQI. In this table, the first column is the iteration number i, the second and the third column are the Ritz values λ_i and the residual norms $\|r_i\|$, the fourth column is the condition number of the iteration matrix, $\kappa_i = \kappa(J_R)$. RQI did not converge to the smallest eigenvalue in this case. However it does converge very quickly to an eigenvalue close to the initial Ritz value, λ_0. One important observation to note here is the trend of κ. As iteration progresses, it quickly grows to practically infinity for 64-bit arithmetic. For later comparisons, we also applied RQI on EX2. The results is shown in table 3.2. Note that in this case, RQI converges to an eigenvalue within the large cluster around zero.

3.2.2 Augmented Newton method

One way of formulating the eigenvalue problem is to express it as finding a zero residual with (λ, x) pair as the independent variable [47, 89, 98]. This formulation can be described as seeking a solution to the following quadratic equation

$$\begin{cases} (A - \lambda I)x &= 0, \\ -\frac{1}{2}x^T x + \frac{1}{2} &= 0. \end{cases} \tag{3.10}$$

The original form of this Newton method is presented in [98]. Many researchers have noticed the scaling scheme in this paper was not practical, different variations have been used [47]. The scheme shown here is natural for 2-norm. This is a unconstrained quadratic problem. Given an initial guess (λ_0, x_0), the Newton iteration can be described as follows

$$\begin{pmatrix} x_{i+1} \\ \lambda_{i+1} \end{pmatrix} = \begin{pmatrix} x_i \\ \lambda_i \end{pmatrix} - J_A^{-1} \begin{pmatrix} (A - \lambda_i I)x_i \\ -\frac{1}{2}x_i^T x_i + \frac{1}{2} \end{pmatrix}, \tag{3.11}$$

$$J_A = \begin{pmatrix} A - \lambda_i I & -x_i \\ -x_i^T & 0 \end{pmatrix}. \tag{3.12}$$

We refer to this scheme as the augmented Newton recurrence because it solves a linear system whose size is larger than the size of original eigenvalue problem. The validity of this Newton scheme has been established in [98]. However we will still repeat part of the argument mainly to introduce the techniques used because they will be used again for other Newton schemes. First, let's give two simple definitions which will be used in a couple of places later.

If (λ, x) is an exact eigenpair, and λ is a simple eigenvalue, we will call (λ, x) a simple eigenpair. If (λ, x) is an eigenpair, the solution for the following equation

$$(A - \lambda I)y = x, \tag{3.13}$$

is called a principle vector of grade 2 corresponding to λ, see section 39 of [154], and λ corresponds to at least a quadratic elementary divisor of the characteristic polynomial of A, see section 9 of [154]. Note that x may be arbitrarily scaled in equation (3.13) even though we prefer a unit vector as an eigenvector. A basic fact about the principle vector is that *if λ is a simple eigenvalue, there is no grade 2 or higher principle vector*. This fact is used immediately in the proof of the following lemma.

Lemma 3.1 *If (λ, x) is a simple eigenpair, the Jacobian matrix J_A for the augmented Newton recursion is nonsingular.*

For a proof of this lemma, see section 4 of [98]. Assuming there is a vector $\binom{y}{\alpha}$ such that

$$J_A \begin{pmatrix} y \\ \alpha \end{pmatrix} = 0, \quad y \in \mathbf{R}^n, \quad \alpha \in \mathbf{R},$$

then both y and α must be zero. According to the definition of J_A, see equation (3.12), y and α must satisfy the following equations,

$$\begin{aligned} (A - \lambda I)y &= \alpha x, \\ x^T y &= 0. \end{aligned}$$

i	λ_i	$\|r_i\|$	κ_i
1	0.02797189896	1.610024501	690.6
2	0.022194266	0.4459502359	1582
3	0.05259548648	2.64641648	1362
4	0.03122099412	0.5640436082	1.344e+04
5	0.03496170267	0.4255608969	1681
6	0.03595909358	0.2368364335	1376
7	0.04138058562	0.3087099513	1300
8	0.03743277517	0.09756956524	3171
9	0.05383252242	0.5691998046	2177
10	0.03957117095	0.07630243347	1560
11	0.03748476578	0.01478631945	768.5
12	0.03718381556	0.006896535548	1829
13	0.03693657348	0.0001830158864	3965
14	0.03692985166	8.291242081e-07	1.481e+05
15	0.03692982107	3.566089834e-12	3.268e+07
16	0.03692982107	3.980562891e-11	7.599e+12
17	0.03692982107	6.599374204e-14	6.807e+11

Table 3.3: The intermediate results from the augmented Newton recurrence when computing the smallest eigenvalue of PLAT362.

i	λ_i	$\|r_i\|$	κ_i
1	305675.0787	1084020.582	1.740e+15
2	18112.55342	165792.3299	1.840e+14
3	-10808.31223	30650.49007	1.736e+13
4	-2704.146524	3329.024029	3.633e+12
5	-194.3086005	199.2559124	1.620e+12
6	-0.2278873939	1.108173596	1.354e+11
7	0.134302848	0.1432648279	2.197e+09
8	0.07799833383	0.2413701758	1.112e+12
9	0.04437308049	3.650974905	9.400e+10
10	0.05875740161	0.8677481763	1.508e+10
11	0.06520476412	0.1461848604	2.186e+10
12	0.06772453288	0.0086436044	2.737e+10
13	0.0680408643	3.913476664e-05	3.036e+10
14	0.06804357153	4.576369965e-09	3.078e+10

Table 3.4: The intermediate results from applying the augmented Newton recurrence on the EX2 test problem.

If y is not zero, either y is another eigenvector corresponding to λ when α is zero, or y is a principle vector corresponding to λ when α is not zero, in either case λ is not a simple eigenvalue which contradicts the assumption.

This lemma has two implications. First, because equation (3.10) is quadratic in (λ, x), it guarantees that given a good initial guess, the Newton recurrence described by equation (3.11) will converge quadratically. Second, the iteration matrix, J_A, is nonsingular. This is important because we are searching for a Newton method for eigenvalue problem with nonsingular Jacobian matrix. We have found our first one.

In our implementation of this method, we only take an eigenvector initial guess as input. This initial guess is normalized to produce the x_0 for equation (3.11). The initial eigenvalue is taken to be the corresponding Rayleigh quotient, $\lambda_0 = x_0^T A x_0$. Table 3.3 shows the convergence process of this method applied to the test

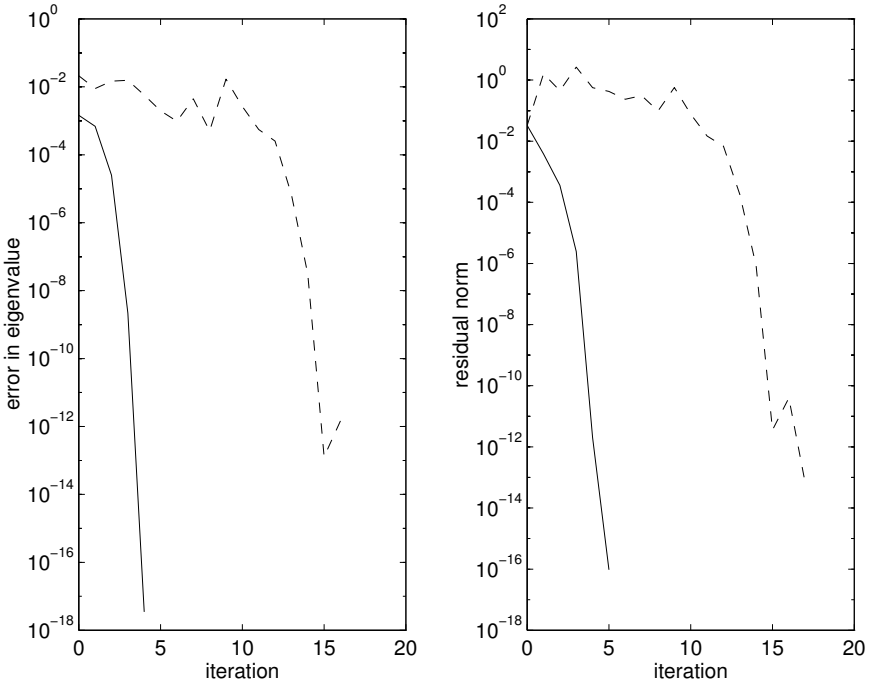

Figure 3.1: Convergence histories of RQI (solid line) and the augmented Newton recurrence (dashed line) applied on the PLAT362 test problem.

matrix PLAT362. We see that the condition number of J_A stay relatively small compared to table 3.1. It also appears that the augmented Newton method takes more steps to converge than RQI which could be attributed to the fact that the approximate eigenvectors are not normalized as in RQI. In [98], a version of this augmented Newton method with a different normalization was shown to be identical to RQI. Later in this section an normalized version of the about augmented Newton recurrence will also be shown to be identical to RQI. In the case of RQI, eigenvalue approximations are Rayleigh quotients which are always contained inside the spectrum of the matrix. However there is no such restriction on the eigenvalue approximation generated by the augmented Newton method. The eigenvalue approximations by the augmented Newton recurrence is clearly different from that by RQI, see tables 3.1 and 3.3.

Figure 3.1 shows the convergence history of both RQI and the augmented Newton recurrence. The error in eigenvalue is estimated as $|\lambda_i - \lambda^*|$ where λ^* is replaced with the approximation at the last step. This error estimate leaves the error at the last iteration as zero which is not plotted in figure 3.1. From the figure, it appears that the augmented Newton recurrence spends a significant number of step in searching for an eigenpair to converge to. Once it has find a destination, it appears to converge almost as fast as RQI.

Table 3.4 shows the intermediate results while solving the EX2 test problem. In this case, the conditioner number of J_A stays fairly large. However, it shows an overall decreasing trend. Near convergence, the condition number settles down near the condition number of the original matrix. This contrasts the trend in table 3.2 where the condition number of the iteration matrix grows as convergence approaches. This test shows that even though, J_A is not singular around the exact solution, it may be ill-conditioned far from convergence. Another possible short-coming of this method is that the approximate eigenvalues are not Rayleigh quotients. Because the Rayleigh quotient has many optimality conditions [95], it would be preferable if the eigenvalue approximation is computed as Rayleigh quotient.

3.2.3 Variations of the augmented Newton method

Because the augmented Newton recurrence solves a linear system of a larger size, it may not be desirable to work with the augmented system directly. One solution to this problem is to factor J_A and write the update

expressions for λ and x separately. When J_R is nonsingular, the block LU factorization of J_A is as follows

$$J_A = \begin{pmatrix} I & 0 \\ -x^T J_R^{-1} & 1 \end{pmatrix} \begin{pmatrix} J_R & -x \\ 0 & -x^T J_R^{-1} x \end{pmatrix}.$$

Note that $J_R = A - \lambda I$ and the variables λ, x and r are assumed to have subscript i if no subscript is present. Defining $\gamma_i = (1 - x_i^T x_i)/2$, $r_i = (A - \lambda_i I)x_i$, equation (3.11) can be transformed as follows,

$$x_{i+1} = x_i - J_R^{-1}\left(r_i - \frac{\gamma_i x_i + x_i x_i^T J_R^{-1} r_i}{x_i^T J_R^{-1} x_i}\right) \tag{3.14}$$

$$\lambda_{i+1} = \lambda_i + \frac{\gamma_i + x_i^T J_R^{-1} r_i}{x_i^T J_R^{-1} x_i} \tag{3.15}$$

Since the Newton iteration only requires x_i to compute x_{i+1}, the above recurrence can be slightly modified by scaling x_{i+1} to norm one which ensures γ_i is always zero. With this modification, equation (3.14) becomes

$$x_{i+1} = \frac{x_i - J_O^{-1} r_i}{\|x_i - J_O^{-1} r_i\|} \tag{3.16}$$

$$J_O^{-1} = (A - \lambda_i I)^{-1}\left(I - \frac{x_i x_i^T (A - \lambda_i I)^{-1}}{x_i^T (A - \lambda_i I)^{-1} x_i}\right).$$

We can make two interesting observations about this equation. One, equation (3.16) resembles a Newton iteration with the Jacobian matrix being J_O. Second, the difference between x_{i+1} and x_i computed by equation (3.16) looks very similar to the Olsen preconditioning scheme, equation (3.1), where $(M - \lambda I)^{-1}$ in equation (3.1) is replaced with $(A - \lambda I)^{-1}$ in equation (3.16). This establishes a connection between the Olsen preconditioning scheme with the augmented Newton method for eigenvalue problem which provides a new interpretation of the Olsen preconditioning scheme.

Another point we should note about the matrix J_O is that we only know what is J_O^{-1}. Because J_O^{-1} is singular, $J_O^{-1} x_i = 0$, we choose to not define what is J_O itself. Since we have given the explicit form for the inverse of the Jacobian matrix in this case, the Newton recurrence can always be carried out as long as $x_i - J_O^{-1} r_i$ is not zero. Let $x_O = x_i - J_O^{-1} r_i$, the following is true.

$$(A - \lambda_i I)x_O = r_i - \left(I - \frac{x_i x_i^T (A - \lambda_i I)^{-1}}{x_i^T (A - \lambda_i I)^{-1} x_i}\right) r_i = -\frac{x_i}{x_i^T (A - \lambda_i)^{-1} x_i}.$$

This indicates that x_O is well defined if λ_i is not an exact eigenvalue and $x_i^T (A - \lambda_i)^{-1} x_i$ is not zero.

We have neglected to mention how to compute λ_{i+1} in the recurrence described in equation (3.16). Now that we use equation (3.16) to compute x_{i+1} rather than equation (3.14), we can no long use equation (3.15) to update λ as in the augmented Newton recurrence. The alternative is to compute λ_{i+1} as a Rayleigh quotient, i.e., $\lambda_{i+1} = x_{i+1}^T A x_{i+1} / x_{i+1}^T x_{i+1}$. According to the properties of Rayleigh quotient, this new λ_{i+1} is not worse than that given by equation (3.15). This modified scheme should converge at least as fast as the augmented Newton scheme. Because the resemblance to the Olsen's scheme for preconditioning, this recurrence is named the Newton-Olsen recurrence. The Jacobian matrix J_O for this scheme looks complicated. However due to the following connection with the Rayleigh quotient iteration, we know that the Newton-Olsen recurrence converges fast.

Lemma 3.2 *Given the same initial unit vector for both the Newton-Olsen recurrence and the Rayleigh quotient iteration, if $A - \lambda_i I$ neve becomes singular and $x_i^T (A - \lambda_i I)^{-1} x_i$ never becomes zero, the two recurrences produce the same result.*

Proof. At step i, assume x_i produced by both recurrences are the same. The subscript i is dropped in the rest of this proof because we will not refer to any other step. Let x_R and x_O denote the solution of the Rayleigh quotient iteration and the Newton-Olsen recurrence before scaling, i.e.,

$$x_R = (A - \lambda I)^{-1} x$$

$$x_O = x - J_O^{-1} r = x - (A - \lambda I)^{-1}\left(I - \frac{x x^T (A - \lambda I)^{-1}}{x^T (A - \lambda I)^{-1} x}\right) r.$$

i	λ_i	$\|r_i\|$	κ_i
1	0.01361736175	0.003830979028	1.153e+16
2	0.01427847652	0.000357647692	4.471e+15
3	0.01430356647	2.468630821e-06	1.74e+15
4	0.01430356864	2.013442306e-12	1.165e+16
5	0.01430356864	8.487764927e-11	3.19e+18
6	0.01430356864	6.952611828e-11	4.76e+18

Table 3.5: The intermediate results of the Newton-Olsen recurrence applied on the PLAT362 test problem.

i	λ_i	$\|r_i\|$	κ_i
1	0.01361736175	0.003830979028	690.6
2	0.01427847652	0.000357647692	1457
3	0.01430356647	2.468630821e-06	3.985e+04
4	0.01430356864	2.006918586e-12	4.609e+08
5	0.01430356864	1.685011223e-16	2.118e+16

Table 3.6: Intermediate results from the normalized augmented Newton recurrence when applied on PLAT362.

Using the definition of $r = (A - \lambda I)x$ and the fact that $x^T x = 1$, we can simplify the equation for x_O as follows

$$x_O = x - x + (A - \lambda I)^{-1}x / x^T (A - \lambda I)^{-1} x = x_R / x^T (A - \lambda I)^{-1} x.$$

If $x^T (A - \lambda I)^{-1} x \neq 0$, x_O and x_R only differ by a finite scalar constant. Since both RQI and the Newton-Olsen recurrence scale their approximate eigenvectors to norm 1, it is clear that they give the same answer. □

This lemma indicates that RQI and the Newton-Olsen method are equivalent to each other if $(A - \lambda_i I)$ is never singular and $x_i^T (A - \lambda_i I)^{-1} x_i$ is never zero. However, when $(A - \lambda_i I)$ is singular, (λ_i, x_i) is an exact eigenpair. Whether it is the desired one or not, there is no reason to continue the current recurrence. If the desired one is found we should stop, else a new initial guess is need to start another recurrence. When (λ, x) is close to a simple eigenpair, there is no solution to the following equations, $(A - \lambda I)y = x$, $y^T x = 0$. Thus $x_i (A - \lambda_i I)^{-1} x_i$ can not be zero. In summary, if λ_i approaches a simple eigenvalue, then RQI and the Newton-Olsen recurrence are equivalent to each other.

When near the exact solution, $x_R \approx x/(\lambda^* - \lambda)$, which is poorly scaled. The same is not true for x_O. Because $x^T (A - \lambda I)^{-1} x \approx 1/(\lambda^* - \lambda)$,

$$x_O \approx x. \tag{3.17}$$

Intuitively, the Rayleigh quotient iteration works to amplify the x^* component in x_i, while the Newton-Olsen scheme works to remove components orthogonal to x_* from x_i. Numerically, this suggests that x_O is a better behaved vector than x_R even though there are theoretically equivalent. However, this benefit is not realized because J_O is very ill-conditioned in practice.

Table 3.5 shows the Newton-Olsen method applied to the test matrix PLAT362. It appears that the eigenvalue is converging to the same one as RQI. Unfortunately the residual norm did not decrease below 10^{-11}. This is due to the fact that multiplying J_O^{-1} by a vector requires computing solving with $A - \lambda_i I$ twice. Significant error can accumulate during this computation.

Since J_A is well conditioned near convergence, instead of using equation (3.16) to compute x_{i+1}, we could use equation (3.11) and scale x_{i+1} after every step. Following the Newton-Olsen recurrence, we can compute the eigenvalue approximation λ_{i+1} as a Rayleigh quotient instead of using the result from equation (3.11). This is a normalized augmented Newton method since the intermediate approximate solutions of the eigenvector are scaled to have unit norm. In normalized augmented Newton scheme, the iteration matrix is still J_A which is known to be nonsingular near convergence. Theoretically, the normalized augmented Newton

i	λ_i	$\|r_i\|$	κ_i
1	-90685.22644	911704.3057	1.740e+15
2	2500.224244	34828.83139	7.433e+13
3	0.2118515631	672.661174	1.877e+12
4	0.2109448739	0.01285108732	1.011e+12
5	0.2106508463	0.00187482772	2.201e+12
6	0.2097380766	0.001081503671	5.322e+11
7	0.2094248973	0.000100056939	1.739e+12
8	0.2094224437	6.722151431e-08	7.831e+12
9	0.2094224435	3.157933201e-09	8.052e+12

Table 3.7: Intermediate results from the normalized augmented Newton recurrence when applied on EX2.

method is exactly the same as the Newton-Olsen scheme. Numerically, this normalize augmented Newton recurrence should be better than the Newton-Olsen scheme since the matrix J_A has smaller condition number than J_O.

Table 3.6 shows the results for this normalized augmented Newton recurrence. We can see that the condition numbers are must smaller than those shows in table 3.5 which are produced with the Newton-Olsen recurrence. This method not only is theoretically equivalent to RQI, it also produces the exact same eigenvalue and residual norm for most steps. The residual norms differ at the last step only because RQI's iteration matrix is too ill-conditioned, and the round-off errors become significant. Note that the condition number of J_A grows at the same pace as that of J_R, see table 3.1. This is because we only prove that J_A is well behaved if the eigenvalue is simple. In this case, we know the eigenvalue is a doublet, J_A is not guaranteed to be well conditioned. The numbers in table 3.6 indicate that J_A becomes singular as the eigenvalue converges.

Table 3.7 shows the results of applying the normalized augmented Newton method on the other test matrix EX2. Since all eigenvalues are simple in this case, the condition number of J_A is not too much larger than the condition number of the original matrix.

The Jacobian matrix used in equation (3.16) can be approximated as follows

$$J_O^{-1} \approx (A - \lambda I)^{-1} \left(I - \frac{\alpha x x^T (A - \lambda I)^{-1}}{1 + \alpha x^T (A - \lambda I)^{-1} x} \right),$$

where α is a scalar constant. When $|\alpha x^T (A - \lambda I)^{-1} x|$ is much larger than 1, the above approximation is an accurate one, in which case $J_O \approx A - \lambda I + \alpha x x^T$ [51, the Sherman-Morrison formula]. This approximate form of the Jacobian matrix is simpler than the original definition of J_O which should make a more viable option as a preconditioner. This inexact Jacobian matrix is J_R with approximate Wielandt deflation [116, 154]. We can use this approximation to define another well behaved Newton recurrence.

Define

$$J_I = A - \lambda_i I + x_i x_i^T. \tag{3.18}$$

The following is true.

Lemma 3.3 *1. If (λ_i, x_i) is a simple eigenpair of A, J_I is nonsingular.*

2. If λ_i is an eigenvalue with multiplicity greater than 1, the matrix J_I is singular. The matrix J_I's null space should be one dimension less than that of $(A - \lambda_i I)$.

3. If the matrix $(A - \lambda_i I)$ is nonsingular and $x_i^T (A - \lambda_i I)^{-1} x_i$ is not -1, the matrix J_I is nonsingular.

Proof.

1. If (λ_i, x_i) is a simple eigenpair, the spectrum of J_I is the spectrum of A shifted by $-\lambda_i$ except the zero eigenvalue of $(A - \lambda_i I)$ is moved to 1. Since λ_i is a simple eigenvalue of A, $(A - \lambda_i I)$ has only one zero eigenvalue. By construction, J_I has moved this zero eigenvalue away from zero.

i	λ_i	$\|r_i\|$	κ_i
1	0.01361736175	0.003830979028	691.1
2	0.01427847652	0.000357647692	1457
3	0.01430356647	2.468630821e-06	3.985e+04
4	0.01430356864	2.006917624e-12	4.609e+08
5	0.01430356864	2.769830393e-17	2.294e+16

Table 3.8: The results from the inflated Newton recurrence applied on PLAT362.

2. If the dimension of the null space of $(A - \lambda I)$ is more than one, the construction of J_I only moves the dimension parallel to x_i. The matrix J_I has a null space of smaller dimension than $(A - \lambda_i I)$.

3. If the matrix $(A - \lambda_i I)$ is nonsingular, we can try to solve the following equation to identify the null space of J_I.

$$(A - \lambda_i I)y + (x_i^T y)x_i = 0.$$

The above equation appears to have the following solutions, $y = \alpha(A - \lambda_i I)^{-1}x_i$ where α is a scale constant. When we substitute this expression of y into the above equation, the following is the outcome.

$$\alpha x_i(1 + x_i^T(A - \lambda_i I)^{-1}x_i) = 0.$$

If $x_i^T(A - \lambda_i I)^{-1}x_i$ is not -1, the above equation can only be satisfied with $\alpha = 0$. This indicates that only $y = 0$ can satisfy equation $J_I y = 0$. In other word, J_I is nonsingular.

□

From this proof, we see that when (λ_i, x_i) is not an exact eigenpair, J_I can be singular if $x_i^T(A-\lambda_i I)^{-1}x_i = -1$. By definition, x_i is a unit vector, which is located on a n-dimensional unit sphere. If the surface defined by $x_i^T(A - x_i^T A x_i I)^{-1}x_i = -1$ intersects this n-dimensional unit sphere, then for those matrices, J_I may be singular when x_i fall on the intersections. In short, there are only very special values of x_i can make J_I singular.

When J_I is nonsingular and $J_I x_i \neq r_i$, the following recurrence is a valid Newton method for eigenvalue problem,

$$x_{i+1} = \frac{x_i - J_I^{-1}r_i}{\|x_i - J_I^{-1}r_i\|}. \tag{3.19}$$

Note that the condition of $J_I x_i \neq r_i$ is satisfied because x_i is a unit vector, $\lambda_i = x_i^T A x_i$ and $r_i = (A-\lambda_i I)x_i$.

Because of the connection with the Wielandt deflation, this recurrence is called the inflated Newton recurrence. We could have defined J_I as $A - \lambda I + \alpha x x^T$. However because of a lack of practical way to choose α we simply set it to 1 in equation (3.18). Here are some suggestions for choosing α if appropriate information is available. First, $\alpha = 0$ is not a good choice. One good choice is to pick a number within the spectrum of J_R, for example, $\alpha = \lambda_0 - \lambda^*$. If an iterative solver like CG is used to solve equation (3.19), it is good to place α at the center of a large cluster in J_R's spectrum. This will improve the convergence of the iterative solver. The matrix J_I is the same as the matrix for the inflated inverse iteration method shown in [47, equation (5.13)]. The derivation of the inflated inverse iteration was quite different from what is shown here, but it is also developed from the Newton method of [98].

From the proof of lemma 3.2, it is easy to see that the following is true.

Lemma 3.4 *Given the same unit vector x to both the inflated Newton recurrence and the Rayleigh quotient iteration, if $A - \lambda I$ is nonsingular and $1 + x^T(A - \lambda I)^{-1}x$ is nonzero, the two recurrence produce the same result.*

Table 3.8 shows the results of applying the inflated Newton recurrence on the test matrix PLAT362. The results of EX2 is shown in table 3.9. The inflated Newton recurrence is a fairly stable scheme, the condition number of J_I is similar to that of the normalized augmented Newton method, see table 3.6. Both J_A and J_I

i	λ_i	$\|r_i\|$	κ_i
1	-90685.22644	911704.3057	762
2	2500.224244	34828.83139	2.567e+04
3	0.2118515633	672.661174	2.807e+05
4	0.2109448763	0.01285108116	1.165e+12
5	0.2106508532	0.001874830432	2.23e+12
6	0.2097380827	0.001081513364	5.322e+11
7	0.2094248975	0.0001000607567	1.739e+12
8	0.2094224435	7.269909007e-08	7.831e+12
9	0.2094224435	2.276624783e-08	8.052e+12

Table 3.9: The results from the inflated Newton recurrence applied on EX2.

becomes singular when the eigenvalue computed is not simple, see table 3.6 and 3.8. When the eigenvalue is simple as in the EX2 case, both the normalized augmented Newton recurrence and the inflated Newton recurrence have similar condition number for their iteration matrices near convergence, see table 3.7 and 3.9.

We have been concentrating on what happens near convergence. To use them as preconditioner, it is just as important to study the recurrences when the solutions is far from convergence. From table 3.1 and 3.2 we see that at the beginning of the Rayleigh quotient iteration, the condition number of J_R is relatively small. Though J_A is well behaved near convergence, it can have very large condition far from convergence, see table 3.4 and 3.7. The matrix J_I is better than J_A in this respect, because far from convergence, the condition number of J_I is fairly small compared to that of J_A. In fact, table 3.8 and 3.9 show that the condition number of J_I is very close to that of J_R, see table 3.1 and 3.2.

3.2.4 Constrained Newton method

Defining $r \equiv (A - x^T Ax I)x$, i.e., replacing λ with $x^T Ax$, the eigenvalue problem can be expressed as an optimization problem with an equality constraint,

$$\begin{cases} Ax - xx^T Ax = 0, \\ \|x\| = 1. \end{cases} \tag{3.20}$$

With the eigenvalue problem stated in this form, Tapia's algorithm for constrained optimization can directly apply [142].

The algorithm given in [142] prescribes a formula for constructing a Newton recursion to solve equation (3.20). To apply to the eigenvalue problem, we need to find the Jacobian matrix, J_C. Specifically, the recursion can be written as

$$x_{i+1} = \frac{x_i - J_C^{-1} r_i}{\|x_i - J_C^{-1} r_i\|}, \tag{3.21}$$

$$J_C \equiv \frac{\mathrm{d}r}{\mathrm{d}x} = A - \lambda_i I - x_i x_i^T (A + A^T). \tag{3.22}$$

When A is symmetric, the Jacobian matrix can be simplified to

$$J_C = A - \lambda_i I - 2x_i (Ax_i)^T.$$

This scheme for computing eigenpairs will be referred to as the constrained Newton recurrence later.

In order for equation (3.21) to be a valid Newton method, we need to be able to invert the Jacobian matrix. The following lemma states when J_C is nonsingular.

Lemma 3.5 *If (λ^*, x^*) is a simple eigenpair and λ is not zero, then J_C is nonsingular.*

Proof. To prove this lemma, we only need to argue that there is no nontrivial vector which satisfies the following equation

$$J_C y = 0.$$

Substitute the definition of J_C, equation (3.22), into the above equation and note that (λ^*, x^*) is an exact eigenpair. The above equation becomes

$$(A - \lambda I)y = 2x^* {x^*}^T (A + A^T)y.$$

If y is not zero, there are two possibilities. If ${x^*}^T(A + A^T)y$ is zero, then y is an eigenvector corresponding to λ^*. Since λ^* is a simple eigenvalue, $y = \alpha x^*$ where α is a nonzero scalar constant. Because ${x^*}^T(A + A^T)x^* = 2\lambda^*$ is not zero, ${x^*}^T(A + A^T)y$ can not be zero. The second possibility is that y is a principal vector of grade 2 when ${x^*}^T(A + A^T)y$ is not zero. This is not possible because λ is a simple eigenvalue. Thus, there is no nontrivial y that can satisfy $J_C y = 0$. □

The residual $r = Ax - x^T Axx$ is a polynomial of x. A consequence of this lemma is that there exists a region around x^* where for any vector x the Jacobian matrix J_C is nonsingular. In order for the recurrence not to break down, x_{i+1} can not be zero. If x_{i+1} is zero, the following is true,

$$J_C x_i = r_i.$$

The left-hand side of the above equation is $J_C x_i = r_i - 2\lambda_i x_i$. It is clear that if λ_i is zero, then x_{i+1} is zero. When λ_i is close to an nonzero eigenvalue, we can expect λ_i to be nonzero. In which case, the above constrained Newton recurrence will not break down. If the matrix A is definite, the method will not break down either. In general, when the recurrence is far from converging, the probability of encounter a zero Rayleigh quotient is small because the x values that can produce zero Rayleigh quotient is a very small portion of all possible x values.

Near a simple nonzero eigenvalue, the recurrence described by equation (3.21) satisfies the conditions of Theorem 3.3 of [142]. Therefore given a good initial guess, it converges quadratically. The following lemma shows that it will convergence cubically for symmetric eigenvalue problems.

Lemma 3.6 *When λ_i is not zero and not an exact eigenvalue, if x_i is the same for both the constrained Newton scheme and the Rayleigh quotient iteration, then, x_{i+1} is also the same.*

When λ_i is an exact eigenvalue, the Rayleigh quotient iteration breaks down. The constrained Newton method breaks down when λ_i is zero or when λ_i is an exact eigenvalue with multiplicity greater than 1. The premise of the lemma ensures that both recurrences will not break down.

Proof. If x_i is the same for both methods, in order to show x_{i+1} is the same too, we will first show that x_C and x_R are parallel to each other, where

$$x_C = x - J_C^{-1} r, \quad x_R = J_R^{-1} x.$$

Since there is no other subscript involved in the rest of this proof, we drop the subscript i from the variables. Substituting the value of J_C from equation (3.22), the relation about x_C can be rewritten as

$$(A - \lambda I - xx^T A - xx^T A^T)(x_C - x) = -r.$$

After some straightforward rearrangement of the terms, and noting that $r = (A - \lambda I)x$, it can be written as

$$(A - \lambda I)x_C = xx^T(A + A^T)(x_C - x). \tag{3.23}$$

Let $\alpha = x^T(A + A^T)(x_C - x)$, equation (3.23) becomes

$$(A - \lambda I)x_C = \alpha x.$$

If both RQI and the constrained Newton method do not break down, then $(A - \lambda I)$ is nonsingular and x_C is nonzero. Thus the left-hand side of equation (3.23) is not zero. This shows that α must be nonzero.

In conclusion, x_C and x_R differ by a nonzero scalar constant α. Since the two methods scale x_C and x_R to generate the eigenvector approximations, they generate the same vector x_{i+1} for the next iteration. □

i	λ_i	$\|r_i\|$	κ_i
1	0.01361736175	0.003830979028	523.5
2	0.01427847652	0.000357647692	1109
3	0.01430356647	2.468630821e-06	3.029e+04
4	0.01430356864	2.006917728e-12	3.503e+08
5	0.01430356864	3.078958720e-17	5.735e+16

Table 3.10: The intermediate results of using the constrained Newton recurrence on PLAT362.

i	λ_i	$\|r_i\|$	κ_i
1	-90685.22644	911704.3057	759.3
2	2500.224244	34828.83139	2.550e+04
3	0.211851563	672.661174	3.237e+05
4	0.2109448741	0.0128510811	1.549e+12
5	0.210650847	0.001874827738	2.271e+12
6	0.2097380775	0.001081505247	5.322e+11
7	0.2094248973	0.000100056626	1.739e+12
8	0.2094224436	6.730182732e-08	7.831e+12
9	0.2094224436	2.973326427e-09	8.052e+12

Table 3.11: The results of applying the constrained Newton recurrence on EX2.

To have an idea of how large the constant $\alpha = x_C/x_R$ is, we replace x_C with αx_R in the definition of α,

$$\alpha = x^T(A + A^T)(\alpha x_R - x).$$

This leads to the following equation for α,

$$\alpha = \frac{x^T(A + A^T)x}{x^T(A + A^T)x_R - 1}.$$

If x is close to the exact eigenvector x^* and A is symmetric, we can approximate α as follows,

$$\alpha \approx \frac{2\lambda}{\frac{2\lambda}{\lambda^* - \lambda} - 1} \approx \lambda^* - \lambda.$$

The assumption of the above approximation is that x is close enough to the exact solution such that $x_R \approx x/(\lambda^* - \lambda)$. Using this approximate α, we know that near convergence,

$$x_C \approx x. \tag{3.24}$$

From this equation, we know that the correction to x_i produced by the constrained Newton recurrence is almost perpendicular to x_i as in the Newton-Olsen recurrence, see equation (3.17). Similar to the Newton-Olsen recurrence, we can expect this recurrence to be numerically more stable than the Rayleigh quotient iteration. In addition, since the Jacobian matrix of this recurrence is nonsingular near convergence, we expect no obstacle to realize this theoretical benefit.

Tables 3.10 and 3.11 show the eigenvalue and residual norm at each step of the constrained Newton recurrence. Note that the eigenvalue approximations and the residual norms are nearly identical with that of RQI. Similar to J_A, the condition number of J_C also grows quite rapidly on the PLAT362 test problem because the eigenvalues has multiplicity of two, see table 3.3 and 3.10.

3.2.5 Jacobi-Davidson recurrence

The preconditioning step of the Jacobi-Davidson method solves the following equation approximately [14, 38, 41, 124, 125, 126],

$$(I - xx^T)(A - \lambda I)(I - xx^T)z = r.$$

The matrix $J_J \equiv (I - xx^T)(A - \lambda I)(I - xx^T)$ is singular, so it is not possible to solve the above equation in the usual sense. One of the objectives of Jacobi-Davidson preconditioner is to make z orthogonal to x. However, the above equation does not enforce this condition. On the contrary, since x is in the null space of J_J, the solution to this equation can have an arbitrarily large component in x direction. Following the example of the correction equation, see equation (3.8), we regard the Jacobi-Davidson preconditioning scheme as solving the following equation,

$$z = \left((I - xx^T)(A - \lambda I)(I - xx^T)\right)^+ r. \tag{3.25}$$

Because the pseudoinverse is always defined for a matrix, the result from this equation is unique and z is orthogonal to x. If λ is the Rayleigh quotient of x and r is the corresponding residual vector, the original Jacobi-Davidson preconditioning equation, equation (3.6), is consistent [10, 7, 51]. In this case, the result from equation(3.25) satisfies the original equation and it is the minimum norm solution. Based on equation (3.25), we define the following recurrence formula for finding an eigenvector starting from a unit vector x_0,

$$x_{i+1} = \frac{x_i - J_J^+ r_i}{\|x_i - J_J^+ r_i\|}. \tag{3.26}$$

This recurrence looks similar to the constrained Newton recurrence. The Jacobi-Davidson method has been argued to be a Newton method [38, 126] based on equation (3.6). It is reasonable to regard equation (3.26) as a form of Newton's method for the eigenvalue problem. Because J_J^+ is always defined, $J_J^+ r$ is computable. Since x_i is perpendicular to $J_J^+ r_i$, $x_i - J_J^+ r_i$ will never be zero. Thus the above recurrence is well defined. Because of the singular Jacobian matrix, we may not expect it to have super-linear convergence as a Newton method with nonsingular Jacobian. However, the following lemma shows that it actually converges cubically because of the connection to the Rayleigh quotient iteration.

Lemma 3.7 *For Hermitian matrices, the Jacobi-Davidson recurrence described by equation (3.26) is equivalent to the Rayleigh quotient iteration as long as the Rayleigh quotient iteration can be carried out and $x_i^T (A - \lambda_i I)^{-1} x_i$ is never zero.*

Proof. Starting with a unit vector x, $\lambda = x^T A x$, let

$$x_R = J_R^{-1} x, \quad x_J = x - J_J^+ r,$$

where $J_R = A - \lambda I$, $J_J = (I - xx^T)(A - \lambda I)(I - xx^T)$, x_R and x_J are the unscaled solutions of RQI and the Jacobi-Davidson recurrence. Define $z \equiv x - x_J$, because of the consistency of equation (3.6), the following must be true,

$$(I - xx^T)(A - \lambda I)(I - xx^T)z = r, \qquad x^T z = 0.$$

From this equation, we know that

$$(A - \lambda I)z = r + \alpha x,$$

where α is an undetermined scalar constant. This equation leads to the following solution for z when λ is not an eigenvalue,

$$z = x + \alpha x_R.$$

Now we can use the orthogonality condition of z to determine α as follows,

$$\alpha = \frac{-1}{x^T x_R} = \frac{-1}{x^T (A - \lambda I)^T x}.$$

Thus the solution of x_J is

$$x_J = \frac{x_R}{x^T x_R}.$$

If $x_i^T (A - \lambda_i I)^{-1} x_i$ is never zero, α is a nonzero finite scalar number. In this case, x_J is parallel to x_R. After they are scaled to unit norm, they should be identical to each other. □

i	λ_i	$\|r_i\|$
1	0.01361736175	0.003830979028
2	0.01427847652	0.000357647692
3	0.01430356647	2.468630821e-06
4	0.01430356864	2.007419541e-12
5	0.01430356864	2.521861421e-17

Table 3.12: Solving PLAT362 with the Jacobi-Davidson recurrence.

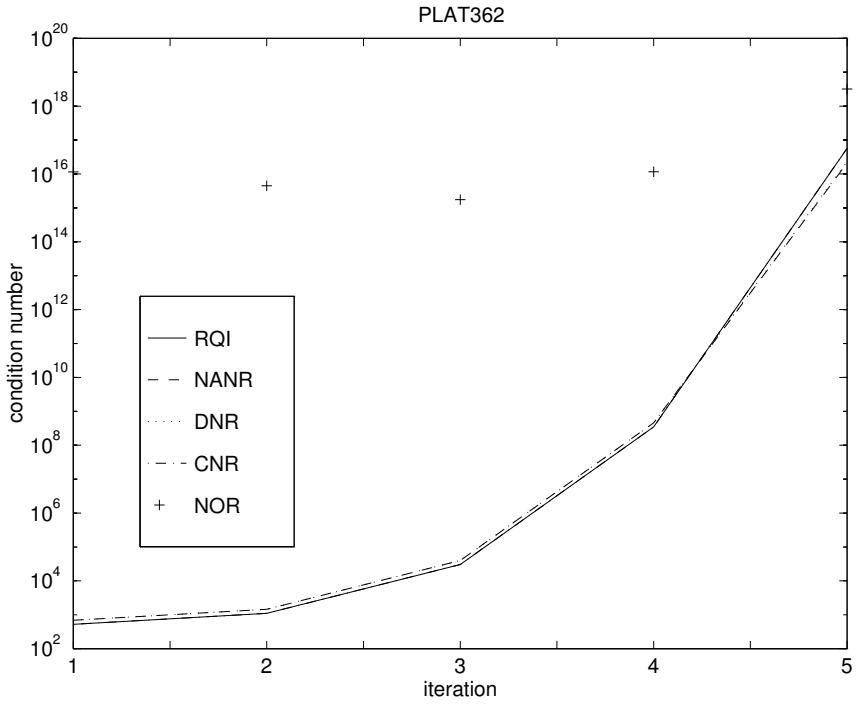

Figure 3.2: Condition numbers of the Jacobian matrix when solving PLAT362.

One observation from this proof is that the solution from the Jacobi-Davidson recurrence is exactly the same as that of the Newton-Olsen recurrence. They are the same even before scaling. This confirms again that the Jacobi-Davidson preconditioning is equivalent to the Olsen preconditioning scheme.

Table 3.12 shows the approximate solutions from applying the Jacobi-Davidson recurrence on PLAT362. We did not show the condition number because J_J is known to be singular. As the lemma predicted, this method produced exactly the same results as that of RQI.

Figures 3.2 and 3.3 depicts the condition number of several iteration matrices versus the iteration number. The names of the recurrences are abbreviated to contain only their initials. From figure 3.2 we can see that the conditioner number of J_O, from the Newton-Olsen recurrence (NOR), is consistent with the fact that it is singular. The four other methods shown in figure 3.2 fall into two groups, the Rayleigh quotient iteration (RQI) and the constrained Newton recurrence (CNR) are one, the normalized augmented Newton recurrence (NANR) and the inflated Newton recurrence (DNR) are in the other. In this case, the condition numbers of J_R (RQI) and J_C (CNR) are nearly identical at every step, the condition numbers of J_A (NANR) and J_I (DNR) are very close to each other. Since the eigenvalue computed is not simple, as the iterations approach convergence, the condition number of all iteration matrices increase to 10^{16} which is infinity for 64-bit IEEE arithmetic operations. In the EX2 case, figure 3.3, the condition number of J_A (NANR) is very large initially which should be considered a disadvantage of the normalized augmented Newton method. Near convergence,

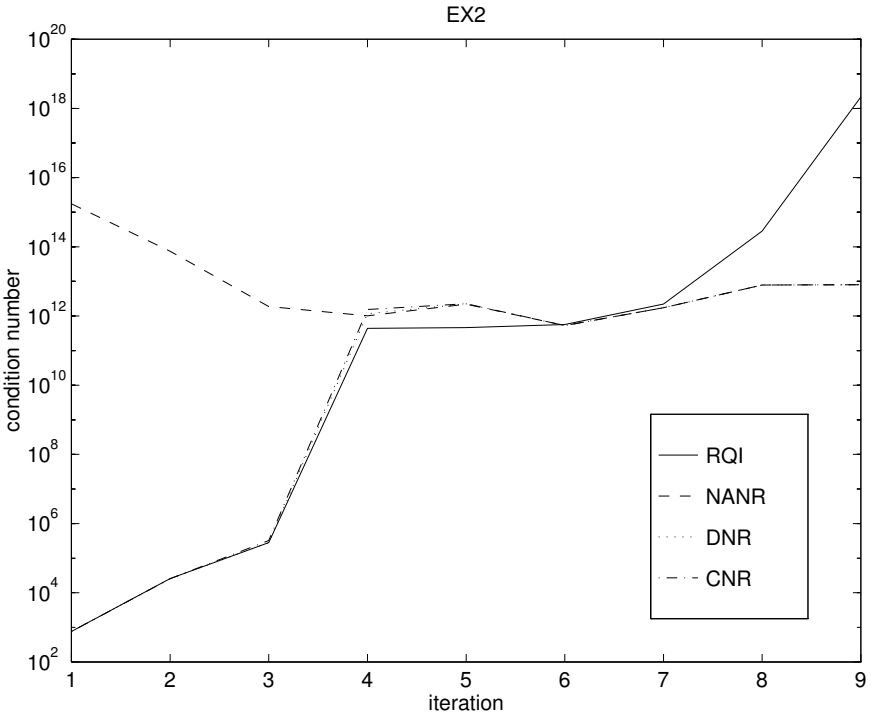

Figure 3.3: Condition numbers of the Jacobian matrix when solving EX2.

only the condition number of J_R did not stop increasing. As we have shown, near convergence, J_A, J_C, J_I are guaranteed to be nonsingular. Figure 3.3 clearly illustrates this fact.

3.2.6 Effects of biased shift

In previous subsections, we have discussed our approaches to avoid the singular iteration matrix problem. Here we revisit a technique used in [133] to address the mis-convergence problem and the indefinite iteration matrix problem.

Table 3.1, 3.3, 3.5, 3.6, 3.8, 3.10, and 3.12 show the progress of seven eigen-system solvers. We wanted the smallest eigenvalue and the corresponding eigenvector. However, none of them converges to the desired eigenvalue. This is a good example of the mis-convergence problem mentioned on page 30. The Newton method converges quickly to an eigenpair close to the initial guess. Obviously, RQI has a different measure of distance from the augmented Newton method, see table 3.2 and 3.4. Nevertheless, they do not converge to the smallest eigenvalue. Additional modification is needed to reach the desired solution.

To make the eigen-system solvers converge to the smallest eigenvalue, we borrow a shifting strategy from preconditioning [133], see equation (3.5). Since we only compute one eigenvalue, no gap information is available, we will always use the residual norm as the error estimate ϵ instead of equation (3.4). The new iteration matrices are

$$\begin{aligned}
J_R &= A - (\lambda - \|r\|)I, \\
J_A &= \begin{pmatrix} A - (\lambda - \|r\|)I & -x \\ -x^T & 0 \end{pmatrix}, \\
J_O^{-1} &= (A - (\lambda - \|r\|)I)^{-1} \left(I - \frac{xx^T (A - (\lambda - \|r\|)I)^{-1}}{x^T (A - (\lambda - \|r\|)I)\, x} \right)^{-1}, \\
J_I &= (A - (\lambda - \|r\|)I + xx^T),
\end{aligned}$$

i	λ_i	$\|r_i\|$	κ_i
1	0.003990869031	0.007600999703	47.02
2	0.001231682013	0.001866070494	215.5
3	0.0004364948496	0.0004969443916	1222
4	2.040621091e-05	8.011132754e-05	1.281e+04
5	5.379204885e-06	1.386776412e-05	1.297e+04
6	1.836408562e-06	2.954308290e-06	9.122e+04
7	4.082588328e-07	7.881009546e-07	6.926e+05
8	1.172643067e-07	1.864027929e-07	2.038e+06
9	3.123300507e-08	4.955108890e-08	1.12e+07
10	8.241978672e-09	1.289119192e-08	4.226e+07
11	1.760465318e-09	3.300845435e-09	1.664e+08
12	4.286225365e-10	8.463469134e-10	5.015e+08
13	4.420207032e-11	2.043842802e-10	1.838e+09
14	4.630879284e-12	2.980025841e-11	4.729e+09
15	3.556038027e-12	1.030103458e-12	2.696e+10
16	3.554631314e-12	1.424944003e-15	7.527e+11

Table 3.13: The intermediate solutions of RQI with the biased shift on PLAT362.

i	λ_i	$\|r_i\|$	κ_i
1	0.003990869031	0.007600999703	3.549e+16
2	0.001231682013	0.001866070494	1.304e+15
3	0.0004364948496	0.0004969443916	1.944e+16
4	2.040621091e-05	8.011132754e-05	9.472e+16
5	5.379204885e-06	1.386776412e-05	2.041e+15
6	1.836408562e-06	2.954308290e-06	1.347e+15
7	4.082588328e-07	7.881009546e-07	1.913e+15
8	1.172643067e-07	1.864027929e-07	1.91e+15
9	3.123300508e-08	4.955108890e-08	3.746e+15
10	8.241978666e-09	1.289119197e-08	2.516e+15
11	1.760465327e-09	3.300847839e-09	4.574e+15
12	4.286230045e-10	8.463534536e-10	2.191e+15
13	4.420288725e-11	2.044201828e-10	9.53e+15
14	4.631278766e-12	2.982954965e-11	8.267e+15
15	3.556041538e-12	1.101367765e-12	6.055e+15
16	3.554626657e-12	1.828728350e-14	2.187e+15

Table 3.14: The intermediate solutions of the Newton-Olsen recurrence with biased shift on PLAT362.

$$\begin{aligned} J_C &= A - (\lambda - \|r\|)I - 2x(Ax)^T, \\ J_J &= (I - xx^T)(A - (\lambda - \|r\|)I)(I - xx^T). \end{aligned}$$

To simplify the equations, we use δ to denote the biased estimate of eigenvalues in later discussions.

Tables 3.1, 3.5, 3.6, 3.8, 3.10, and 3.12 show the results of find an eigenpair of PLAT362 without shifting, and tables 3.13, 3.14, 3.15, 3.16, 3.17, and 3.18 show the results with the biased shift discussed before. We did not show the results from the augmented Newton method because it did not converge with the biased shift. The shift has changed the algorithm enough to break it. This may make it less favorable compared to others. The other six methods converge to the smallest eigenvalue. Again we notice the difference in the magnitude of the condition numbers, see table 3.13–3.17. Another point to note here is that even with biased shift, the six methods also produces identical intermediate solutions.

i	λ_i	$\|r_i\|$	κ_i
1	0.003990869031	0.007600999703	60.45
2	0.001231682013	0.001866070494	278.1
3	0.0004364948496	0.0004969443916	1578
4	2.040621091e-05	8.011132754e-05	1.655e+04
5	5.379204885e-06	1.386776412e-05	1.675e+04
6	1.836408562e-06	2.954308290e-06	1.178e+05
7	4.082588328e-07	7.881009546e-07	8.945e+05
8	1.172643067e-07	1.864027929e-07	2.633e+06
9	3.123300508e-08	4.955108890e-08	1.446e+07
10	8.241978671e-09	1.289119191e-08	5.458e+07
11	1.760465313e-09	3.300845428e-09	2.149e+08
12	4.286225377e-10	8.463469076e-10	6.477e+08
13	4.420207444e-11	2.043842793e-10	2.374e+09
14	4.630879693e-12	2.980025340e-11	6.107e+09
15	3.55604338e-12	1.030113584e-12	3.481e+10
16	3.554634202e-12	1.437247262e-15	9.721e+11

Table 3.15: The intermediate solutions of the normalized augmented Newton recurrence with biased shift on PLAT362.

i	λ_i	$\|r_i\|$	κ_i
1	0.003990869031	0.007600999703	61.45
2	0.001231682013	0.001866070494	279.1
3	0.0004364948496	0.0004969443916	1579
4	2.040621091e-05	8.011132754e-05	1.655e+04
5	5.379204885e-06	1.386776412e-05	1.675e+04
6	1.836408562e-06	2.954308290e-06	1.178e+05
7	4.082588328e-07	7.881009546e-07	8.945e+05
8	1.172643067e-07	1.864027929e-07	2.633e+06
9	3.123300508e-08	4.955108891e-08	1.446e+07
10	8.241978667e-09	1.289119191e-08	5.458e+07
11	1.760465324e-09	3.300845435e-09	2.149e+08
12	4.286225373e-10	8.463469144e-10	6.477e+08
13	4.420206633e-11	2.043842666e-10	2.374e+09
14	4.630881439e-12	2.980025683e-11	6.107e+09
15	3.556041954e-12	1.030106692e-12	3.481e+10
16	3.554625284e-12	1.421387398e-15	9.721e+11

Table 3.16: The intermediate solutions of the inflated Newton iteration with biased shift on PLAT362.

Even with the biased shift, there is no guarantee that given an arbitrary initial guess, any of the method will converge to the smallest eigenvalue. For example, if the initial guess is an eigenvector not corresponding to the smallest eigenvalue, all Newton-type methods will find the initial residual is zero and stop. Table 3.19 is a more ordinary example. The Rayleigh quotient iteration with biased shift is applied to the EX2 with the initial guess $[1, 1, \ldots, 1]^T$, it converges to an eigenvalue much smaller than without the biased shift, see table 3.2. However the eigenvalue it converges to is still far from the smallest one.

For Hermitian eigenvalue problem the biased shift also cures the indefinite iteration matrix problem of regular Davidson preconditioning, see page 30. Because the biased estimated of the smallest eigenvalue is usually less than the actual one, $J_R(= A - (\lambda - \|r\|)I)$ is positive definite. By construction J_A is indefinite. If J_R is positive definite, J_J is positive semi-definite, and J_I is also positive definite. The Jacobian matrices for the Newton-Olsen recurrence and the constrained Newton recurrence still may be indefinite.

i	λ_i	$\|r_i\|$	κ_i
1	0.003990869031	0.007600999703	87.34
2	0.001231682013	0.001866070494	528.8
3	0.0004364948496	0.0004969443916	2.89e+04
4	2.040621091e-05	8.011132754e-05	4.085e+04
5	5.379204885e-06	1.386776412e-05	1.559e+04
6	1.836408562e-06	2.954308290e-06	1.42e+05
7	4.082588328e-07	7.881009546e-07	3.186e+06
8	1.172643067e-07	1.864027928e-07	4.497e+06
9	3.123300508e-08	4.955108888e-08	7.077e+07
10	8.241978654e-09	1.289119190e-08	2.471e+08
11	1.760465324e-09	3.300845447e-09	8.113e+08
12	4.286225581e-10	8.463469248e-10	1.195e+09
13	4.420204006e-11	2.043841908e-10	3.181e+09
14	4.630877630e-12	2.980023791e-11	5.303e+09
15	3.556036657e-12	1.030109282e-12	3.593e+10
16	3.554631124e-12	1.419808576e-15	7.527e+11

Table 3.17: The intermediate solutions of the constrained Newton recurrence with the biased shift on PLAT362.

i	λ_i	$\|r_i\|$
1	0.003990869031	0.007600999703
2	0.001231682013	0.001866070494
3	0.0004364948496	0.0004969443916
4	2.040621091e-05	8.011132754e-05
5	5.379204885e-06	1.386776412e-05
6	1.836408562e-06	2.954308290e-06
7	4.082588328e-07	7.881009546e-07
8	1.172643067e-07	1.864027929e-07
9	3.123300506e-08	4.955108889e-08
10	8.241978667e-09	1.289119191e-08
11	1.760465327e-09	3.300845443e-09
12	4.286225457e-10	8.463469260e-10
13	4.420207340e-11	2.043842919e-10
14	4.630881801e-12	2.980026807e-11
15	3.556038783e-12	1.030102692e-12
16	3.554632854e-12	1.419106276e-15

Table 3.18: The intermediate solutions of the Jacobi-Davidson recurrence with biased shift on PLAT362.

Figure 3.4 plots the condition numbers shown in tables 3.13–3.17. Compared with figure 3.2, we see that the conditioner numbers are smaller in figure 3.4. The fact that the conditioner number of $A-(\lambda-\|r\|)I$ is less than that of $A-\lambda I$ can be explained by the well known fact that λ converges faster than the residual norm. Since $|\lambda^*-\lambda|$ is less than $\|r\|$, $A-\lambda I$ is closer to singularity than $A-(\lambda-\|r\|)I$. Because $A-(\lambda-\|r\|)I$ is the dominant part of J_A, J_C and J_I, reducing the conditioner number J_R may lead to a reduction of their condition number as well.

3.2.7 Characteristics of the Newton methods

Because of the equivalence properties, the Newton methods described actually belongs to two categories, the Rayleigh quotient iteration and the augmented Newton method. When used as preconditioner, usually we

i	λ_i	$\|r_i\|$	κ_i
1	-6144041.085	10062581.84	577.8
2	-17016751.02	6701780.037	213.2
3	-21415222.33	2193673.55	1151
4	-23089128.21	402183.1026	1415
5	-23130707.73	37601.76191	1878
6	-23131028.49	403.1330178	1.815e+04
7	-23131028.52	0.0495892453	1.679e+06
8	-23131028.52	1.648623805e-08	1.365e+10
9	-23131028.52	1.228988946e-08	5.924e+16

Table 3.19: The intermediate solutions of RQI with biased shift on EX2.

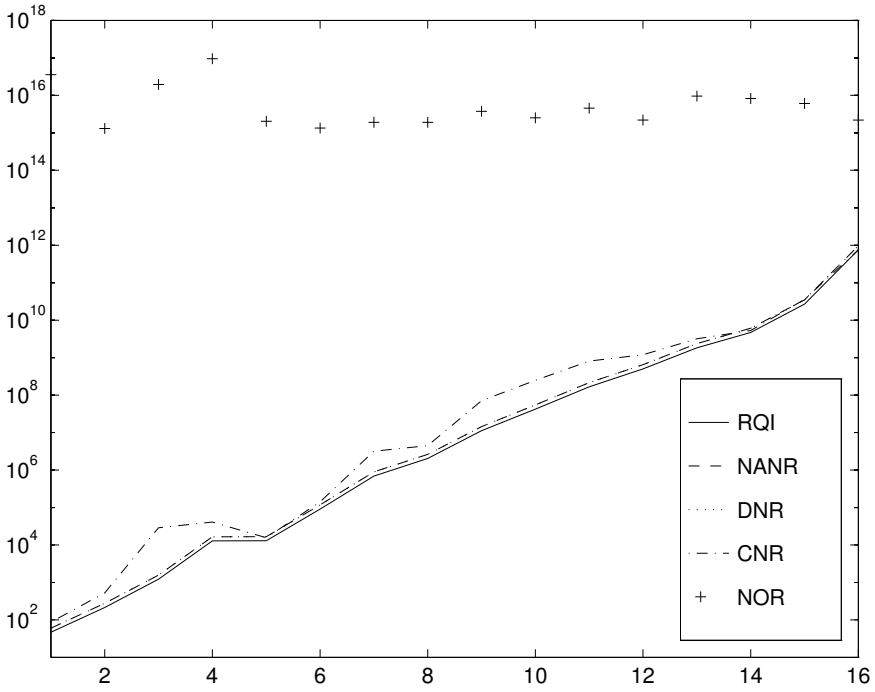

Figure 3.4: Condition numbers of the Jacobian matrix with biased shift on PLAT362.

only take one step of the Newton iteration, in which case, the augmented Newton method is equivalent to the normalized augmented Newton method if the approximate eigenvectors are unitary. Thus all eigen-system solvers of interests are variations of the Rayleigh quotient iteration. We have found six different way of implementing the Rayleigh quotient iteration which operate the following iteration matrices, J_R, J_A, J_O, J_I, J_C, and J_J, see page 47. Out of these matrices, we have shown that J_I, J_C and J_A are nonsingular near convergence. The next lemma shows how the spectra of some of them are related to each other.

Lemma 3.8 *For any vector y perpendicular to x, $y^T J_R y = y^T J_I y = y^T J_C y = y^T J_J y$.*

Proof. If y is perpendicular to x, $x^T y = y^T x = 0$.

$$y^T J_I y = y^T (A - \lambda I + x x^T) y = y^T (A - \lambda I) y = y^T J_R y.$$

$$y^T J_C y = y^T (A - \lambda I) y - y^T x x^T (A + A^T) y = y^T (A - \lambda I) y.$$

$$y^T J_J y = y^T (1 - xx^T)(A - \lambda I)(I - xx^T)y = y^T (A - \lambda I)y.$$

□

In the limit where x is the exact eigenvector, this lemma shows that the spectrum of J_R, J_I, J_C and J_J are only a simple sift of the spectrum of A except one eigenvalue corresponding to λ^*. In the case of J_R and J_J, λ^* is translated to zero, i.e., they are singular. J_I translates λ^* to 1. J_C translates it to $-2\lambda^*$ if the matrix A is Hermitian. Since J_A is not of the same size as A, it is not easy to characterize how its spectrum relates to that of A. We could image its effects is represented by J_I because J_I can be viewed as an approximate Schur form of J_A. In short, the new recurrences described here avoid singularity in their Jacobian matrices by transforming the zero eigenvalue into a different location.

We have addressed the problem of singular iteration matrix. Another problem is also addressed at the same time is the Linear Dependent Basis problem, see page 30. By construction, the Newton-Olsen recurrence and the Jacobi-Davidson recurrence produce corrections that are orthogonal to current approximate eigenvector x. Because of the connection to the Newton-Olsen scheme, the normalized augmented Newton recurrence should also produce orthogonal corrections as well. The constrained Newton recurrence and the inflated Newton recurrence do not have this orthogonality. However, the problem should be less severe than the original Davidson preconditioning scheme.

The properties of the iteration matrices near convergence are important. For practical use, the properties of the Jacobian matrices far from convergence are as important. It is clear that the condition number of $A - \lambda I$ is small when λ is far from any eigenvalue. Therefore, J_R is well-behaved far from convergence. This fact can partly explain the success of the Davidson preconditioning scheme. Since a small modification can significantly change the spectrum of a matrix, it is hard to quantify the behavior of other Jacobian matrices analytically. We have seen one example where J_A has large condition number far from convergence, see figure 3.3 or table 3.7. This indicates that the normalized augmented Newton method is more likely to encounter difficulties than others. From figures 3.2, 3.3, and 3.4, we see that the condition number of the Jacobian matrices from RQI, the inflated Newton recurrence and the constrained Newton recurrence are close to each other far from convergence. The conditioner number of PLAT362 is about 10^{11}. The condition number of EX2 is about 10^{10}. Compared with this numbers, the condition numbers of J_C, J_I, and J_R are remarkably small at the beginning of the Newton iterations, see figures 3.2, 3.3 and 3.4.

3.3 Newton-type preconditioners

In the framework of algorithm 2.3, the ultimate purpose of preconditioning is to generate a good basis V_m. The Davidson preconditioning scheme mentioned above basically approximates the correction equation (3.2). In this section we search for new preconditioning scheme by discussing different ways of approximate the the Newton recurrences described in the previous section.

The Rayleigh quotient iteration method is an effective method for find one eigenvalue and its eigenvector [51, 93]. In order to build an eigen-system solver conform to the structure of algorithm 2.3, we can simply define V_m to be a basis spanning $\{x_0, x_1, \ldots, x_{m-1}\}$. For convenience of reference, we will call this basis the Rayleigh quotient basis. Because the optimality of the Rayleigh-Ritz projection, the residual norm of the solution found by the Rayleigh-Ritz projection on this Rayleigh quotient basis should be no greater than the residual normal of the solution (λ_{m-1}, x_{m-1}) from the Rayleigh quotient iteration. This simple observation indicates that if we approximate the Rayleigh quotient iteration as preconditioner for the Davidson method, it is possible to achieve cubic convergence rate for symmetric eigenvalue problems. The Davidson preconditioner computes $z \approx (A-\lambda I)^{-1} r$ to augment the current basis. The Rayleigh quotient iteration based preconditioner gives $z \approx (A - \lambda I)^{-1} x$ to augment the current basis. One might expect this second form of preconditioner to perform better than the Davidson preconditioning because the observed difficulties related to the Davidson preconditioning, see page 30. A small experiment indicates that the contrary is true, see table 3.20.

Table 3.20 contains the number of matrix-vector multiplications used to compute 5 smallest eigenvalues and their corresponding eigenvectors of three diagonal matrices. The test matrices are from Harwell-Boeing collection [37]. The three matrices used in this test are the three largest diagonal matrices in the collection, see table 2.12. Diagonal matrices are used here because it is easy to invert the preconditioners. This small

preconditioner	$(A-\lambda I)^{-1}r$	$(A-\lambda I)^{-1}x$
BCSSTM21	12	25
BCSSTM24	28	78
BCSSTM25	31	107

Table 3.20: Number of MATVEC required to compute 5 smallest eigenpairs.

test indicates that the Davidson preconditioner is more effective. This is because $(A-\lambda I)^{-1}x$ is badly scaled, and almost parallel to x.

Note that the Rayleigh quotient iteration has most of the problems we have observed in the Davidson preconditioning, see page 30. This test shows that a Newton type of preconditioner is better suited to be a preconditioner for the Davidson method. In the remaining of this section we discuss how to turn the other Newton methods for eigenvalue problems into preconditioners for the Davidson method.

For all the Newton recurrence, it is very simple to modify the preconditioning step of the Davidson method to use the New preconditioning scheme. In step 2.d.iii of algorithm 4.1, instead of solving $z = (\mathrm{diag}(A) - \lambda I)^{-1}r$, we simply compute

$$z = J^{-1}r,$$

where J is one of the Jacobian matrices described in the previous section. Because of the equivalence properties, if the above preconditioning step is solved exactly, we should expect cubic convergence with these new preconditioning schemes.

Augmented Newton preconditioner. The nonzero structure of J_A is a straightforward modification to the nonzero pattern of A. It should be easy to modify most of the incomplete factorization schemes to factorized J_A. Because we always have unitary approximate eigenvectors, the preconditioning step with the augmented Newton preconditioner approximately computes $J_A^{-1}\binom{r}{0}$ and discards the last element of the solution. The exact form of this preconditioner is the normalized augmented Newton recurrence.

Olsen preconditioner. The simplest way of realizing the Olsen preconditioning scheme is to implemented it as a driver over existing Davidson preconditioners. Denote the original preconditioner as $(M-\delta I)$, and let $z_r = (M-\delta I)^{-1}r$, $z_x = (M-\delta I)^{-1}x$. The Olsen preconditioning scheme computes

$$z = z_r - \frac{x^T z_r}{x^T z_x} z_x. \tag{3.27}$$

In exact form, the above formula can be simplified slightly as follows

$$z = (A-\lambda I)^{-1}r - \frac{(A-\lambda I)^{-1}x}{x^T(A-\lambda I)^{-1}x}.$$

This leads to a modified form of the Olsen preconditioning scheme,

$$z = z_r - \frac{z_x}{x^T z_x}. \tag{3.28}$$

This new form is slightly simpler than equation (3.27), but its output is no longer orthogonal to x if $M \neq A$. We know the the jacobian matrix of the Newton-Olsen scheme is singular. However, implementing this scheme either use equation (3.27) or (3.28), the matrix J_O is not directly used.

Inflated Newton preconditioner. The Jacobian matrix for the inflated Newton recurrence is $J_I = A - \lambda I + xx^T$. This matrix could be inverted through the Sherman-Morrison formula in which case it will lead back to an approximate form of the Newton-Olsen recurrence. Since there is already two variations of the Olsen preconditioning scheme, we will find a different way of using J_I. A Matrix-vector multiplication with J_I is not much more complex than a matrix-vector multiplication with A. This suggests that a matrix-vector multiplication based preconditioner. Common preconditioners based on matrix-vector multiplications

include polynomial preconditioners [6, 39, 149, 62, 113], Krylov based iterative solvers [10, 119], and approximate inverse built by Krylov iterative solvers [24, 66]. Because polynomial preconditioners usually require information about the spectrum of J_I which is unlikely to be available, they are not as easy to use as the other choices. Approximate inverse preconditioners are attractive because they can be efficient in parallel environment. However the algorithms for computing approximate inverses are not as widely available as Krylov subspace methods. For simplicity, we will only use the Krylov iterative solvers, e.g., CG, GMRES, to test this preconditioning scheme.

Constrained Newton preconditioner. Similar to the Inflated Newton preconditioner case, the iteration matrix J_C of the constrained Newton recurrence is again a rank one modification of J_R. For a Hermitian eigenvalue problem, multiplying the Jacobian matrix $J_C = A - \lambda I - 2x(Ax)^T$ by a vector can be easily performed, because x and Ax are computed by the Davidson method in the process of computing the residual vector. It is possible to design a special incomplete factorization method that factories J_C without explicitly computing the whole matrix. However the simplest option to approximate a constrained Newton step is to use a Krylov subspace method as we have indicated in the inflated Newton preconditioner case.

Jacobi-Davidson preconditioner. The iteration matrix in this case is $J_J = (I - xx^T)(A - \lambda I)(I - xx^T)$. This matrix is more complex than J_C and J_I, the only practical option of using this matrix is to use it in a matrix-vector multiplication. As before we have three choices, polynomial preconditioner, Krylov iterative solver, and approximate inverse. Since the matrix J_J is singular, the approximate inverse algorithm need to compute an approximation of the pseudoinverse. How effective are current approximate inverse algorithms in finding a pseudoinverse approximation is an open question. Usually the Jacobi-Davidson preconditioner is solved with Krylov iterative solvers. Because the right-hand side r is in the range of J_J, the approximate solution generated by Krylov subspace iterative solvers and polynomial methods are essentially approximations to $J_J^+ r$.

In summary, we have three different kinds of preconditioners, incomplete LU factorizations on J_A, Olsen preconditioning schemes, and iterative methods applied on linear systems with J_R, J_I, J_C or J_J.

3.4 Numerical Comparisons

This section contains a number of small examples to show how the different preconditioning schemes work. Due to the large number of possible implementations, it is not possible to cover every case. We have selected to implement the following preconditioners

- **Incomplete factorization of J_A**. We decided to test the potential benefit of augmented Newton preconditioner with two incomplete LU factorizations: ILU0, ILUTP [115]. ILU0 is an incomplete factorization where the LU factors have the same nonzero pattern as the original matrix. ILUTP modifies ILU0 in three ways, (1) elements of LU factors with small absolute values are dropped, (2) a maximum number of nonzero elements in a row of LU factors, i.e., level of fill, is controlled by the user, (3) column pivoting is performed if the diagonal element is significantly smaller than another element on the same row. The ILUTP used has the level of fill equals to half of the average number of nonzero elements per row plus one, in other word, the ILUTP factorization can be slightly larger than ILU0. On average, this ILUTP allows one more nonzero elements in both L and U than that of ILU0. Elements in the incomplete LU factor that is less than 3×10^{-5} of the row norm are discarded, and pivoting is performed if the absolute value of the diagonal element is less than 0.1 of the largest one in the same row.

- **Olsen preconditioners**. As described before we have two different implementation for the Olsen preconditioner, see equations (3.27) and (3.28). In each case, one of the following three approximate solution schemes is used to approximate $(A - \delta I)$, the diagonal scaling, ILU0 and ILUTP.

- **Iterative solvers**. We have implemented CG and GMRES to work with J_R, J_C, J_I, and J_J. Two different kinds of tolerance schemes are used. First, a fixed number of matrix-vector multiplications and a fixed residual tolerance are used, i.e., when either the residual norm of the linear system has reduced

	PLAT362		EX2	
	MATVEC	time(sec)	MATVEC	time(sec)
(NONE)	>5000	-	35	0.6
diagonal	>5000	-	88	1.7
ILU0	>5000	-	63	15.4
ILUTP	>5000	-	41	10.6

Table 3.21: Conventional preconditioners applied to PLAT362 and EX2.

	PLAT362		EX2	
	MATVEC	time(sec)	MATVEC	time(sec)
ILU0	>5000	-	525	19.8
ILUTP	>5000	-	>5000	-

Table 3.22: Augmented Newton preconditioners on PLAT362 and EX2.

below the tolerance, or the number of matrix-vector multiplications used is more than the maximum specified, the iterative solver is terminated. Second, a dynamic tolerance on both the residual and the matrix-vector multiplication are used. Each time an iterative solver is called, the relative tolerance is decreased by half until the minimum 10^{-10} is reached. The maximum number of matrix-vector multiplications increase by 10 every step in the first 20 calls, it only increase by 1 every time after the 20th call.

Our first set of tests is to apply Davidson method with above preconditioning scheme on the two test matrices used in the previous section. The basis size for the Davidson method is 20. We seek 5 smallest eigenvalues from each matrix with the same initial guess, $[1, 1, \ldots, 1]^T$, for the eigenvector. In this experiment, the tolerance on residual norm is set to 14 orders smaller than the matrix norm. The maximum number of matrix-vector multiplications allowed is 5000. Note that this is a very large number compared with the matrix sizes. The test is performed on a SPARC 10 running at 40MHz.

Table 3.21 shows the results of using some of the common preconditioners used for iterative linear system solvers. We only used the Davidson method in this test. The preconditioners used here were also used in Chapter 2, more information can be found there. It is hard to compute the smallest eigenvalues of PLAT362 because the smallest eigenvalues are clustered together. It is easy to find the smallest eigenvalues of EX2 because they are well separated. Matrix EX2 has more than 60 nonzero elements per row which makes solving with ILU preconditioners significantly more expensive than diagonal scaling. Even though the two ILU preconditioners use significantly less iterations to reach convergence than the diagonal preconditioning case, they still use more time than the diagonal scaling preconditioner. Ironically, the unpreconditioned Davidson uses the least number of matrix-vector multiplication and least amount time in this case.

Table 3.22 shows the results of using ILU factorizations on J_A as preconditioners. Both preconditioners were not able to help Davidson method to reach convergence faster. Not even one eigenvalue has reached the residual norm requirement within 5000 matrix-vector multiplications for PLAT362. The Davidson method with ILU0 on J_A as the preconditioner was able find the five smallest eigenvalues of EX2. However it took significantly more matrix-vector multiplications and time than applying ILU0 on $J_R = A - \delta I$. This is largely due to the zero diagonal elements in J_A which makes it indefinite. Part of this may be also attributed to the fact that the condition number of J_A can be large when x far from any eigenvector, see figure 3.3. As a preconditioner to the Davidson method, the ILU0 on J_A works better than ILUTP on J_A on the EX2 matrix.

Tables 3.23 and 3.24 show the results of using the two Olsen preconditioning schemes with three incomplete factorizations of $(A - \delta I)$. The Davidson method failed to find any eigenpair of PLAT362. Thus we are only left to compare the result of EX2 with table 3.21. In this case, it is clear that the original Olsen scheme, see equation (3.27), improves the effectiveness of the three incomplete factorizations, while the modified Olsen scheme, see equation (3.28), does not. This difference could be explained by the fact that the original

	PLAT362		EX2	
	MATVEC	time(sec)	MATVEC	time(sec)
diagonal	>5000	-	85	1.7
ILU0	>5000	-	40	10.2
ILUTP	>5000	-	35	10.0

Table 3.23: The Davidson method with the Olsen preconditioning scheme.

	PLAT362		EX2	
	MATVEC	time(sec)	MATVEC	time(sec)
diagonal	>5000	-	475	10.8
ILU0	>5000	-	355	21.2
ILUTP	>5000	-	405	24.6

Table 3.24: The Davidson method with the modified Olsen preconditioning scheme.

Olsen scheme produces a result that is orthogonal to the approximate eigenvector while the modified form does not produce a result with this orthogonality.

Table 3.25 shows some results of using the iterative linear system solvers as preconditioner. The maximum number of matrix-vector multiplication the linear system solver can use is limit to 200. The residual tolerance is set to be 1.6×10^{-5}. In this case, we are able to find the smallest five eigenvalues for both PLAT362 and EX2. When the matrix A is symmetric, three out of the four Jacobian matrices, J_R, J_I, and J_J, are symmetric. The remaining one, J_C, must be fairly close to be symmetric in most cases, because the table shows that CG performs fairly well on it. For these reason, later on we will only use CG as the solver. Comparing among the four different preconditioners, the Jacobi-Davidson scheme uses the least amount of time on PLAT362, the original Davidson scheme uses the least amount of time on EX2.

The contents of table 3.26 is similar to that of table 3.25, the difference is that a dynamic tolerance is used on the linear system solvers. In most instances, the dynamic scheme reduced the number of Davidson steps taken to reach convergence. But this achieved at the expense of using more matrix-vector multiplications in the preconditioners. Comparing table 3.26 and 3.25 we see that this trade-off often leads to reduction of total execution time and/or total number of matrix-vector multiplications. For example, in the fixed tolerance case, the execution time with CG on J_R as preconditioner takes 137.6 seconds to reach convergence for PLAT362, in the dynamic tolerance case, it reduces to 111.3 seconds.

	PLAT362			EX2		
	MATVEC (Davidson)	MATVEC (total)	time (sec)	MATVEC (Davidson)	MATVEC (total)	time (sec)
$A - \delta I$						
CG	466	84248	137.6	155	2223	12.1
$A - \delta I + xx^T$						
CG	506	91487	159.0	178	2742	15.2
$A - \delta I - 2x(Ax)^T$						
CG	475	86075	149.2	162	3076	16.5
GMRES(10)	2669	483160	1193.4	165	5062	29.9
$(I - xx^T)(A - \delta I)(I - xx^T)$						
CG	162	29562	58.7	485	83976	510.5

Table 3.25: Using iterative solvers with fixed tolerance as preconditioners to solve PLAT362 and EX2.

	PLAT362		
	MATVEC (Davidson)	MATVEC (total)	time (sec)
$A - \delta I$	250	69176	111.3
$A - \delta I + xx^T$	261	73498	126.0
$A - \delta I - 2x(Ax)^T$	270	77049	131.5
$(I - xx^T)(A - \delta I)(I - xx^T)$	285	35722	72.5
	EX2		
	MATVEC (Davidson)	MATVEC (total)	time (sec)
$A - \delta I$	55	1329	6.4
$A - \delta I + xx^T$	58	868	4.7
$A - \delta I - 2x(Ax)^T$	55	863	4.7
$(I - xx^T)(A - \delta I)(I - xx^T)$	485	92159	559.7

Table 3.26: Using CG with dynamic tolerance as preconditioner to solve PLAT362 and EX2.

3.5 Summary

In this chapter we discussed a number of Newton methods for eigenvalue problem. Through the discussion we identified pros and cons of the Davidson preconditioning scheme in connection to the correction equation. In order for the correction equation to generate nontrivial solution, we need to use the pseudoinverse. Computing the pseudoinverse is an impossible task in most large scale applications. However, an approximate solution can be reached through Krylov subspace methods if the linear system is consistent. Since the iteration matrix $(A - \lambda I)$ is usually not exactly singular, the Jacobi-Davidson preconditioning scheme forces it to be singular by explicitly orthogonalization. From a small number of experiments we have conducted, it appears that simply applying an iterative method on $(A - \lambda I)$ could work just as well as the Jacobi-Davidson scheme in some cases.

The correction generated by solving the correction equation with pseudoinverse is orthogonal to the exact eigenvector. The Olsen preconditioning scheme was designed explicitly to have this property. This orthogonality is apparently important to the success of the Olsen scheme. We tested a theoretically equivalent form of Olsen scheme which does not enforce this orthogonality property, see equation (3.28). The test results clearly indicate that the original Olsen scheme is more effective, see table 3.23 and 3.24.

The thrust of this chapter is to find a Newton iteration with a nonsingular Jacobian matrix in order to avoid the difficulty of operating with a ill-conditioned linear system for preconditioning. We identified 6 recurrences which could be considered as Newton methods for eigenvalue problems. Out of the 6, the augmented Newton method, the Jacobi-Davidson recurrence and the Newton-Olsen recurrence have been published before. The rest seems to be new methods. Even though the Jacobi-Davidson recurrence and the Newton-Olsen recurrence are not new, we presented them from a different point of view and proved their equivalence to the Rayleigh quotient iteration. A normalized augmented Newton method is derived through a simple modification to the augmented Newton method which is also equivalent RQI because it is simply a different implementation of the Newton-Olsen recurrence. Among the various Jacobian matrices, we have identified 3 of them, namely J_A, J_C, and J_I, to be nonsingular near convergence. When the Ritz pairs are not close to any eigenpairs, our experiments show that J_R, J_C and J_I tend to have fairly small condition numbers.

We have implemented a number of the preconditioners and conducted a small number of tests on them. The experiment reveals that using CG to solve the Newton recurrences as preconditioners performs well compared to other schemes.

Chapter 4

Practical Davidson Algorithm

In previous chapters, we identified the Arnoldi method and the Davidson method to be the most promising eigen-system solvers for our application among the methods compared. The overall objective of this study is to construct a preconditioned eigen-system solver, how well a method works with preconditioner is important to us. Compared with the different variations of the Arnoldi method, the Davidson method is usually better in taking advantage of preconditioners. Though we will not completely abandon other possibilities, the primary focus of this chapter is to discuss practical issues related to enhancing the performance of the Davidson algorithm for symmetric eigenvalue problems. In the implementation of different eigenvalue methods, the techniques discussed here will be applied to all methods wherever applicable, not just to the Davidson method.

Briefly, the eigenvalue problems we are facing have the following characteristics. The matrices are large, sparse and Hermitian. For the moment we will only consider real matrices, therefore the matrices are symmetric. These matrices are not stored explicitly because the eigen-system solvers under consideration only need to access them through matrix-vector multiplications which can be conveniently performed with the matrix stored in compact forms. In addition, cheap and effective preconditioners can be constructed for the problem without direct reference to the matrices. A typical eigenvalue problem might require the solution of a few hundred of the smallest eigenvalues and corresponding eigenvectors of a 100,000 $\times$ 100,000 matrix. Several sequences of eigenvalue problems may be solved in one simulation, where differences between two consecutive matrices diminish toward the end of each sequence. This makes the solution of the proceeding eigenvalue problem a good initial guess for the next one. Therefore, the eigenvalue solver should be able to take advantage of available initial guesses.

There are many issues which need to be addressed in constructing a practical Davidson eigen-system solver [31, 32, 137], many of which have been discussed in a review paper by Davidson [31]. Here we will only add a few remarks on some recent work that have strongly influenced our research. One focal point of our research is the preconditioning issue which has been reviewed in previous chapter. Another significant development in eigenvalue methods is the restarting techniques. Recent introduction of the Implicit Restarted Arnoldi (IRA) method has sparked renewed interest in restarting. Two restarting schemes for the Davidson method are of particular of interests here [86, 134]. The first one is due to Murray and colleagues, which saves Ritz vector of previous iteration in addition to the latest Ritz vectors [86]. The second one, called dynamic thick restart, is from Stathopoulos and co-authors. It dynamically decides the number of Ritz vectors to save when restart. Both of these schemes have been proven to work well in tests. We will study how will they work in computing a large number of eigenpairs. The computing environment has changed significantly since the introduction of the Davidson method. Part of this change is the emergence of parallel computers. There are a number of references on parallel and distributed Davidson method [40, 136]. Many issues related to parallel implementation are not unique to the Davidson eigenvalue method, there are many publications devoted to parallel and distributed computing issues which address a number of important aspects of a parallel Davidson method. The publications that are most directly related to our work are, [57, 107] on overall program structure for sparse computation, [99, 108] on matrix-vector multiplications design, and [117, 146] on parallel preconditioning.

The particular issues most important to our application includes how to compute a large number of

	N	NNZ	Eigenvalues
Si_2	1939	57289	15
Si_4	4451	165385	30
Si_6	7949	295099	60

Table 4.1: Size of the test problems.

eigenpairs efficiently, how to maintain a good orthonormal basis when there are many vector to orthogonalize, and how to restart. This chapter will address these issues one by one.

To test the techniques numerically, we will use a set of three small test matrices generated from the electronic structure simulation project, see table 4.1. These three matrices are generated from the first self-consistent iteration in the simulation of a two-silicon cluster Si_2, a four-silicon cluster Si_4, and a six-silicon cluster Si_6. The wave-function is taken to be zero, 6.8 atomic units away from the atoms. The core radius for nonlocal interaction is 2.8 atomic units, and the grid size is 1 atomic unit [60]. Table 4.1 shows the size of the matrices extracted and the number of eigenvalues to be computed. The average number of nonzero elements per row ranges from 29.5 for Si_2 to 37.2 for Si_6. For larger problems, we expect this average to be slightly above 37. The number of eigenvalues to be computed here is much more than needed to simulate structures indicated. However the ratio of eigenvalues over the matrix size is close to typical applications, see table 1.1. An eigenpair is considered converged if its residual norm is less than 10^{-6}. The maximum number of matrix-vector multiplications allowed is 100 per eigenpair. On average less than 30 matrix-vector multiplications are needed per eigenpair in most of our simulations, so this is a reasonable upper bound. In all experiments, the maximum basis size is limited to 20. Unless otherwise specified, 10 Ritz vectors corresponding to 10 smallest Ritz values are kept for restarting.

4.1 Program structure

After reviewing the basic operations performed in the eigenvalue methods considered, we noticed that they have the same basic blocks identified for Krylov linear system solvers [107]. They are

SAXPY, dot-product, matrix-vector multiplication, preconditioning.

To simplify the transition from a scalar program to a parallel program, we implement our eigenvalue routines in the Single-Program Multiple-Data (SPMD) paradigm [68]. Each long column vector is divided evenly among the processors if there is more than one. Excluding the matrix-vector multiplication and the preconditioner, the only difference in the body of the eigen-system solvers between a scalar version and a parallel version is the dot-product, $\alpha = x^T y$ [107]. In the parallel program, the dot-product can be performed in two step, first finding the dot-product of components of the long vectors that are on the same processor, then add up the partial sums from different processors and return the result to everyone. This second step is called a global sum. On most distributed environment the global sum is supported as one of the primitive functions. The global sum operation is often considered as an expensive operation, because all processors involved have to wait for each other. If one processor is lagging behind for any reason, the rest of the processors will be slowed down. To reduce the number of times the global sum operation is performed, a number of the dot products may be lumped together, and their partial sums are communicated in one global sum operation.

One of the central steps of the eigenvalue methods under consideration is the matrix-vector multiplication operation. Before building a eigenvalue routine for sparse matrices, a decision has to be made on the storage format of sparse matrices. Many sparse matrix storage formats exist [115], and in many cases, it is most convenient to store the matrix in a specialized compact form, see Chapter 1. One method of isolating the dependence of sparse matrix storage format from the eigenvalue routine is to use reverse communication [57, 107]. Since the eigenvalue algorithms under consideration only need to access matrices through matrix-vector multiplication, the reverse communication technique enables us to remove the effects of sparse matrix storage format from body of the eigenvalue routine. This data-hiding feature is an important characteristic

of Object-Oriented programming [140]. By removing the details of sparse matrix storage problem from the body of the eigenvalue routine, it also significantly simplifies the implementation.

An additional advantage of using reverse communication is that the preconditioning step can performed outside of the core of the eigen-system solver as well. This gives us the flexibility of switching between arbitrary preconditioners which makes testing of preconditioners considerably easier.

With reverse communication, an eigen-system solver may return to the caller for one of three reasons: termination, request for matrix-vector multiplication, or request for preconditioning. The reverse communication protocol we use is close to that of [107]. A parameter array `ipar` is used to carry the reverse communication information between the eigenvalue routine and the caller. Here is a piece of pseudo-code to demonstrate the design of the reverse communication protocol. The eigenvalue routine is named `pesdvd`, the matrix-vector multiplications routine is `matvec`, and the preconditioner is `preconditioning`.

```
10     call pesdvd(..., ipar, ...)
       if (ipar(1) == 1) then
          call matvec(...)
          goto 10
       else if (ipar(1) == 3) then
          call preconditioning(...)
          goto 10
       endif
```

Note that using 1 for matrix-vector multiplication and 3 for preconditioning is exactly the same as the design of P_SPARSLIB [107].

4.2 Computing large number of eigenvalues

The application of interests requires large number of eigenpairs, it is crucial for the eigen-system solver to be able to find a large number of eigenpairs effectively. This is a major issue we will address now.

There are a number of guidelines for considering solution to this problem. First, the workspace size should be independent of the number of eigenvalues wanted. Second, all initial guesses provided should be fully utilized. One technique that is most important to realizing these two requirements is called *locking*.

When computing a large number of eigenpairs, some of them will reach convergence earlier than others. The process of taking the converged ones out of the working set is referred to as locking. Once the converged ones are taken out, their corresponding eigenvectors are used in the orthogonalization procedure to make sure the active basis vectors are perpendicular to them. This is part of step 1 in algorithm 2.6 which orthogonalize the new vector to converged eigenvectors and existing basis vectors. Let m be the basis size, and p is the size of the working set of approximate eigenpairs. If the size of the working set is small, say, $p = 5$, it is possible to keep m relatively small, say $m = 20$, therefore requiring only a small work space. Once some of the p Ritz vectors reach convergence, they are taken out of the working set and new initial guess are put into the working set to replace the converged ones. This process repeats until enough eigenpairs are found.

In the Davidson method, at least one residual vector is computed at every step. Let V denote the basis vectors computed by the Davidson method. If the basis size m is small, it is often necessary to save $W \equiv AV$ in memory, and compute the residual vector as $r = Vy - \lambda Wy$ where (λ, y) is an chosen eigenpair of $H \equiv V^T AV$. In this case, only one matrix-vector multiplication is required to compute the Ritz pair and its residual vector after a new vector is added to the basis. If the basis size is large, computing Wy may be more expensive than perform a matrix-vector multiplication, in which case two matrix-vector multiplications per step would be used, one to compute a column of $V^T AV$, one to compute the residual.

Without locking, it would be necessary to keep all the converged eigenvectors in the basis as in [137]. The advantage of this scheme includes the larger basis size which could lead to faster convergence in some cases. In addition, the accuracy of eigenpairs converged earlier will be improved continuously which lead to smaller residual norms on some eigenpairs. In the type of eigen-systems of interest, the number of eigenpairs wanted is exceedingly large, say, 1000, which makes locking a necessity. Maintain orthogonality among 1000 vectors is a difficult task. Perform Rayleigh-Ritz projection on a basis of size 1000 is expensive both in terms

of memory requirement and arithmetic operation count. Because the basis size is so large, it is impractical to save AV. In this case, two matrix-vector multiplications are needed at every iteration of the Davidson method. In the distributed environment, the matrix-vector multiplication is the communication intensive operation. Using two matrix-vector multiplications per step is not a desirable option.

Another alternative could be that we keep work on all the unconverged eigenpair at once and lock the converged ones as they become ready. This was suggested in [31]. One advantage of this alternative is that the basis can be stored in the same space reserved to store the eigenvectors. Since the number of eigenvectors wanted is very large, the basis size may be too large initially. If less than a few hundreds of eigenpairs are wanted, then it is possible that the advantage of working with larger bases out-weigh the disadvantages. Because we design our program to compute more than a thousand eigenpairs, we have not favored this option.

If a group of eigenvectors is being processed, then a group of initial guesses is available at the start of the Davidson iterations. We could use a block version of the Davidson method, for example the Davidson-Liu variant [32]. However the block version is often not the most effective approach [31]. In addition, the number of vectors in the working set is usually too large to use as block size, see discussion about the block size in section 4.6. In [31], it was suggested that the basis V should be extended one vector at a time. The following algorithm takes all of these issues into consideration and makes two additional modifications to algorithm 2.5.

- It starts to build the basis V with b_0 starting vector, see also algorithm 2.6,
- it computes more than one Ritz pair in the Rayleigh-Ritz procedure. In the following algorithm it is referred to as the active window size, p.

Both of these modifications were mentioned in [31]. The following algorithm combines them to compute a large number of eigenpairs by working on a small active set at a time.

ALGORITHM **4.1 Restarted Davidson's algorithm** *for finding the d smallest eigenvalues and corresponding eigenvectors, $\lambda_i, x_i, i = 1, \ldots, d$, of a symmetric matrix A.*

1. **Start**. *Choose d initial vectors, $x_1, x_2, \ldots, x_d$. Choose the active window size p to be an integer less than m, say $p = m/4$. Let $c = 0$.*

2. **Iterate** *to build an orthonormal basis $V_m = [v_1, v_2, \ldots, v_m]$ that is orthogonal to $X_c \equiv [x_1, x_2, \ldots, x_c]$, where c is the number of eigenpairs converged.*

   Fill array Z with first p available unconverged eigenvectors. If there are less than p eigenvectors left, reduce p to the actual number of eigenvectors remained. Choose an initial basis size b_0, fill columns $p+1$ to b of Z with Ritz vectors of previous iteration if available. Let b to be the actual number of column vectors in Z. Let $j = 0$.

   (a) Orthogonalization.

$$\begin{aligned} Z &= (I - V_j V_j^T - X_c X_c^T) Z, \\ QR &= Z, \\ [v_{j+1}, \ldots, v_{j+b}] &= Q. \end{aligned}$$

   (b) $w_i = A v_i, i = j+1, \ldots, j+b.$

   (c) $h_{ik} = v_i^T w_k$, *for* $i = 1, \ldots, k; k = j+1, \ldots, j+b.$
   $j = j + b$, $b = b_1$.

   (d) **if** *$(j < m)$ then perform preconditioning,*

      i. compute the smallest eigenvalue and corresponding eigenvector of $H_j = V_j^T W_j$, say, λ, y.

      ii. $r = W_j * y - \lambda V_j * y$, *where* $V_j = [v_1, \ldots, v_j]$, $W_j = [w_1, \ldots, w_j]$.

      iii. $Z = (\mathrm{diag}(A) - \lambda I)^{-1} r$.

	Arnoldi		Orthogonalized		Davidson		Harmonic	
d	MATVEC	time	MATVEC	time	MATVEC	time	MATVEC	time
15	504	139.7	485	149.4	443	152.6	484	209.2
30	1519	491.8	1289	454.2	1246	490.7	1409	713.6
60	5399	2143.6	5049	2182.5	3856	1802.1	5219	3395.9

Table 4.2: Finding different number of eigenvalues from Si_6 matrix.

	Arnoldi		Orthogonalized		Davidson		Harmonic	
	MATVEC	time	MATVEC	time	MATVEC	time	MATVEC	time
15→30	3.0	3.5	2.7	3.0	2.8	3.2	2.9	3.4
30→60	3.6	4.4	3.9	4.8	3.1	3.7	3.7	4.8
15→60	10.7	15.3	10.4	14.6	8.7	11.8	10.8	16.2

Table 4.3: Increase in MATVEC and time versus increase in number of eigenvalues.

iv. Goto 2.a.

else *continue to next step.*

3. **Rayleigh-Ritz procedure**. *Let $H_m = (h_{ij})_{m\times m}$, μ_i, $i = 1, \ldots, p$, be the p smallest eigenvalues of H_m, y_i be the corresponding eigenvectors, i.e., $H_m y_i = \lambda_i y_i$. Let $Y = [y_1, \ldots, y_p]$,*

$$V_p = V_m Y, \quad W_p = W_m Y, \quad H_p = \mathrm{diag}(\mu_1, \mu_2, \ldots, \mu_p). \tag{4.1}$$

The new approximation to the eigenvalue and the eigenvectors are,

$$\begin{aligned} \lambda_{c+i} &= \mu_i; \quad i = 1, \ldots, \min(p, d-c); \\ x_{c+i} &= v_i; \\ r_{c+i} &= w_i - \lambda_{c+i} x_{c+i}. \end{aligned}$$

4. **Convergence test**.

If $\|r_{c+1}\| < \tau$, increment c by 1 and test next Ritz pair.

If $c < d$, return to step 2.

This algorithm computes d eigenvalues by working on p of them at a time which is an extension to what is proposed in [31], where the algorithm would only compute p eigenvalues. It should be trivial to extend the Arnoldi method or any other eigenvalue method mentioned in the previous chapter to compute large number of eigenpairs in a similar fashion.

In the Rayleigh-Ritz procedure of the above algorithm, we order the eigenvalues of H_m from small to large. In which case, the p smallest eigenvalues are simply the first p eigenvalues. If the eigenvalues are in descending order instead, the above procedure can be used to compute the largest eigenvalue. Similarly, if the eigenvalues of H_m are ordered according to the distance from a given number, the above algorithm can be used to compute eigenvalues around that chosen number.

Table 4.2 shows the number of matrix-vector multiplications and execution time of four different eigenvalue methods namely the Arnoldi method, the orthogonalized Arnoldi method, the Davidson method and the harmonic Davidson method, see Chapter 2 for more detailed description. The MATVEC column shows the number of matrix-vector multiplications used. The time column shows the execution time in seconds. Only the Si_6 matrix is used in this test. No preconditioning is used. Other parameters used are as follows,

$$m = 20, \quad p = 5, \quad b_0 = 10, \quad b_1 = 1.$$

	Arnoldi		Orthogonalized		Davidson		Harmonic	
	MATVEC	time	MATVEC	time	MATVEC	time	MATVEC	time
No preconditioning								
Si_2	512	29.4	434	27.2	334	24.6	443	58.1
Si_4	1429	243.2	1400	262.7	1177	240.3	1399	482.9
Si_6	5399	2143.2	5049	2182.5	3856	1802.1	5219	3395.9
Diagonal scaling								
Si_2	1164	82.1	1494	118.8	627	54.8	1134	126.4
Si_4	2719	492.1	-25	583.6	1778	392.7	2779	791.6
Si_6	-51	2277.7	-42	2325.9	5064	2367.7	-51	3497.5
SOR preconditioner								
Si_2	-7	145.7	944	104.2	244	43.1	-8	1005.5
Si_4	-15	787.4	1929	597.4	573	266.3	-15	7170.7
Si_6	-28	3047.4	4069	2573.7	1497	1221.0	-32	26871.5
ILU0 preconditioner								
Si_2	-4	165.7	-6	175.3	685	107.8	-7	966.7
Si_4	-7	850.6	-6	839.1	-20	1076.6	-11	7375.0
Si_6	-15	3132.3	-12	3091.1	-38	4318.9	-22	29063.7

Table 4.4: The eigenvalue routines with different preconditioning.

On average, when the number of eigenvalues doubles, the number of matrix-vector multiplications increases about 3–4 times, the CPU time increases about 3–5 times, see table 4.3. Overall, the increase in time and matrix-vector multiplications is slower for the Davidson method compared to the other three. It is interesting to note that the Arnoldi method uses the least amount of time to compute 15 eigenvalues, the orthogonalized Arnoldi method uses the least amount of time to compute 30 eigenvalues, and the Davidson method uses the least amount of time to compute 60 eigenvalues. Since no preconditioning is used, the four methods shown in table 4.2 are equivalent to each other. Because the Arnoldi method is the least expensive per step, it usually uses the least amount of time to execute the same number of steps. When the number of eigenpairs wanted is small, there are less opportunity for round-off errors to erode the stability of the Arnoldi method. Thus the number of steps taken by the Arnoldi method and others are almost the same. In this case, the Arnoldi method will use less time to compute the same solution. The orthogonalized Arnoldi method is more expensive and numerically more stable than the Arnoldi method. In turn, the Davidson method is more expensive than the orthogonalized Arnoldi method and even resistant to round-off errors which might impede the convergence. When computing 60 eigenpairs, the Davidson method uses significantly less matrix-vector multiplications than other three. This test indicates that as we increase the number of eigenvalues computed, the Davidson method becomes more competitive compared with the Arnoldi method.

Table 4.4 shows how the eigenvalue routines work with different preconditioners. The maximum number of matrix-vector multiplication 100 per eigenpair. If the MATVEC column has a negative number it indicates the method did not reach convergence on all wanted eigenvalues within the allowed number of matrix-vector multiplications. The absolute value of the number in the cell indicates the number of eigenvalues actually converged. The corresponding time column is the time used. For example, with SOR preconditioner, the Arnoldi method computed only 28 eigenpairs of the Si_6 matrix using 6000 matrix-vector multiplications. The corresponding time, 3047.4 seconds, is the time taken to computed these 28 eigenpairs. In every case where more than one method reached convergence, the Davidson method uses less time than the others. The results in this table mirrors what we have observed with the Harwell-Boeing test matrices in the previous chapter. The preconditioners do not improve the performance of the eigen-system solvers except the Davidson method. Even for the Davidson method only the SOR preconditioner uses less time on the Si_6 case compared to the unpreconditioned case. A better preconditioner is necessary for this type of eigenvalue problems.

4.3 Restarting

The Lanczos method, the Arnoldi method and the Davidson method all require restarting after a certain number of steps. The same reasons will also require restarting of algorithm 4.1, because as the number of steps increases, the space required to store V_j increases proportionally. This creates many difficulties, such as, storage problem, and loss of orthogonality. As j increases, the complexity of Gram-Schmidt orthogonalization procedure increases as $O(nj^2)$. The complexity of performing both Rayleigh-Ritz projection and harmonic Ritz projection increase as $O(j^3)$. For these reasons, the maximum basis size is usually very small. In most cases, the maximum basis size we use is only 20 or 30.

The restarting issue has been addressed in many papers [31, 70, 150]. One natural choice is to start with Ritz vectors. This is what is done in algorithm 4.1. In [31], Davidson suggested that one should save more than d Ritz vectors if d eigenvectors are sought. Exactly how many was a question left unanswered. Observations of the convergence properties of the Conjugate Gradient method motivated the authors of [150] to save the Ritz vectors from the last two steps as the initial guesses. This scheme is quite successful in some cases. A relatively new scheme named "Implicit Restarting" due to Sorensen restarts with a set of Schur vectors rather than Ritz vectors [131]. One advantage of using Schur vectors instead of Ritz vectors is that Schur vectors always exist even for defective matrices but Ritz vectors do not. In addition, Schur vectors can be real in case the matrix has complex eigenvalues or eigenvectors. Both of these advantages are only applicable if the matrix is nonsymmetric. In the symmetric or Hermitian case, a Schur vector differ from a Ritz vector only by a scalar constant [83]. For this reason, only Ritz vectors are considered for restarting in the symmetric case.

Before discussing the details of our restarting scheme, we review guidelines for restarting.

- The first and most important requirement on restarting scheme is that the quality of the approximate solution should not decrease after restarting. For Hermitian eigenvalue problems, residual norm is a good measure of the quality of approximation, in which case, the residual norm should not increase after restarting. The simplest way of achieving this is to include the latest Ritz vectors in the basis, see algorithm 4.1 and [137].

- Preserve the Ritz vectors near the wanted eigenvectors at restart. This was suggested as a feature for the Davidson method [31]. A formal justification for this technique on the Arnoldi method was given by Morgan [83]. Keeping Ritz vectors with Ritz values close to the desired one has the effect of increasing the separation between the wanted eigenvalue and the rest of the spectrum, therefore increasing the convergence rate. A variation of this scheme saves Schur vectors instead of the Ritz vectors is implemented in ARPACK [129].

- Preserve enough information to prevent repetition. Without preconditioning, many eigenvalue methods are essentially performing Rayleigh-Ritz projection on an orthonormal basis generated from a power series. These methods always favor the extreme eigenvalues, if some Ritz vectors corresponding to extreme eigenvalues are discarded, they will likely to reappear. Thus one technique for avoiding repetition is to keep the Ritz vectors corresponding to the extreme eigenvalues in the basis. In [72], a technique for preserving unwanted eigenvectors is discussed. In the linear system case, the Augmented Krylov subspace technique is based on the same principle [18, 106]. Based on the three-term recurrence of the Lanczos method, Murray and co-authors suggest preserving Ritz vectors from an earlier iteration can avoid repetition [86].

- Restart based on convergence predictions rather than heuristics, for example, the dynamic "thick restart" scheme [134].

The intuition on the number of restarting vectors is that the more information saved the more likely later iterations will not repeat the previous work; the more steps taken before restarting, the more new information can be incorporated into the solutions, i.e., more accurate the next approximation solution will be. In considering the restarting scheme, we assume the goal of the eigen-system solver is to make one eigenpair converge at any given time. With this in mind, the question on what to do to restart includes two components:

1. how many *nearby* Ritz vectors to save?

2. whether or not to save the Ritz vector of previous iteration?

On the first question, until recently, the general guideline has been to keep all the Ritz vectors corresponding to wanted eigenvectors plus a small number of near by ones [31, 129]. Recently, Stathopoulos and colleagues proposed a dynamic scheme to determine the number of Ritz vectors to be saved [135]. The proposed scheme is based on the observation that the saved Ritz vectors in the basis approximately deflate the spectrum [83]. To determine how many Ritz vectors to save, the dynamic thick restarting scheme uses the Ritz values to approximate the gap ratio after deflation. Saving many vectors for restarting is called *thick restart* [135]. This section describes a modified form of the dynamic thick restarting scheme.

To simplify the description, we will only describe the case where the smallest eigenvalue is wanted. It is straightforward to extend this to the case where more than one eigenpair is sought or the largest eigenvalue is needed. In [83], it was shown that the convergence rate of the thick-restarted Arnoldi method is approximated proportional to $\gamma = (\lambda^*_{t+1} - \lambda^*_1)/(\lambda^*_n - \lambda^*_1)$ assuming λ^*_1 is the smallest eigenvalue, λ^*_n is the largest eigenvalue, and t is the number of *nearby* Ritz vectors saved at restarting. The super script $*$ indicates they are exact eigenvalues. Since the exact eigenvalues are unknown, the gap ratio γ is approximated as follows

$$\gamma \approx (\lambda_{t+1} - \lambda_1)/(\lambda_m - \lambda_1) \tag{4.2}$$

where m is the basis size, i.e., the number of Ritz values computed by the Rayleigh-Ritz projection. In this case, only the Ritz vectors corresponding to the t smallest Ritz values are saved. In [135], the authors extended this to saving Ritz vectors corresponding to both the largest and the smallest Ritz values. If t_L Ritz vectors corresponding to the smallest Ritz values and t_R Ritz vectors corresponding to the largest Ritz values are saved at restart, the convergence rate of Arnoldi method is expected to be proportional to $\gamma \approx (\lambda_{t_L+1} - \lambda_1)/(\lambda_{m-t_R+1} - \lambda_1)$. In [135] this estimate of the gap ratio is used to determine the optimal t_L and t_R. This strategy can be used for all projection eigen-system solvers including the Davidson method and the implicitly restarted Arnoldi method. The reduction in the number of matrix-vector multiplications with this dynamic restart scheme is significant in many cases.

If a total of $t = t_L + t_R$ Ritz vectors are saved for restarting, computing these t Ritz vectors from V_m requires roughly $2tmn$ floating-point operations, or FLOP for short. Note that n is the dimension of the vectors. We represent this step by $V_t \Longleftarrow V_m Y$, where Y contains the selected eigenvectors of H_m. To avoid matrix-vector multiplications, we can also recombine columns of W_m to form the first t columns of $W_t = AV_t$, $W_t \Longleftarrow W_m Y$. This again requires about $2tmn$ FLOP. The residual vector corresponding to the smallest Ritz value needs to be computed for testing convergence which needs about $4n$ FLOP. The total floating-point operation count for extending the basis size from t to m is about

$$\sum_{j=t}^{m-1} (10j + 6)n,$$

excluding the matrix-vector multiplications and preconditioning operations. The average FLOP count per step is about

$$\frac{4(mt+1)n}{m-t} + 5(m+t)n + n + op(A) + op(M), \tag{4.3}$$

where $op(A)$ denote the operation count in a matrix-vector multiplication and $op(M)$ denotes the operation count in a preconditioning operation. Based on equation (4.2), the larger t is, the faster the eigenvalue method will converge. However the larger t is, the more expensive an average Davidson iteration is. To achieve convergence in the shortest amount of time, it is necessary to balance the two factors.

Note that when t is close to m, the estimated γ by equation (4.2) becomes invalid. However, since the dynamic scheme is shown to be effective by numerical tests, we will continue to use this formula [135]. One slight modification to equation (4.2) is that instead of use λ_m computed at the current step, the maximum of all λ_m ever computed is used. This modification makes the γ computed slightly closer to the actual gap ratio.

When restarting to build a new basis, if it is not the first time step 2 is entered, there is a significant amount of information not utilized in Algorithm 4.1. For example, the vectors in V are orthonormal, so it

	Arnoldi		Orthogonalized		Davidson		Harmonic	
	MATVEC	time	MATVEC	time	MATVEC	time	MATVEC	time
gap ratio only								
Si_2	314	52.7	307	50.8	306	54.4	331	64.2
Si_4	1154	433.7	1240	483.7	1095	439.8	1367	595.8
Si_6	4470	3711.9	4657	4014.9	5351	4315.9	4958	4912.8
gap ratio and work								
Si_2	600	27.3	508	24.8	351	21.8	-2	109.3
Si_4	2153	319.1	2131	347.0	1582	291.3	-1	549.1
Si_6	-53	2043.9	-52	2195.2	5427	2291.0	-6	2201.4

Table 4.5: Effects of dynamic restarting on the test problems.

is not necessary to orthogonalize them again. If no Ritz pair reached convergence in the previous iteration, there are p Ritz vectors at the beginning of array V, see step 3. If the first p columns of V do not need to go through the orthogonalization step, then the corresponding columns of W need not undergo step 2.b either. Similarly, step 2.c can be skipped because step 3 has put the appropriate data in H. If some Ritz pairs reached convergence, we could place the unconverged Ritz vectors at the beginning of V, and permute W and H to avoid performing step 2.a–2.c on these unconverged Ritz vectors.

Table 4.5 shows the results of the two dynamic restarting schemes applied to the test problems. The upper half of the table shows the results of using the dynamic thick restart. Generally speaking, this scheme based on gap ratio tends to decrease the total number of matrix-vector multiplications required. It does so by saving most of the Ritz vectors when restarting. In fact, a upper bound on the thickness t is needed to prevent it from become m. For example, if the basis size is 20, this dynamic scheme doesn't allowed more than 17 Ritz vectors to be saved. Most of the time, this scheme will save the maximum allowed number of Ritz vectors. This method is expensive because recombining the basis to form a large number of Ritz vectors is an expensive operation.

The second scheme we implemented tries to find the maximum value for the following quantity

$$(\gamma - \log(FLOP/n))$$

where $FLOP$ refers to equation (4.3). Table 4.5 shows that this new scheme generally uses more matrix-vector multiplications than the fixed restarting scheme and the dynamic thick restart from [135]. Because it restarts with fewer Ritz vectors it is less expensive per matrix-vector multiplication than the other two. From the table we can see that even though more matrix-vector multiplications are needed, it still uses less time in many case.

Comparing table 4.5 with the unpreconditioned case in table 4.4, we notice that the dynamic scheme reduces the total execution time in the Si_2 case, but in the Si_6 case, the fixed-size thick restart scheme uses less time than both dynamic restarting schemes in computing 60 eigenpairs.

The previous Ritz vector is appended to the regular Ritz vectors saved for restarting if it is used. For example, if the restarting scheme decides to save 10 Ritz vectors, the Ritz vector of the previous step is appended as the 11th vector in V. Let y_- be the eigenvector of H_{m-1} corresponding the smallest eigenvalue of H_{m-1}. This 11th vector is $V_{m-1}y_-$.

When the old Ritz vector is used in the basis, this old Ritz vector is not orthogonal to the p new Ritz vectors. A fairly inexpensive way of ensuring the new basis is orthonormal is to ensure that $[y_1, \ldots, y_p]$ is orthogonal to y_-. Since y_- has one element less than $y_1, \ldots, y_p$, we need to augment y_- to the same size as the others. After this, the Gram-Schmidt procedure can be applied to make $\binom{y_-}{0}$ orthogonal to $[y_1, \ldots, y_p]$. One characteristic of this orthogonalization worth mentioning is that it is performed on a short vector, in other word, it is inexpensive. In addition, no data communication is necessary in distributed computing environments. Denote the resulting vector by y_a, $y_a = (I - Y_p Y_p^T)\binom{y_-}{0}/\beta$, where $Y_p = [y_1, \ldots, y_p]$, $\beta = \|(I - Y_p Y_p^T)\binom{y_-}{0}\|$. This vector can be append to Y_p and used in step 3 of algorithm 4.1 as if p has been incremented by 1. V_p and W_p in equation (4.1) can be computed as follows,

$$V_{p+1} = V_m[y_1, \ldots, y_p, y_a], \quad W_{p+1} = W_m[y_1, \ldots, y_p, y_a].$$

	Arnoldi		Orthogonalized		Davidson		Harmonic	
	MATVEC	time	MATVEC	time	MATVEC	time	MATVEC	time
thick restart								
Si_2	512	29.5	434	27.2	334	24.5	-3	133.2
Si_4	1429	242.3	1400	262.7	1177	238.9	-3	665.4
Si_6	5419	2140.9	5049	2169.1	3856	1786.7	-3	2466.9
dynamic restart – gap ratio								
Si_2	314	43.9	307	41.4	306	46.7	-4	256.3
Si_4	1152	365.3	1255	417.7	1080	384.1	-11	1182.8
Si_6	5140	3677.1	4708	3489.2	5335	3806.4	-16	4575.6
dynamic restart – gap ratio and work								
Si_2	600	27.3	508	24.7	351	22.1	-2	109.3
Si_4	2153	320.2	2131	348.0	1582	295.0	-2	558.9
Si_6	-53	2076.1	-52	2228.8	5427	2311.1	-6	2228.3

Table 4.6: Effects of saving previous Ritz vector on the test problems.

By definition, $H_{p+1} = V_{p+1}^T A V_{p+1}$, since the first p vectors in V are Ritz vectors, the corresponding part of H_{p+1} is diagonal, only the last column needs special attention. Let $v_{p+1} = V_m y_a$, where V_p denotes the first p columns of V_{p+1}, the first p elements of the last column of H_{p+1} is

$$V_p^T A v_{p+1} = Y_p^T V_m^T A V_m y_a = Y_p^T H_m y_a = \begin{pmatrix} \mu_1 & & \\ & \ddots & \\ & & \mu_p \end{pmatrix} Y_p^T y_a = 0.$$

Thus the matrix H_{p+1} is also diagonal, the last diagonal element is a Rayleigh quotient of v_{p+1}, $\mu_{p+1} = v_{p+1}^T A v_{p+1} = y_a^T H_m y_a$.

Table 4.6 shows the results of saving one previous Ritz vectors in addition to the normal restarting scheme. No preconditioning is used here in this test. In many cases, the same number of matrix-vector multiplications are needed with previous vector compared to not using the previous Ritz vector. The observation here is that the previous Ritz vector does not help the type of eigenvalue problem of interest. This could be explained as the high-end of the spectrum is not significantly easier to compute than the smallest ones. For example, to compute the 15 largest eigenvalues of Si_2, it takes the Davidson method 213 matrix-vector multiplications which is about two thirds of the number of matrix-vector multiplications needed to compute 15 smallest eigenvalues. Thus the discarded Ritz vectors corresponding to the largest eigenvalues will not emerge very quickly.

4.4 Orthogonalization

The most straightforward way of implementing the orthogonalization step of the eigenvalue algorithm 4.1 is the Classic Gram-Schmidt (CGS) procedure.

ALGORITHM **4.2 The classic Gram-Schmidt procedure** *to make a new vector z orthogonal to an orthonormal set of vectors $[v_1, v_2, \ldots, v_j]$.*

1. $c = [v_1, v_2, \ldots, v_j]^T z$;
2. $z = z - [v_1, v_2, \ldots, v_j] c$.

This algorithm is simple, but experience shows that it is not very stable [51]. A more stable alternative is the following Modified Gram-Schmidt (MGS) procedure.

ALGORITHM **4.3 The modified Gram-Schmidt procedure** *to make a new vector z orthogonal to an orthonormal set of vectors $[v_1, v_2, \ldots, v_j]$.*

For $i = 1, \ldots, j$, do

1. $c_i = v_i^T z$;

2. $z = z - v_i c_i$.

The main difference between the CGS and MGS is that CGS performs the j dot-products together while MGS does it one after the other. On distributed environment, a global sum operation needs to be inserted between step 1 and 2 of the above two algorithms. As have discussed before, the global sum operation requires every processor to synchronize, i.e., there is a significant start-up time associated with each global sum operation. The Classical Gram-Schmidt procedure allows one to combine j dot-products to reduce communication time by reducing the number of times the global sum operation is performed. Therefore CGS is more suited for distributed environment. To increase the quality of the solution from CGS, reorthogonalization is performed under certain conditions [29]. The CGS with reorthogonalization can be stated as follows [29].

ALGORITHM **4.4 The Classic Gram-Schmidt procedure with reorthogonalization** *to make a new vector z orthogonal to an orthonormal set of vectors $[v_1, v_2, \ldots, v_j]$.*

0. $\zeta = \|z\|$;

1. $c = [v_1, v_2, \ldots, v_j]^T z$;

2. $z = z - [v_1, v_2, \ldots, v_j]c$;

3. *if* $\|z\| < \alpha\zeta$ *go to step 0; else continue.*

4. $z = z/\|z\|$.

It is clear that the larger the α is, the more likely it is that the Gram-Schmidt procedure is repeated. Any number between 0.1 and 0.7 can be considered as reasonable. We use 0.1 throughout this thesis unless otherwise specified. A normalization step is appended to this algorithm because this procedure is intended to generate an orthonormal basis.

To increase the efficiency of the program on parallel environment, algorithm 4.4 can be slightly modified to remove the computation of $\|z\|$ at step 0. The vector c in the following algorithm is a short vector that resides in every processor, thus computing $c^T c$ does not involve global communication. The test in step 3 in the following algorithm is exactly equivalent to step 3 in algorithm 4.4 because $[v_1, v_2, \ldots, v_j]$ is orthonormal.

ALGORITHM **4.5 The classic Gram-Schmidt procedure with re-orthogonalization** *to make a new vector z orthogonal to an orthonormal set of vectors $[v_1, v_2, \ldots, v_j]$.*

1. $c = [v_1, v_2, \ldots, v_j]^T z$;

2. $z = z - [v_1, v_2, \ldots, v_j]c$;

3. *if* $c^T c > (\alpha^{-2} - 1) z^T z$ *go to step 1; else continue.*

4. $z = z/(z^T z)^{1/2}$.

A natural way of extending algorithm 4.5 to handle multiple vectors is to orthonormalize each new vector one after another with the newly orthonormalized vector appended to $[v_1, v_2, \ldots, v_j]$ immediately. This requires at least two global sums for each new vector. Since all the new vectors are available at the same time, it is trivial to orthogonalize the new vectors against the old ones by extending algorithm 4.2 to a block version. As is suggested in algorithm 4.1, the normalization step of the above algorithm should be replaced by a QR factorization. Denote the new vectors by Z, $Z = [z_1, z_2, \ldots, z_b]$. To fully exploit different possibilities, consider a more general decomposition, $Z = QB$ where $Q \in \mathbf{R}^{n \times b}$ is an orthonormal matrix and $B \in \mathbf{R}^{b \times b}$ is an arbitrary real matrix. Let $S \equiv Z^T Z$. To compute $Q = ZB^{-1}$, we need to compute B^{-1}. If Z is full rank, S is a symmetric positive definite matrix. A Cholesky factorization can be used to compute an upper triangular matrix B [51]. In general S is symmetric positive semidefinite, a more stable decomposition is the eigenvalue decomposition, i.e., $S = YLY^T$ where columns of Y are eigenvectors of S, and L is a diagonal matrix where each diagonal element is an eigenvalue of S. Denote the eigenvalue by λ_i, $\lambda_i = L_{ii}$, $i = 1, \ldots, b$. Assume there are r nonzero eigenvalues and they are in descending order, i.e.,

$\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_r, > \lambda_{r+1} = \ldots = \lambda_b = 0$. For stability reason, Q is actually computed by replacing B^{-1} with the nonsingular part of the pseudoinverse B^+, i.e.,

$$Q = ZY \begin{pmatrix} \lambda_1^{-1/2} & & 0 \\ & \ddots & \\ 0 & & \lambda_r^{-1/2} \\ & 0 & \end{pmatrix}.$$

Now that Q has only r columns. We will replace the first r columns of Z with them. In most cases, the eigenvalue routine can continue with only one new vector added to the basis, thus as long as $r > 0$, the orthogonalization can stop.

The process just described essentially computes a SVD decomposition of Z, $Z = QL^{1/2}Y$ [51]. The approach taken here is not a stable technique for computing the SVD decomposition, the main reason for using it is to reduce interprocessor communication in distributed environment. Using 64-bit IEEE arithmetic, the unit round off error is about 2.2×10^{-16}, any eigenvalue of S that is smaller than $2.2 \times 10^{-16}\lambda_1$ should be considered unreliable. This algorithm uses about twice as many floating-point operations as that of a Gram-Schmidt approach. If the new vectors are known to be linearly independent, then the Cholesky factorization of S can be performed to reduce the complexity of computing ZB^{-1}. Even though the Cholesky approach is less expensive in floating-point arithmetic operations, it is still more expensive than the Gram-Schmidt approach. This is the price for reducing data communication. In most cases this orthogonalization procedure receives only one new vector at a time, in which case, this SVD procedure reverts back to simply normalizing the vector z as in step 4 of algorithm 4.5.

The columns of Z may be badly scaled, in which case, the accuracy of the eigenvalues of S could be compromised. To avoid this difficulty, the diagonal elements of S are scaled to 1 through a symmetric scaling that is equivalent to scaling the columns of Z to norm 1. Let Y_r denote the first r columns of Y. It is clear that (ZY_r) has r orthogonal columns, the norm of these columns are $\lambda_1^{1/2}, \ldots, \lambda_r^{1/2}$. If $\lambda_i^{1/2}$ is small, which means the b unit vectors combine to form a small vector, the cancelation error may be serious in this case. It is necessary to recompute this column of Q.

A block version of the Gram-Schmidt procedure can be described as follows.

ALGORITHM **4.6 Block Gram-Schmidt procedure** *for orthonormalizing* $Z = [z_1, \ldots, z_b]$ *against* $V_j = [v_1, \ldots, v_j]$ *where* $V_j^T V_j = I$.

1. $C = V_j^T Z$;
2. *perform global sum on all elements of* C;
3. $Z = Z - V_j C$;
4. $S = Z^T Z$;
5. *perform global sum on all elements of* S;
6. *let* c_i *denote the* i*th column of* C, s_{ij} *denote the element of* S *at the* i*th row and* j*th column, and set flag* `REORTH` *to 0;*

   for $i = 1, \ldots, b$,
   if $c_i^T c_i > (\alpha^{-2} - 1)s_{ii}$, `REORTH = 1`;

7. *let* $\zeta_i = s_{ii}^{1/2}$, $i = 1, \ldots, b$, $\hat{S} = \begin{pmatrix} \zeta_1^{-1} & & \\ & \ddots & \\ & & \zeta_b^{-1} \end{pmatrix} S \begin{pmatrix} \zeta_1^{-1} & & \\ & \ddots & \\ & & \zeta_b^{-1} \end{pmatrix}$;
8. *eigenvalue decomposition,* $\hat{S} = YLY^T$;

# vectors	CGS	SVD
1	0.23	0.23
2	0.49	0.34
4	1.06	0.45

Table 4.7: Execution time of the orthogonalization routines.

	CGS	SVD
Si_2	24.8	24.5
Si_4	241.6	238.9
Si_6	1829.1	1790.8

Table 4.8: Execution time of the Davidson method with different orthogonalization scheme.

9. *assume the first r eigenvalues of S, $\lambda_1, \ldots, \lambda_r$, are nonzero, and $Y_r = [y_1, \ldots, y_r]$ be the corresponding eigenvectors, replace the first r columns of Z with*

$$Z \begin{pmatrix} \zeta_1^{-1} & & \\ & \ddots & \\ & & \zeta_b^{-1} \end{pmatrix} Y_r \begin{pmatrix} \lambda_1^{-1} & & \\ & \ddots & \\ & & \lambda_r^{-1} \end{pmatrix};$$

10. *if* `REORTH` $= 0$*, stop, else, go back to step 1.*

Table 4.7 shows the time orthogonalize three different sets of new vectors on a SGI R10000 workstation. The vector dimension is 100,000. The new vectors need to orthogonalize against 14 orthonormal vectors. As number of new vectors doubles, the execution time of CGS increase by a factor slightly larger than 2. However, the time used by the SVD based orthogonalization routine increases considerably slower despite the fact that SVD uses more floating-point operations than CGS. The primary reason for the saving in execution time is that the SVD based orthogonalization uses BLAS-3 operation as oppose to BLAS-2 operations used by CGS. For every memory access, considerably more floating-point operations are performed in the SVD based orthogonalization routine.

Table 4.8 shows the comparison of two different orthogonalization routines on the test problems. In the Davidson method, there are only a small number of cases where the orthogonalization is performed on more than one vectors. Therefore the difference between using CGS and SVD orthogonalization is not significant.

4.5 Targeting

In the preconditioning step of algorithm 4.1, it is possible to generate j residual vectors. Since only one will be used, there is a choice to be made on which residual to be used. The process of making this choice is referred to as targeting, because the eigen-pair whose residual is chosen will benefit the most from preconditioning. The preconditioning equation is of the form $(M - \delta)z = r$. Thus, along with the residual vector, it is also necessary to choose an appropriate shift δ. In our implementation of targeting, the shift is chosen to be the biased estimate of the eigenvalue corresponding to the residual vector [133]. Some of the common targeting schemes are as follows [31].

1. **Sloppy targeting**. Always target the first unconverged one. For example, if the smallest eigenvalues are wanted, this scheme always targets the smallest unconverged Ritz value. This is the scheme shown in algorithm 4.1.

2. **Minimum residual targeting**. Target the Ritz pair with the smallest residual norm. The rationale is that it will converge quickly and be locked out of the basis. Disadvantages of this scheme is that residual norms are needed at every step. Since residual norms can not be estimated conveniently in

	Sloppy		Minimum		Maximum	
	MATVEC	time(sec)	MATVEC	time(sec)	MATVEC	time(sec)
Si_2	334	24.6	411	30.8	686	83.9
Si_4	1177	240.3	1261	255.1	1020	209.6
Si_6	3856	1802.1	4355	2015.3	4020	1852.9

Table 4.9: Comparison of different targeting schemes.

the Davidson method as in the Arnoldi method. A significant amount of work is required to compute all residual norms. In most of our applications, extreme eigenvalues are wanted, this targeting scheme will encourage convergence of interior eigenvalues if one happens to have a small residual norm.

3. **Minimum improvement targeting**. Because the residual norms are not always available the Minimum Residual scheme is usually not a viable option. For the unpreconditioned Arnoldi and Lanczos there is a direct proportional relation between the ith residual norm and the last element of the eigenvector y_i. At step j, one could choose the ith Ritz pair corresponding to the smallest $|y_{ji}|$ as the target. The value y_{ji} indicates to how much the jth basis vector v_j is used in forming the ith Ritz vector, x_i. If $|y_{ji}|$ is small, the contribution of the newest basis vector v_j to x_i is small which can be interpreted as the improvement to x_i is small. Thus this targeting scheme is named the minimum improvement targeting. Since it is an imitation of the minimum residual targeting scheme, it may also suffer the mis-convergence problem.

4. **Maximum residual targeting**. This scheme targets the Ritz pair with the largest residual norm. It tries to keep all wanted Ritz pairs converge at about the same speed. In other word, it minimizes the residual norm of the whole set of wanted eigenpairs. Similar to the minimum residual targeting, we need to compute the residual norms which is expensive in the Davidson method.

5. **Maximum improvement targeting**. This is similar to the Minimum Improvement scheme, the difference is that the ith Ritz vector corresponding to the largest $|y_{ji}|$ is chosen as the target. This is the practical form of the maximum residual targeting scheme which avoids computing of the residual norms.

In general, the Rayleigh-Ritz projection on a Krylov subspace basis gives accurate solutions to the extreme eigenvalues. In selecting the target from all possible Ritz pairs, half of the Ritz pairs are never considered. For example, if there are j Ritz values and the smallest eigenvalues are wanted, $j/2$ largest Ritz values are discarded automatically. The target, is sought among the $j/2$ smallest Ritz values only. Among these 5 targeting strategies, we have chosen to test 3 of them, the sloppy scheme, the minimum improvement scheme and the maximum improvement scheme. For most of our experiment so far, we have been using the sloppy targeting scheme. Here we will provide additional tests for the other two.

Table 4.9 shows the results using different targeting schemes on the three test problems. Only the Davidson method is used here. In the Si_2 case, the maximum improvement scheme found an eigenvalue missed by the other two schemes which might be the cause for it to take extra iterations. In general, the maximum improvement scheme uses less matrix-vector multiplication than the other schemes if small number of eigenpairs are sought. In the Si_6 case, the sloppy targeting scheme uses the least number of matrix-vector multiplications. This indicates that if the number of wanted eigenpairs is large, the sloppy targeting is the more effective approach.

In the unpreconditioned case, the residual vectors from both the Arnoldi method and the Lanczos method are parallel to each other, therefore targeting does not apply to them. Even if the residual vectors are not parallel to each other as in the case of variable preconditioning and thick restart, computing the residual vectors and performing targeting will turn them into the Davidson method. For this reason, targeting is only applicable to the Davidson method.

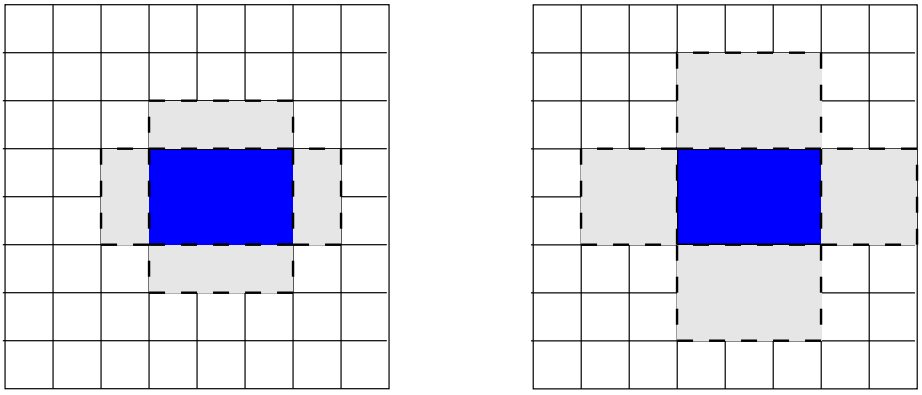

Figure 4.1: Boundary of finite difference domains.

4.6 Miscellaneous Issues

This section describes a number of issues that can be described in a few paragraphs. Some of these might be very important to our application. However we feel that they are well understood and it is fairly safe to take the solutions as is. Our implementation of the Davidson method is strongly influence by [137].

Matrix-vector multiplication To move the eigenvalue solvers on to a distributed environment, one of major tasks is to build a parallel matrix-vector multiplication. The matrix-vector multiplication can be divided into three components, the Laplacian operator, the local potential, and the non-local potential. The Laplacian operator is discretized on a uniform grid with high-order finite difference, see section 1.3 for details. Because of this, a significant portion of the grid points are on the boundary of two or more neighboring regions. Figure 4.1 illustrates this point on a small 2-D grid. For example, using a 12th order finite difference scheme on a 3-D, $50 \times 50 \times 50$ grid distributed among 8 processors, if the grid is divided into 8 $25 \times 25 \times 25$ cubes, the number of boundary points for each domain is 11,250 which is about 72% of the number of grid points in the domain. If a one-way dissection scheme is used, the grid point in the boundary of each domain is 30,000. On average, each domain only have 15,625 grid points. In this case, every grid point in each domain is a neighbor to the two neighboring domains. In order to perform matrix-vector multiplications, each domain needs to collect information about its neighboring grid points [74, 108]. The high-order finite difference scheme used generates a heavy communication burden for matrix-vector multiplications.

The local potential is translated into a diagonal matrix which is easy to multiply with a vector. The non-local potential consists of a series of rank-one updates. Each rank-one update only involves a group of nearby grid points. Thus it is advantageous to keep grid points near each other in physical space on the same processor.

Currently, because of the cut-off used to eliminate the region far away from all atoms, the actual shape of the domain is more complex than a simple cube. Because of this, a one-way dissection is used.

Block size One of the main reasons for using a block version of the Davidson method is to find eigenvalues with multiplicity. In most of the matrices we encounter, a small fraction of the eigenvalues have multiplicity of 2. In this case a block Davidson method may be more effective in finding these pairs. Table 4.10 shows the 15 eigenvalues found by the Davidson method with three different block sizes. The eigenvalues found by the one with large block size are smaller. This is one advantage of the block versions. Table 4.11 shows the number of matrix-vector multiplications and time spent by the Davidson method with different block sizes. As the block size increases, the number of matrix-vector multiplication also increases. With block size equals to 3, the Davidson method only found 54 eigenvalues in 6000 matrix-vector multiplications. Note that in

index	block size 1	block size 2	block size 3
1	-1.2750880627204270	-1.2750880627204220	-1.2750880627203820
2	-0.8992141228203836	-0.8992141228204291	-0.8992141228203198
3	-0.6568632812722515	-0.6568632812722639	-0.6568632812722880
4	-0.6568632812720828	-0.6568632812721114	-0.6568632812721611
5	-0.6041182133481225	-0.6041182133481178	-0.6041182133480028
6	-0.4435022950906883	-0.4435022950906463	-0.4435022950906686
7	-0.4435022950905120	-0.4435022950905874	-0.4435022950905792
8	-0.2028945606052125	-0.2028945606051076	-0.2028945606051557
9	-0.0507442640005357	-0.0507442640004036	-0.0507442640004178
10	0.0198086197809004	0.0198086197806211	0.0198086197809959
11	0.0230854125436574	0.0230854125442987	0.0230854125437922
12	0.0752754990856610	0.0752754990857491	0.0752754990855957
13	0.2245746976397505	0.0752754990859121	0.0752754990859352
14	0.4978875801877848	0.2011440266899505	0.2011440266913496
15	0.7908817209903563	0.2245746976397852	0.2170778546873740

Table 4.10: Eigenvalues of Si_2 found by the Davidson method.

	block size 1		block size 2		block size 3	
	MATVEC	time(sec)	MATVEC	time(sec)	MATVEC	time(sec)
Si_2	334	24.6	435	31.5	1291	88.9
Si_4	1177	240.3	1444	316.1	2692	585.2
Si_6	3856	1802.1	5180	2520.5	-54	2728.3

Table 4.11: Eigenvalues of Si_2 found by the Davidson method.

both the Si_4 and the Si_6 cases, the eigenvalues found with different block sizes are the same.

Table 4.10 may cause alarm about the quality of the eigenvalues found. In general, it is always possible that some desired eigenvalues are not found by a projection based eigenvalue method. If the 100 smallest eigenvalues are desired, usually, the Davidson method can find the smallest 80–90 without a problem. In table 4.10, 12 of the 15 smallest eigenvalues are the same for all three cases. This suggest that if 100 eigenvalues are desired, we should ask the Davidson method for 120. Also note that in this particular set of test matrices, the negative eigenvalues are the ones of interest. The Davidson method with block size 1 does find all negative eigenvalues correctly.

Workspace We have decided to save both V and $W(=AV)$ in memory. The basis vectors V are needed in many operations, unless we have to store a large number of them it is the easiest to just keep them around. There are alternatives to saving W. The array W is used to compute the residual vectors, $r = Wy - \lambda Vy$, and used to compute the projection, $H = V^T W$. The projection matrix H can be built progressively, therefore, only one column of W is needed at a time for this purpose. However all the columns are required to compute r. We could perform one matrix vector multiplication use the following formula instead $r = Ax - \lambda x$. On average there are less than 40 nonzero elements per row in the matrices. In terms of floating-point operations, performing one matrix-vector multiplication is equivalent to combining about 40 vectors to form Ax. But performing the matrix-vector multiplication has two disadvantages compared with performing linear combinations. First, the matrix-vector multiplication involves a sparse matrix. Because its irregular memory access pattern, it is usually slower than performing linear combination. Second, on parallel computers, the matrix-vector multiplication demands data communication between processors while the linear combination doesn't require any communication. In all of our experiments we always keep the basis size less than 40, therefore it is more efficient to save W rather than performing matrix-vector multiplications to compute residuals.

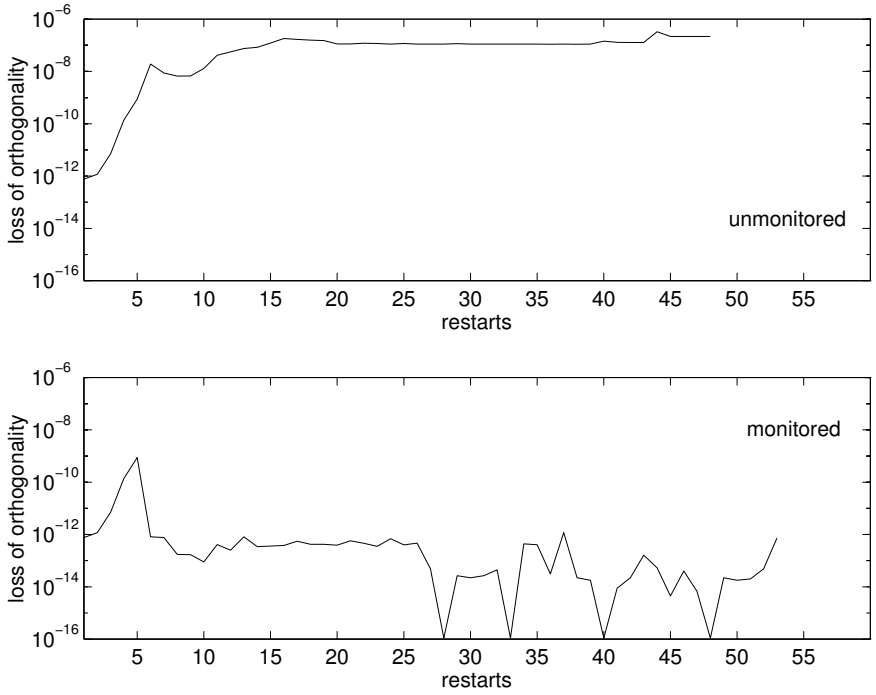

Figure 4.2: Loss of orthogonality in Arnoldi bases.

index	without monitoring	with monitoring
1	-1.2750880627204260	-1.2750880627204260
2	-0.8992141228204374	-0.8992141228204383
3	-0.6568632812723210	-0.6568632812723436
4	-0.6568632812722531	-0.6568632812723463
5	-0.6041182133481187	-0.6041182133481187
6	-0.4435022950906814	-0.4435022950906887
7	-0.4435022950906830	-0.4435022950906908
8	-0.2028945606049544	-0.2028945606051070
9	-0.0507442640005358	-0.0507442640005366
10	0.0198086197809159	0.0198086197811722
11	0.0230854125435991	0.0230854125438236
12	0.0752754990856684	0.0752754990856461
13	0.0752754990859336	0.0752754990858102
14	0.2245746976397545	0.2011440266897905
15	0.4978875801876783	0.2245746976397520

Table 4.12: Eigenvalues of Si_2 found by the Arnoldi method.

Normality of Ritz vectors In order to save matrix-vector multiplications, at restart, the Ritz vectors in V are assumed to be orthonormal, and the corresponding vectors in W are assumed to satisfy relation $W = AV$. After a number of restarts, the Ritz vectors may gradually lose orthogonality because the accumulation of round-off error. This is especially a problem with the Arnoldi method. Thus we use the Arnoldi method as an example. Here is a list of Frobenius norms of $V^TV - I$ from the first 10 bases built by the Arnoldi method when solving the Si_2 test problem,

$$10^{-12}, 10^{-12}, 10^{-11}, 10^{-10}, 10^{-9}, 10^{-8}, 10^{-8}, 10^{-8}, 10^{-8}, 10^{-7}.$$

The above list shows that the orthogonality of the basis deteriorates quickly from one restart to the next restart. Theoretically, the unpreconditioned Arnoldi method is equivalent to the unpreconditioned Davidson method. However, in reality, the Arnoldi method often takes more iterations to compute the same eigenpair. This loss of orthogonality problem the root cause of the difference. This loss of orthogonality problem also happens to the Davidson method especially when a good preconditioner is used. Thus, there is a need to moderate the loss of orthogonality in the basis. Computing $(V^TV - I)$ is fairly expensive, so we choose to monitor the norm of the first Ritz vector computed at step 4 in algorithm 4.1. If the norm of this Ritz vector deviates from 1 significantly, we will discard the content of W and re-orthogonalize the vectors in V before continuing. The first Ritz vector is computed by $x_1 = Vy_1$. Let V^* and y_1^* be the quantity V and y_1 in exact arithmetic, the error in the computed Ritz vector x_1 is $\delta x_1 = V^*(y_1 - y_1^*) + (V - V^*)y_1^* + (V - V^*)(y_1 - y_1^*)$. If the errors in V and y_1 are small, the last term can be neglected. To estimate the difference between the norm of x_1 and 1, we need to evaluate the following expression $x_1^T \delta x_1$. The norm of the error is $\|\delta x_1\| \leq \|y_1 - y_1^*\| + \|V - V^*\|$. Since the vector y_1 is an eigenvector of H computed by a QR based eigenvalue routine from LAPACK, it should be very close to be a unit vector. Thus, the deviation in norm of x_1 is mostly contributed from V. If each entry of V is allowed to have ϵ difference from its exact value, $\|V - V^*\| \leq \sqrt{mn}\epsilon$ where n is the number of rows in V and m is the number of columns. The value of ϵ can be the unit round-off error. If the first Ritz vector norm deviates more than $2\epsilon\sqrt{mn}$ from one, we consider that the basis V has lost orthogonality, the vectors are orthonormalized again before building the next basis. The following are the orthogonality measure $\|V^TV - I\|_F$ from the first ten bases built by the Arnoldi method with the above monitoring scheme,

$$10^{-12}, 10^{-12}, 10^{-11}, 10^{-10}, 10^{-9}, 10^{-12}, 10^{-12}, 10^{-13}, 10^{-13}, 10^{-13}.$$

Figure 4.2 show the orthogonality of the Arnoldi bases with and without this monitoring scheme. Without monitoring, the Arnoldi method build 48 bases, with monitoring, the Arnoldi method build 53 bases. Obviously, extra work is done with monitoring, the payoff is the extra eigenvalue found, see table 4.12. The eigenvalue 0.2011 was missed by the Arnoldi method without the monitoring scheme.

Here we demonstrated the importance of maintaining orthogonality of the basis by one example and showed one way of maintain the orthonormality of the basis. We should point out that the interplay between the orthonormality of the basis and the quality of the eigenpairs is a fairly complex issue. Loss of normality of the Ritz vectors is only one results from loss of orthonormality in basis. Reducing the tolerance on deviation of Ritz vector norm does not always increase the quality of the basis. Re-orthogonalizing the basis before every restart does not reduce the number of restarts required to compute the same solution. Maintaining good orthogonality of the basis is only one of the many factors that can affect the quality of eigenpairs. This example happens to involve a missing eigenvalue. In general, we don't believe the missing eigenvalue problem can not be addressed solely by orthogonality alone.

4.7 Summary

This chapter is devoted to some of the implementation issues related to the development of an effective eigenvalue code for large sparse eigenvalue problems. We discussed techniques adopted to computed a large number of eigenpairs without requiring proportionally large amount of workspace, variations of thick-restarting schemes and targeting schemes. We have also shown the reverse communication protocol used in the program to remove the dependencies on the sparse matrix storage format and to ensure an easy transition to a parallel environment. Some key components of the Davidson method are re-orthogonalized to enhanced the program's performance on parallel machines.

To show that it is in fact easy to adopt the program to the parallel environment, we will show some preliminary results of using the eigenvalue code described above on the eigenvalues problem generated from a slightly larger simulation cases. To limit the waiting time during the experiment, we only tested the eigenvalue code on the $Si_{47}H_{60}$ case. The matrix size is about 30,000 and 162 eigenpairs are wanted from the lower end of the spectrum. The test is done on a small IBM SP2 cluster with high performance switches. The execution time to find all required eigenpairs during the first self-consistent iteration is shown in table 4.13. The results in this table is far from optimal and we are actively seek to identify the strength and weakness of the code on parallel environment.

# procs	time	speedup
1	971	-
2	722	1.3
4	588	1.7

Table 4.13: The execution time of the eigenvalue code on a small SP2 cluster.

Chapter 5

Polynomials in Eigenvalue Methods

5.1 Introduction

Use of optimal polynomials has been an integral part of eigenvalue methods [15, 110, 112, 116, 120, 121]. Let A denote the matrix of interest, $P(A)$ denote a polynomial of A. In these eigenvalue methods, the polynomial $P(A)$ is not explicitly required. The operation of multiplying $P(A)$ by a vector can be broken into a series of matrix-vector multiplications with A. Since the matrix-vector multiplication with A is an essential part of any projection eigenvalue method, only a minimal amount of additional work is required to add a polynomial preconditioner or to construct a hybrid method that alternates between a Krylov method and a polynomial method.

Polynomials can be used to directly improve the quality of an approximate eigenvector. Some of the earlier eigenvalue methods rely on this property exclusively for computing eigenvector approximations, for example, RITZIT [104, 105]. Recently, it has been shown that a more effective way of using polynomials is to construct a hybrid method that alternates between a polynomial method and a projection method [112, 116, 121]. Hybrid methods of this type are attractive on new parallel computing environments because they require little additional work beyond the basic projection eigenvalue method. In addition, the Chebyshev polynomial has been shown to successfully complement the Arnoldi method and other Krylov methods in solving linear systems and eigen-systems [17, 116, 121, 123, 132]. In the hybrid method mentioned above, this step of applying the polynomial to improve the quality of eigenvectors is also referred to as purification, or polynomial purification.

Preconditioning techniques which seek to approximately solve $(A - \lambda I)z = r$ are known to be error-prone, see page 30 and [36]. For this reason, polynomial preconditioners used in this chapter are not linear system solvers. Instead, they are designed to improve eigenvector approximations like in the polynomial purification. This avoids some of the difficulties of solving the nearly singular preconditioning equation. The second reason for not directly using polynomials to solve the preconditioning equation is that there are many Krylov methods that are easier to use and are just as effective as polynomial methods with optimal parameters. These optimal parameters for polynomial methods are often computed from the spectrum of the matrix which is usually unknown. Since Krylov methods like the Conjugate Gradient (CG) method, do not require detailed knowledge about the spectrum of the matrix, they are easier to use than polynomial methods. In practice, CG is also observed to be more effective than the Chebyshev method on symmetric linear systems. However, during computation of the eigenvalues, it is possible that the parameters for the polynomials can be better estimated than what is possible during the solution of a linear system. Through the use of better parameters and avoiding some pitfalls of the Davidson preconditioning, polynomial preconditioning could be more effective for eigenvalue problems than for linear systems.

In Chapter 2, we adopted a locking mechanism to reduce the active basis size in eigenvalue routines. When computing a large number of eigenpairs using these algorithms, a significant amount of execution time is spent on orthogonalizing the current basis vectors against the converged eigenvectors. One motivation for considering polynomials is to reduce this orthogonalization time. In terms of arithmetic operations, if an average row of a sparse matrix has 40 nonzero elements, performing a matrix-vector multiplication is

equivalent to orthogonalize against 20 eigenvectors. In this case, orthogonalizing against 1000 eigenvectors is equivalent to 50 matrix-vector multiplications, i.e., a polynomial of degree 50 can be constructed using the same amount of arithmetic operations. If (λ, x) is an eigenpair of A, the corresponding eigenpair of $P(A)$ is $(\mu = P(\lambda), x)$. With a polynomial of degree 50, an interior eigenvalue of A may be transformed into an extreme eigenvalue of $P(A)$. In this case, the Davidson method applied on $P(A)$ will not be required to enforce orthogonality against the converged eigenvectors of A. This use of polynomials will be called polynomial spectrum transformation. It is a simpler version of spectrum transformation techniques, since commonly used ones, for example, shift-invert Lanczos techniques [53], and rational Krylov methods [103], all require inversion of a sequence of different matrices during the solution of one eigenpair. The polynomial spectrum transformation technique can not only be used to transform interior eigenvalues into extreme ones, but also used to increase the separation between the wanted eigenvalue and the unwanted ones.

In short, this chapter will study three ways of using polynomials in eigenvalue methods, purification, preconditioning, and spectrum transformation. The rest of this chapter is arranged as follows. In the next section, we will describe some characteristics of the polynomials to be used. Section 5.3 gives some examples of how the Davidson method behave for interior eigenvalue problems since we have not done so before. Section 5.4 describes how we determine the coefficients of the polynomials. Sections 5.5, 5.6 and 5.7 will be devoted each to one of the three techniques of using polynomials. Section 5.8 describes an example of how to use polynomials to increase efficiency when computing a large number of eigenpairs. A brief summary is provided at the end.

5.2 Characteristics of the polynomials

Polynomials we will use in this study are: the Chebyshev polynomials [51, 116], the Chebyshev polynomials of the second kind [101], the least-squares polynomials [111] and the Kernel polynomials [1, 23, 33, 141]. This section will not give detailed algorithms for these polynomials since they are widely available. The main purpose of this section is to familiarize readers with some general characteristics of these polynomials and review the features that are relevant to their performance in solving eigenvalue problems.

The polynomials mentioned above can be divided into two categories according to their usage. They are either used to compute extreme eigenvalues or interior eigenvalues. We use the following polynomials for extreme eigenvalue problems: the Chebyshev polynomials, the Chebyshev polynomials of the second kind, and the least-squares polynomials on one interval. The following polynomials are used for interior eigenvalue problems: the least-squares polynomial on two intervals and the Kernel polynomial. Note that the Kernel polynomials may also be used for extreme eigenvalues. However, we will mainly use it for interior eigenvalue problems in this study.

The Chebyshev polynomial can be computed through a simple three-term recurrence. Let $T_k(\zeta)$ denote the kth degree polynomial of ζ, the recurrence is

$$T_{k+1}(\zeta) = 2\zeta T_k(\zeta) - T_{k-1}(\zeta),$$

where $T_0 = 1$ and $T_1 = \zeta$. The Chebyshev polynomial defined this way is an orthogonal polynomial in the interval $[-1, 1]$ which is also known as the Chebyshev interval. Within this interval, the maximum of the polynomial is 1. Outside of this interval, the absolute value of the polynomial grows rapidly. Let (λ_i^*, x_i^*), $i = 1, \ldots, n$, denote the exact eigenpairs of the matrix A, and the initial guess of the eigenvector be the following

$$x = \sum \alpha_i x_i^*. \tag{5.1}$$

For simplicity, we assume the eigenvalues are in ascending order. The result of applying a Chebyshev polynomial on x can be expressed as follows,

$$T_k(A)x = \sum \alpha_i T_k(\lambda_i^*) x_i^*. \tag{5.2}$$

Compared with the initial value x, we see that $T_k(A)x$ has larger components corresponding to λ_i^* outside of the Chebyshev interval. In other word, the components corresponding to λ_i^* inside the Chebyshev interval

is suppressed. If the wanted eigenvalue is the only one outside of $[-1, 1]$, with sufficiently large k, $T_k(A)x$ can be made arbitrarily close to the wanted eigenvector.

Normally, the unwanted part of spectrum does not conveniently fall into $[-1, 1]$. In which case, it is necessary to shift and scale the polynomial, i.e., change the Chebyshev interval to cover the unwanted eigenvalues. For a complete algorithm for computing this shifted and scaled Chebyshev polynomial, see, for example, reference [116]. From now on, unless stated otherwise, all polynomials used are shifted and scaled. If the approximate eigenvalue is λ, the polynomials are scaled so that $P(\lambda)$ is 1. For symmetric eigenvalue problems, the Chebyshev polynomial is defined on a Chebyshev interval that contains the unwanted part of the spectrum. If the wanted eigenvalue is λ, the center of the interval is at c, the half width of the interval is d, i.e., the Chebyshev interval is $[c - d, c + d]$, then the following is true,

$$\frac{T_k(\lambda)}{T_k(c-d)} \approx \left(\frac{|\lambda - c| + \sqrt{(\lambda - c)^2 - d^2}}{d} \right)^k. \tag{5.3}$$

This is a well known result about the Chebyshev polynomial which will be used to select the degree of the polynomial and the boundaries of the interval.

Assuming the first eigenpair is to be computed, in order to reduce error of the approximate solution, the Chebyshev interval should contain eigenvalues $\lambda_2^*, \ldots, \lambda_n$. To measure the effectiveness of polynomial, the following definition of the damping factor γ is commonly used.

$$\gamma = \max_{i \neq 1} \frac{T_k(\lambda_i^*)}{T_k(\lambda_1^*)}.$$

In practical use, the damping factor of T_i is often approximated as

$$\gamma \approx \frac{T_k(c-d)}{T_k(\lambda)}.$$

Since we also scale $T_k(\lambda)$ to 1, $\gamma \approx T_k(c - d)$.

Figure 5.1 shows two instances of the Chebyshev polynomial. The examples are designed with the following scenario in mind: the eigenvalues of the matrix are in the range of [0, 1], and the smallest eigenvalue 0 is to be computed. The polynomials are scaled so that $T_k(0) = 1$. The figure shows that a 5-degree Chebyshev polynomial can reduce the contribution of eigenvalues in the range of [0.1, 1] by about one-eighth. In order to reduce the contribution from all eigenvalues in the range of [0.01, 1] by the same ratio, a 17-degree Chebyshev polynomial is needed. This clearly demonstrates the dependency between the damping factor and the separation between wanted and unwanted eigenvalues.

An important characteristic of the Chebyshev polynomial is that among all polynomials of the same degree it has the fastest growth rate outside of the same interval [101, section 2.7.1]. Many aspects regarding the effective use of the Chebyshev polynomial have been discussed. In this study, we emphasis how to use Chebyshev polynomial to transform part of the spectrum to help the Davidson method to reach convergence. In this context, we do not rely on the polynomial method to handle the whole spectrum, the results and observations from previous studies may not apply. However, we are strongly influenced by previous researches in adaptive Chebyshev polynomial for linear systems and eigenvalue problems [6, 15, 112, 113, 116, 120, 121]. Another property of the Chebyshev polynomial of interest is that it has the largest derivative among all the polynomials that has the same maximum value in the interval [101, section 2.7.4]. If the extreme eigenvalues are clustered, this property indicates that using the Chebyshev polynomial to transform the spectrum is most effective in increasing the separations.

The Chebyshev polynomial we referred to in the above discussion is the Chebyshev polynomial of the first kind [101]. The above reference to the derivative leads us to consider the Chebyshev polynomial of the second kind which is the derivative of the Chebyshev polynomial of the first kind. Let $U_k(\zeta)$ denote kth order Chebyshev polynomial of the second kind. The first two U_ks are: $U_0 = 0$, $U_1 = 2\zeta$. The remaining ones follow the same recurrence relation as the Chebyshev polynomial the first kind [101, Ex 1.5.19]. Figure 5.2 shows two examples of the two kinds of Chebyshev polynomials. Note that the Chebyshev polynomials shown are the same as in Figure 5.1. In figure 5.2, the Chebyshev polynomial of the first kind is indicated

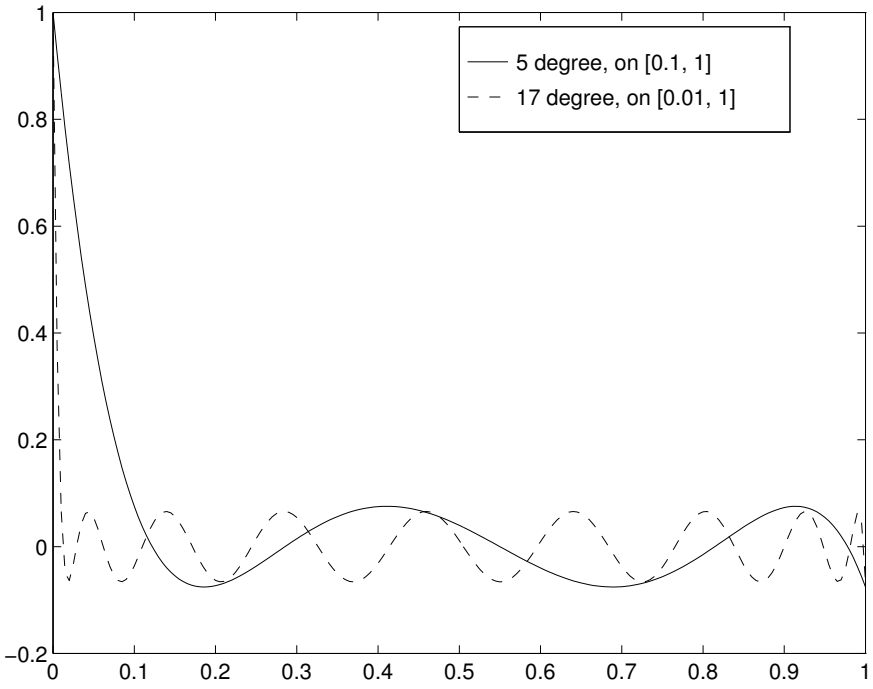

Figure 5.1: Two Chebyshev polynomials.

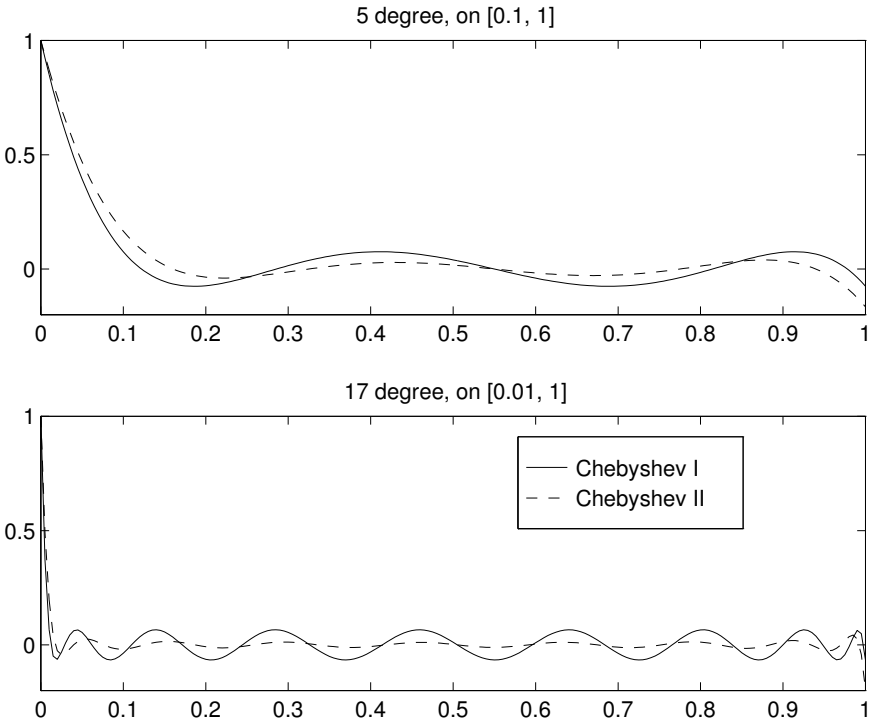

Figure 5.2: Comparison of Chebyshev polynomials of the first kind and the second kind.

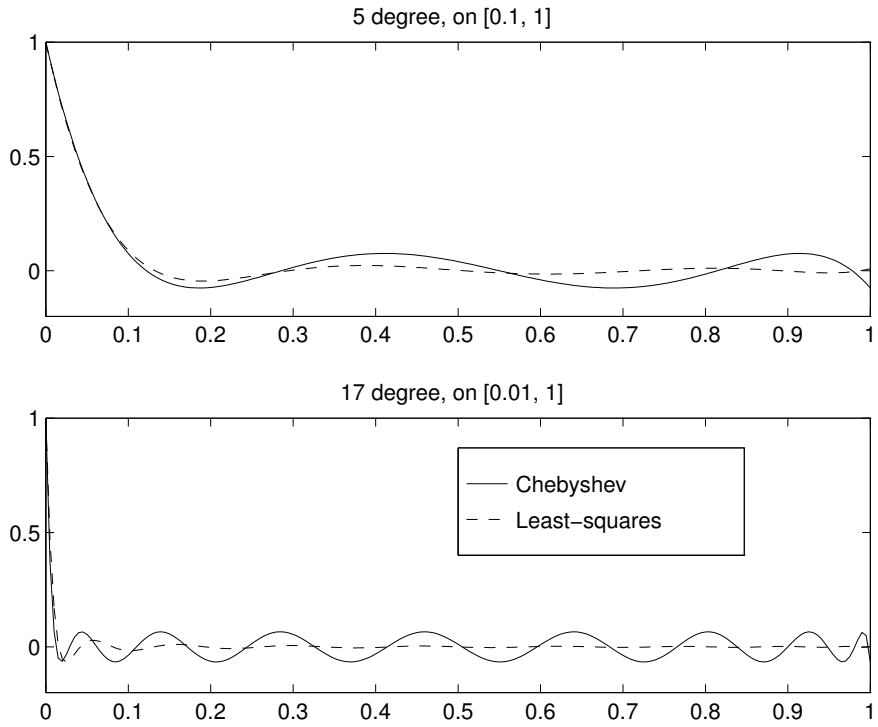

Figure 5.3: Comparison of the Chebyshev polynomial and the least-squares polynomial.

by 'Chebyshev I' and the Chebyshev polynomial of the second kind is indicated by 'Chebyshev II'. We note that the two Chebyshev polynomials are close to each other near 0. In the range $[0.1, 0.9]$, the value of the Chebyshev polynomial of the second kind is smaller than the Chebyshev polynomial of the first kind. This indicates that the Chebyshev polynomial of the second kind could damp interior eigenvalue components better than the Chebyshev polynomial of the first kind. Near 1, the Chebyshev polynomial of the second kind is again larger in magnitude because the Chebyshev polynomial of the first kind changes rapidly near the ends of the Chebyshev interval. Overall, if the initial guess of an extreme eigenvector does not contain any contribution from the opposite end of the spectrum, then using the Chebyshev polynomial of the second kind could be more effective than the commonly used Chebyshev polynomial of the first kind.

The least-squares polynomial described in [111] was originally developed for solving indefinite linear systems. It was designed to damp eigen-components in two disjoint intervals of the spectrum. Further study of using the least-squares polynomial as preconditioner for linear system can be found in [6]. The same least-squares principle can be used to develop a polynomial that only damps eigen-components in one interval like the Chebyshev polynomial. This is what is shown in Figure 5.3. The recursive formula for computing the coefficients of the polynomial is given in [111] and [116]. Without referring to the details of the algorithm, we note that the least-squares polynomial is computed through a three-term recurrence which is similar to the Chebyshev polynomial. This least-squares polynomials used here are built from the Chebyshev basis, other orthogonal basis could also be used.

Figure 5.3 shows two examples of the least-squares polynomials on one interval. The Chebyshev polynomials shown in this figure are again the same as in figure 5.1 and the least-squares polynomial in each plot is of the same order and on the same interval as the corresponding Chebyshev polynomial. One important observation here is that the least-squares polynomial is much closer to zero far away from 0 compared to the Chebyshev polynomial. This is the main motivation for us to consider the least-squares polynomial. Similar to the Chebyshev polynomial, the interval on which the least-squares polynomial is built determines its property. Selection of this interval is the main issue to be addressed.

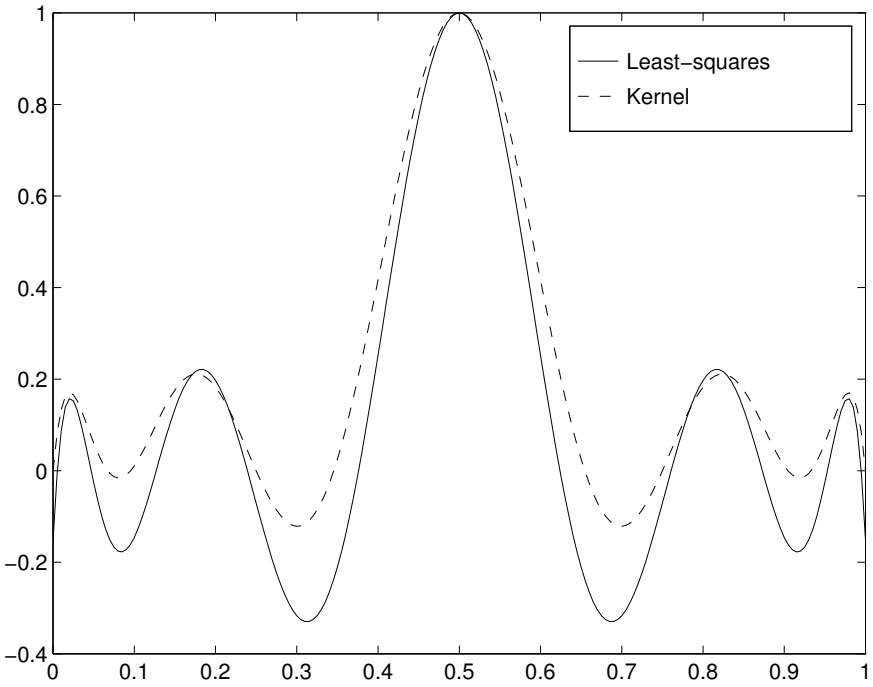

Figure 5.4: The two polynomials for interior eigenvalue problems.

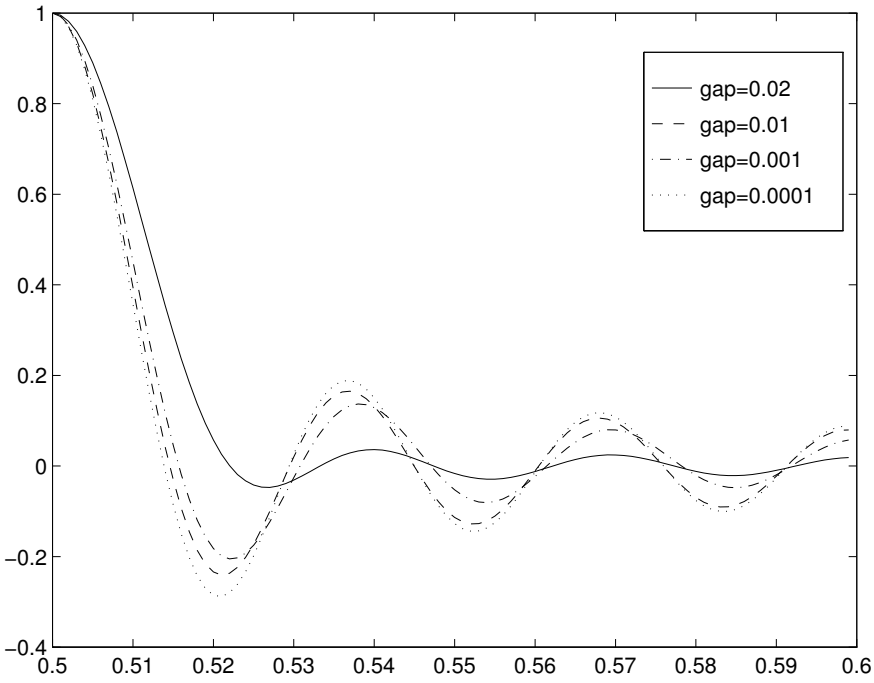

Figure 5.5: The 2-interval least-squares polynomial with inner boundaries.

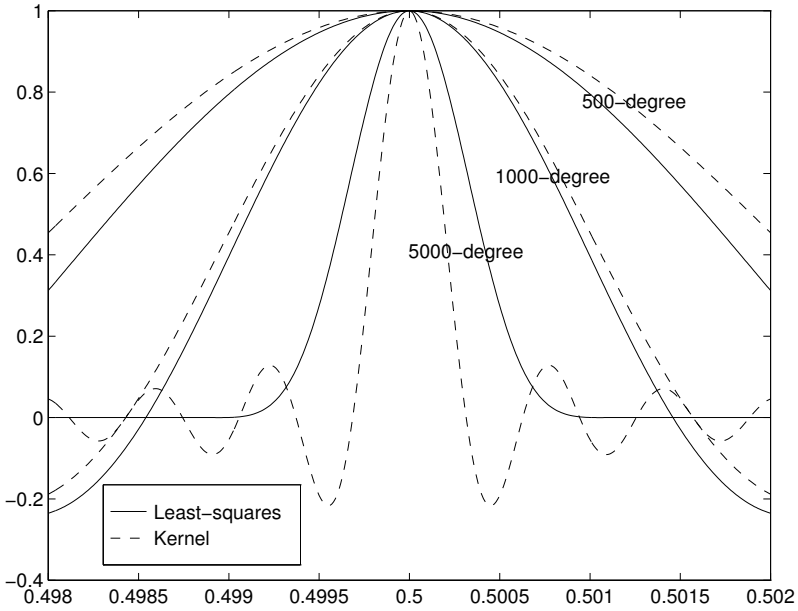

Figure 5.6: The Kernel polynomial and the 2-interval least-squares polynomial at very high degree.

Figure 5.4 shows an example of the two polynomials we plan to use for interior eigenvalue problems, namely, the least-squares polynomial on two intervals and the Kernel polynomial. In this case, the two intervals are $[0, 0.45]$ and $[0.55, 1]$. The Kernel polynomial centers at 0.5 and it is constructed from a sequence of Chebyshev polynomials on the interval of $[0, 1]$. Both polynomials are scaled to be one at the center. Kernel polynomials can be constructed using different basis polynomials, we have chosen to only use the Chebyshev basis. Kernel polynomials has been used for interior eigenvalue problems in the literature. For example, in [151], the authors described how to use Kernel polynomials in simulation of material properties. The magnitude of a Kernel polynomial reduces as it goes away from the specified center. In this respect, it can be regarded as a polynomial approximation of the δ-function [33]. Using approximate δ-function for interior eigenvalue problem has been studied also, for example, [92]. Using Kernel polynomial for linear systems can be traced back to 1955 [138]. More recent references include [45, 46].

Figure 5.5 uses a few examples to show the dependencies of the least-squares polynomial on the intervals chosen. The polynomials shown in this figure are of degree 100. The wanted eigenvalue is still assumed to be 0.5. The gap refers to the distance between 0.5 and the end of the intervals. For example, if the gap is 0.01, then the two intervals are $[0, 0.49]$, and $[0.51, 1]$. To effectively use the least-squares polynomial and the Kernel polynomial, the key is again how to select the intervals on which to build these polynomials. This is especially important for the least-squares polynomial since there are two intervals to be determined when seeking interior eigenvalues. In this chapter, we study different schemes for selecting the intervals for the polynomials, and see how to effectively combine them with the Davidson method to solve eigenvalue problems.

5.3 Davidson method for interior eigenvalues

Computing interior eigenvalues is generally regarded as a more difficult problem then computing extreme eigenvalues. Theoretically, the Lanczos method converges faster toward extreme eigenvalues than toward interior eigenvalues [116]. The same is true for the Arnoldi method and the Davidson method when applied to a symmetric problem without preconditioning. The following is an example shows that the interior eigenvalue problem is more difficult for polynomial methods as well. Figure 5.6 shows the Kernel polynomial and the

2-interval least-squares polynomial at three different orders, namely, 500, 1000, and 5000. The two intervals for the least-squares polynomial is $[0, 0.499]$ and $[0.501, 1]$. The Kernel polynomial is build on the interval $[0, 1]$. The centers for both polynomials are 0.5. At 5000 degree, the two-interval least-squares polynomial is about 1.4×10^{-4} at 0.499, see Figure 5.6. If the extreme eigenvalue has 0.001 separation from the rest, i.e., the extreme eigenvalue is 0, the rest of the spectrum is in $[0.001, 1]$, a 150 degree Chebyshev polynomial can achieve the damping factor of 1.4×10^{-4}, and a 172 degree least-squares polynomial on the same interval can also achieve the same damping factor. As expected, with the same separation, a polynomial of much higher degree is required to achieve the same damping factor for interior eigenvalues than for extreme eigenvalues.

The main task of this section is to identify a basic algorithm for interior eigenvalue problems for later experiments. Based on our past experience with the Davidson method, it is a good candidate for computing a few interior eigenvalues.

As we have mentioned in previous chapters, one way to compute a few eigenvalues around a given number $\hat{\lambda}$ is to slightly modify the Davidson method, algorithm 4.1. The modification can be limited to the the Rayleigh-Ritz procedure in algorithm 4.1. Instead of ordering the eigenvalues of H_m in ascending order, we now order them according the distance from $\hat{\lambda}$ and place the ones closer to $\hat{\lambda}$ at the beginning of the array. Aside from the Rayleigh-Ritz procedure, we can also use harmonic Ritz projection scheme to compute the eigenpairs [78]. Since the harmonic Ritz projection technique is designed to compute interior eigenvalue, the harmonic Davidson method for interior eigenvalues use it to find the eigenvalues during the expansion of the basis and during restart, i.e., the harmonic Ritz method are used in both step 2.d and step 3 of algorithm 4.1. This is different from the harmonic Davidson method for extreme eigenvalues where the harmonic Ritz projection is only used in step 2.d of algorithm 4.1. Because the harmonic Ritz vectors are not orthonormal in Euclidean norm, they are orthonormalized before restart to maintain the orthonormality of the basis. One alternative would be to choose to work with non-orthonormal basis which would require more modification to the code we have for computing the extreme eigenvalues. Thus we will not use this alternative form.

To test the performance of the Davidson method and the harmonic Davidson method for interior eigenvalue problems, we conducted an experiment on the non-diagonal symmetric matrices in the Harwell-Boeing collection, see tables 2.2 and 2.3 and reference [37]. In this experiment, we compute the two eigenvalues that are closest to the center of the Gershgorin interval $[a_G, b_G]$ where [51, 139]

$$a_G = \min_i(|a_{ii}| - \sum_{j \neq i} |a_{ij}|), \qquad b_G = \max_i(|a_{ii}| + \sum_{j \neq i} |a_{ij}|).$$

The number of matrix-vector multiplications used to compute these two eigenvalues are shown in tables 5.1 and 5.2. The maximum basis size used in the Davidson method and the harmonic Davidson method is 20. The tolerance on the residual norm is set to be $10^{-12}\|A\|_F$. No preconditioning is used. The basis is built from one initial vector $[1, 1, \ldots, 1]^T$. In chapter 2, we have shown that the Davidson method without preconditioning is equivalent to the Arnoldi method in theory. However, in practice, the Davidson method is more stable than the Arnoldi method and it is more effective with preconditioning as well.

In tables 5.1 and 5.2, the first column is the matrix name, the second column is the number of matrix-vector multiplications used by the Davidson method for interior eigenvalue problems, the third column shows the number of matrix-vector multiplications used by the harmonic Davidson method, and the last two columns are the results from two polynomial methods. The polynomial methods simply apply either the 100-degree least-squares polynomial or the 100-degree Kernel polynomial on a set of initial guesses, then apply Rayleigh-Ritz projection on the resulting vectors. This process is repeated until the residual norms are less than $10^{-12}\|A\|_F$. The interval used for the Kernel polynomial is the Gershgorin interval. Denote the two intervals of the least-squares polynomial as $[a_1, b_1]$ and $[a_2, b_2]$. The values of a_1 and b_2 are the left end and the right end of the Gershgorin interval. Order the Ritz values according to the distance from the center of the Gershgorin interval, then a_2 and b_1 are computed as follows,

$$b_1 = c - |c - \lambda_3|, \qquad a_2 = c + |c - \lambda_3|,$$

where $c = (a_1 + b_2)/2$ in this test. The polynomials are applied on a block of 20 vectors. The initial guesses are the first 19 columns of the identity matrix plus the vector $[1, 1, \ldots, 1]^T$.

	Davidson	harmonic Davidson	least-squares	Kernel
494 BUS	28	>100000	10820	62620
662 BUS	37	972	7020	32320
685 BUS	19	>100000	4940	54540
1138 BUS	347	905	>100000	>100000
BCSSTM07	476	2082	24240	24240
BCSSTM10	32724	>100000	66660	78780
BCSSTM12	238	328	15340	28280
BCSSTM13	47	>100000	2020	2020
BCSSTM27	9114	27487	44440	54540
GR 30 30	25365	80615	40400	40400
LUND A	385	2890	20200	18180
LUND B	357	5063	24240	22220
NOS1	197	2554	36360	38380
NOS2	2927	14246	34340	36360
NOS3	14368	>100000	>100000	>100000
NOS4	401	4582	24240	24240
NOS5	6272	19491	42420	30300
NOS6	1123	>100000	28680	>100000
NOS7	44	971	44760	>100000
PLAT1919	8141	54674	>100000	>100000
PLAT362	755	>100000	40400	34340
ZENIOS	>100000	>100000	2020	2020

Table 5.1: Number of matrix-vector multiplications used to compute two interior eigenvalues of Harwell-Boeing matrices (PART I).

Among the four methods tested in this experiment, the Davidson method is the most effective one. The two polynomial methods tested generally require considerably more matrix-vector multiplications compared with the Davidson method. The only exception to above observation is the matrix ZENIOS. There are a number of zero rows near the top of the matrix, which makes 0 an exact eigenvalue and e_i the exact eigenvector. Since the center of the Gershgorin interval is also 0 in this case, the initial guess for the two polynomial methods contains the exact solutions. Overall, on the problems tested, the Davidson method performs better than the harmonic Davidson method. There is only one exception found, the harmonic Davidson method used less matrix-vector multiplications to find two interior eigenvalues of BCSSTK14 than the Davidson method. In this case, the harmonic Davidson method not only used less matrix-vector multiplications, but also found two eigenvalues that are closer to the center of the Gershgorin interval.

Comparing between the two polynomial methods, the least-squares polynomial performs better than the Kernel polynomial because is used less matrix-vector multiplications to achieve convergence in a large number of the test cases. This can be partly explained by the fact that the intervals for the least-squares polynomials are dynamic selected. Because this variations in the boundaries of the intervals, the least-squares polynomial has peaks and valleys located at different locations which makes it more effective in damping out the unwanted eigenvector components, see figure 5.5.

Based on the results of this experiment, we will only use the Davidson method for the later tests on interior eigenvalue problems. The results of the polynomial methods for interior eigenvalues provides a reference point for later comparisons.

	Davidson	harmonic Davidson	least-squares	Kernel
BCSSTK01	35	3012	18180	18180
BCSSTK02	36	3480	14140	18180
BCSSTK03	22	30	3740	6060
BCSSTK04	138	1085	20200	22220
BCSSTK05	128	602	22220	22220
BCSSTK06	139	167	12120	30300
BCSSTK07	139	167	12120	30300
BCSSTK08	20	20	3100	60600
BCSSTK09	1191	10380	46460	60600
BCSSTK10	1735	3729	44440	38380
BCSSTK11	171	1852	17960	38380
BCSSTK12	171	1852	17960	38380
BCSSTK13	61	80	32320	>100000
BCSSTK14	23294	3207	34340	>100000
BCSSTK15	1660	7638	58580	>100000
BCSSTK16	43679	>100000	>100000	>100000
BCSSTK17	461	3141	66660	>100000
BCSSTK18	123	1361	4940	36360
BCSSTK19	189	2332	21840	>100000
BCSSTK20	21	7704	22340	>100000
BCSSTK21	60705	>100000	>100000	>100000
BCSSTK22	118	962	10100	44440
BCSSTK23	45	721	27780	>100000
BCSSTK24	86	433	46460	56560
BCSSTK25	17	142	27020	38380
BCSSTK26	71	5251	32320	>100000
BCSSTK27	1029	13427	36360	38380
BCSSTK28	31	54150	30300	>100000

Table 5.2: Number of matrix-vector multiplications used to compute two interior eigenvalues of Harwell-Boeing matrices (PART II).

5.4 Intervals of unwanted eigenvalues

As indicated before, in order to effectively use the polynomials, we need to determine their coefficients. To fully determine the three polynomials for extreme eigenvalue, we need to identify an interval covering unwanted eigenvalues. For the Kernel polynomial, we need to know the range of the entire spectrum. In the two-interval least-squares polynomial case, two intervals each from one side of the wanted eigenvalue need to be identified. In addition to these intervals, we need to also identify an appropriate degree for the polynomials.

When considering extreme eigenvalues, two boundaries of an interval covering all unwanted eigenvalues need to be determined. For convenience of discussion, let $[a, b]$ denote the interval and assume that the smallest eigenvalue and the corresponding eigenvector is to be computed. We will refer to a as the left end, and b as the right end. In this study, we consider three different ways of selecting a and two different ways of selecting b. Since these boundaries completely determine the properties of the three polynomials we have chosen for extreme eigenvalue problems, it is crucial to select the optimal values for both a and b.

We will use the damping factor of the Chebyshev polynomial to guide the selection of degree of the polynomials to use after the interval $[a, b]$ is selected. Note that the center c and the half-width of the interval d are defined as follows,

$$c = \frac{a+b}{2}, \qquad d = \frac{b-a}{2}.$$

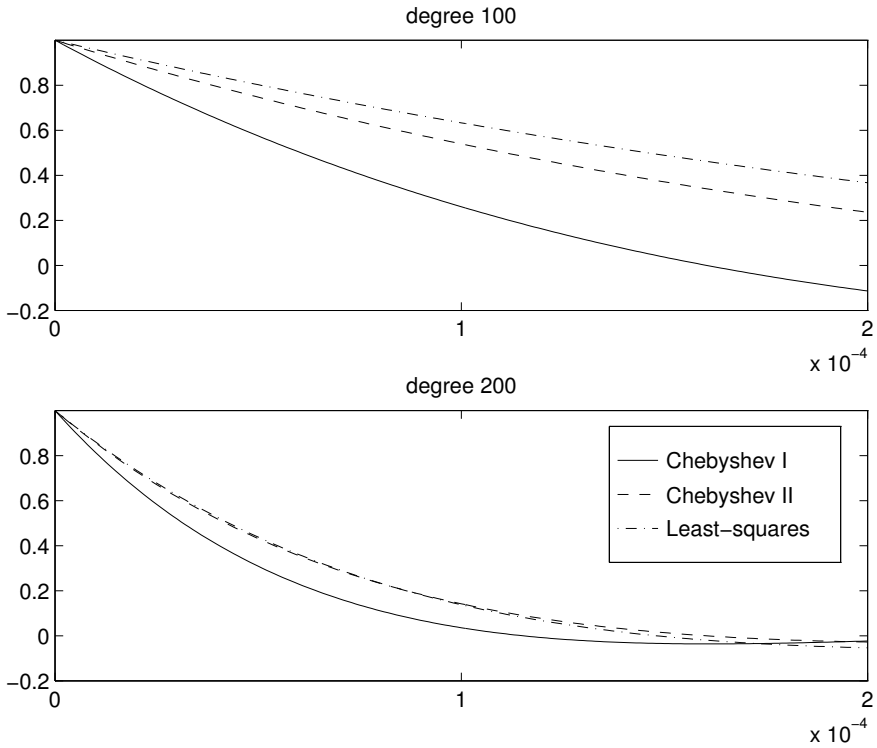

Figure 5.7: Left ends of the three polynomials for extreme eigenvalue problems.

	100 degree		200 degree	
ζ	0.0001	0.0002	0.0001	0.0002
Chebyshev I	0.2607	-0.1132	0.0359	-0.0229
Chebyshev II	0.5397	0.2359	0.1422	-0.0275
least-squares	0.6333	0.3673	0.1378	-0.0531

Table 5.3: Values of the three polynomials for extreme eigenvalue problems at selected points.

Using equation (5.3), the damping factor of a k-degree Chebyshev polynomial is given by the following equation,

$$\gamma \approx \left(\frac{d}{c - \lambda + \sqrt{(c-\lambda)^2 - d^2}} \right)^k = \left(\frac{b-a}{b + a - 2\lambda + 2\sqrt{(a-\lambda)(b-\lambda)}} \right)^k. \tag{5.4}$$

The degree of the polynomial is computed in such a way that the components associated with the unwanted eigenvalues are reduced by a factor of $1/2$, in other word, $T(\lambda_i^*) \leq 1/2$, $i = 2, \ldots, n$, see equation (5.2). By definition, this means the damping factor γ is $1/2$. In practice, it is not unusual to have $a - \lambda < 10^{-10}(b-a)$, in which case, the above formula computes a k of 34657. Too avoiding computing polynomial of exceedingly high order, we limit the highest degree of a polynomial to 200, which means if $a - \lambda \geq 3 \times 10^{-6}(b-a)$ we can achieve $\gamma = 1/2$, otherwise we use a 200-degree polynomial.

Figure 5.7 shows the left end of three polynomials chosen for extreme eigenvalues. The interval to be damped is $[0.0001, 1]$. The top half of the figure shows three 100-degree polynomials and the bottom half shows the same polynomials at 200-degree. The values of the polynomials at selected points are shown in table 5.3. The values at 0.0001 is the same as the damping factors. From Figure 5.7 and Table 5.3 we can see that there are significant differences among the three polynomials. In particular, the damping factor of the Chebyshev polynomial may differ significantly from the least-squares polynomial. However, since we don't know how to estimate the damping factors of the Chebyshev polynomial of the second kind and the

least-squares polynomial, using equation (5.4) for all three polynomials is a reasonable alternative.

Based on past experiences reported, for example, [6, 113], we know that the performance of the Chebyshev polynomial does not depend strongly on the exact location of the boundary opposite to the wanted eigenvalue so long as all unwanted eigenvalues are in the Chebyshev interval. In this study, all examples for extreme eigenvalue problem are to find the smallest eigenvalues. In this case, the right end of the Chebyshev interval is not as important as the left end. Thus we have decided to only test two choices for selecting the right end b. They are,

1. Let b be the right end of the Gershgorin interval [51, 139]. This can be determined before the start of the eigenvalue algorithm and used through-out.

2. Let b be the opposite extreme of the spectrum of the projected matrix $H = V^T AV$. If the smallest eigenvalue is sought and the eigenvalues of H are sorted in ascending order, then $b = \lambda_m$, the largest eigenvalue of H, where m is the size of H.

In the second scheme, the interval does not cover all unwanted eigenvalues, it is possible that the component corresponding to the largest eigenvalues will dominate the vector $P(A)x$. One strategy to reduce the impact of this problem is to use only odd degree polynomials for extreme eigenvalue problems. Using odd degree polynomials, the smallest eigenvalues of A and the largest eigenvalues are mapped to values with different signs. Since the Davidson method is effective in separating them, so long as this does not cause overflow in the floating-point numbers, the impact of this potential problem is limited. The same strategy is also used in [6] where only odd degree Chebyshev polynomials are used to precondition linear system of equations.

The value of λ_m is generally much smaller than the right-end of the Gershgorin interval. Given the same left-end for the Chebyshev interval, using the second scheme for b will generate a polynomial with better separation between the wanted eigenvalue and the nearby unwanted ones. This reflects our emphasis on eigenvalues near the wanted ones. We rely on the Davidson method to deal with the whole spectrum of A.

We have chosen three schemes for selecting the left end of the Chebyshev interval a. They are,

1. Let $a = \lambda_i$, where λ_i is an eigenvalue of the projected matrix H. The value λ_1 is considered the next eigenvalue sought. Usually 20 eigenvalues are computed. The first 5 of them could be good candidates. Thus there are 5 choices for this option, a can be λ_2, λ_3, λ_4, λ_5, or λ_6.

2. Let $a = \lambda_{p+1}$, where p is the active window size used in algorithm 4.1. In the examples that will be shown later, the basis size for the Davidson method is 20, the window size is 5 or the actual number of wanted eigenpairs still to be computed. In this case, the value p decreases as number of eigenpairs left becomes less than 5.

3. Choose a value for a so that the estimated damping factor under the given number of matrix-vector multiplications is 1/2. This scheme always use a polynomial of degree 100. It compute the value of a to ensure the damping factor is 1/2. Let $\alpha = 2^{1/100}$, the formula on damping factor, equation (5.4), leads to the following equations.

$$\begin{aligned} a+b-2\lambda+\sqrt{(a+b-2\lambda)^2-(b-a)^2} &\geq \alpha(b-a) \\ (a+b-2\lambda)^2-(b-a)^2 &\geq (\alpha(b-a)-(a+b-2\lambda))^2 \\ 2\alpha(b-a)(a+b-2\lambda) &\geq (\alpha^2+1)(b-a)^2 \\ (\alpha+1)^2 a &\geq (\alpha-1)^2 b+4\alpha\lambda. \end{aligned}$$

In deriving the above inequalities, the following factor is also used, $b > a > \lambda$. From the last equation, we can compute a as follows,

$$a = \frac{(\alpha-1)^2 b + 4\alpha\lambda}{(\alpha+1)^2}.$$

As expressed in equation (5.1), an approximate eigenvector contains components of unwanted eigenvectors, i.e., $\alpha_i \neq 0$, $i = 2, \ldots, n$. To improve the quality of the approximation is to reduce the magnitude of α_i, $i \neq 1$, while maintaining $\|[\alpha_1, \alpha_2, \ldots, \alpha_n]\|$ to be a constant. The Davidson method can be considered as

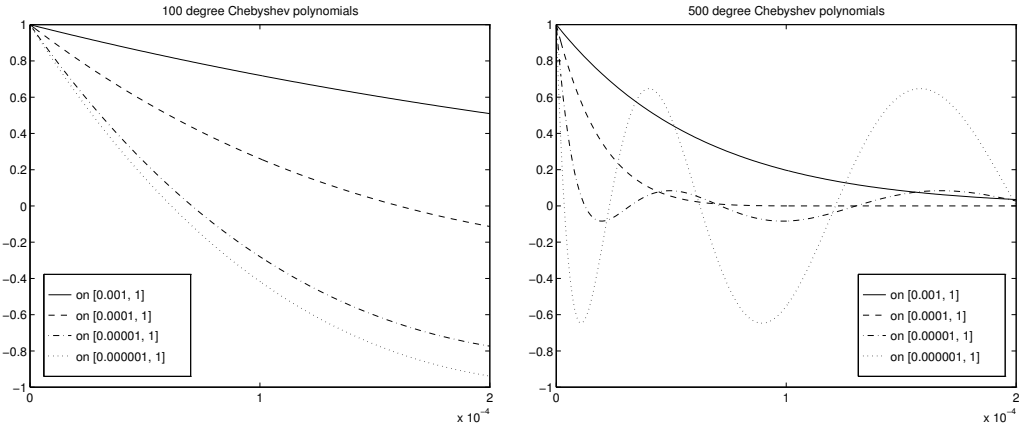

Figure 5.8: High-degree Chebyshev polynomials on different intervals.

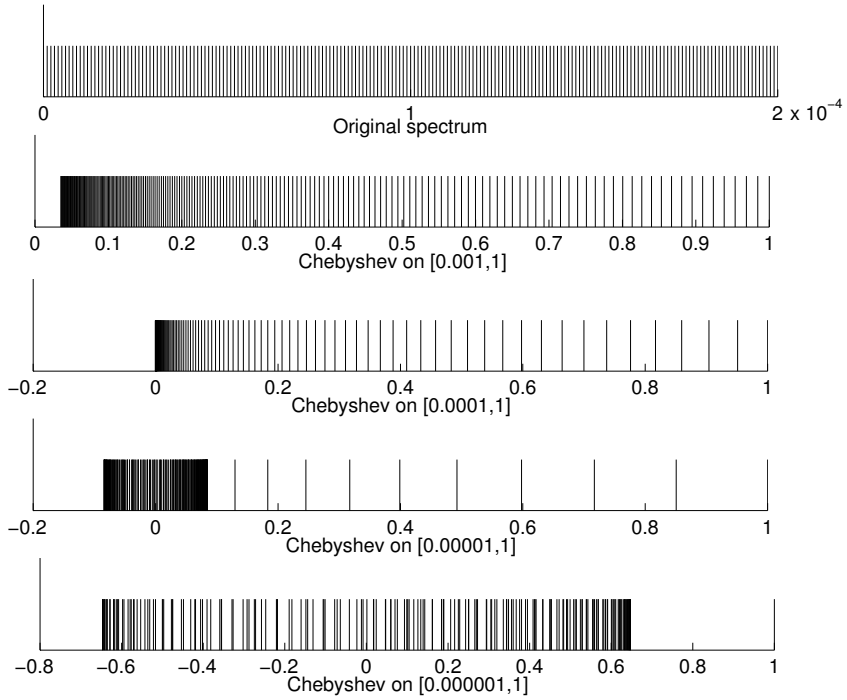

Figure 5.9: A dense spectrum transformed by Chebyshev polynomials on different intervals.

good at damping components corresponding to far away eigenvalues, in other word, it is effective in reducing the magnitude of $\alpha_n, \alpha_{n-1}, \ldots$. One way of using the polynomials is to make them damp the components corresponding to eigenvalues near the wanted eigenvalue, i.e., reducing $\alpha_2, \alpha_3, \ldots$. Figures 5.1, 5.2 and 5.3 show that all three polynomials can be used to damp contributions from large range of eigenvalues. Figure 5.7 confirms that the Chebyshev polynomials have the fastest growth rate outside the interval. Next we will attempt to show how the selection of a might affect the behavior of Chebyshev polynomials near the wanted eigenvalue.

Figures 5.8 and 5.9 offer some intuition on which scheme might be better in improving the quality of

a	$T_{100}(10^{-6})$	$T_{100}(10^{-5})$	$T_{100}(10^{-4})$	$T_{100}(2 \times 10^{-4})$
10^{-3}	0.9968	0.9685	0.7205	0.5095
10^{-4}	0.9903	0.9051	0.2607	-0.1132
10^{-5}	0.9820	0.8257	-0.2797	-0.7736
10^{-6}	0.9799	0.8054	-0.4166	-0.9385
10^{-7}	0.9797	0.8031	-0.4325	-0.9576
10^{-8}	0.9797	0.8028	-0.4341	-0.9595

a	$T_{500}(10^{-6})$	$T_{500}(10^{-5})$	$T_{500}(10^{-4})$	$T_{500}(2 \times 10^{-4})$
10^{-3}	0.9843	0.8531	0.1967	0.0352
10^{-4}	0.9510	0.5980	0.0001	-0.0001
10^{-5}	0.8505	0.0840	-0.0837	0.0268
10^{-6}	0.6471	-0.6411	-0.5533	0.0011
10^{-7}	0.5532	-0.9518	-0.7907	-0.0287
10^{-8}	0.5401	-0.9947	-0.8241	-0.0332

Table 5.4: Values of the Chebyshev polynomial with different intervals.

the eigenvector. Assuming the wanted eigenvalue is 0, the closest unwanted one is 10^{-6} and the largest eigenvalue is 1. The value of the Chebyshev polynomial of degree 100 and degree 500 at the point $x = 10^{-6}$ is shown in table 5.4. One obvious trend in figure 5.8 and table 5.4 is that $T(10^{-6})$ is smaller as a becomes closer to 0. The optimal case in terms of reducing the magnitude of $T(10^{-6})$ is to have $T(10^{-6})$ equal to 0 which is possible by increasing the degree of the Chebyshev polynomial and adjusting the left end of the Chebyshev interval. However, as we decrease the value of $T(10^{-6})$, the value of the polynomial at other points may start to increase. For example, when a is 10^{-8}, $T_{500}(10^{-5})$ is close to -1.

Figure 5.9 shows how the lower end of the spectrum is transformed with Chebyshev polynomials on different intervals. We assume that the original spectrum is uniform in the range of $[0, 0.0002]$, and only show how the 201 eigenvalues in this range is transformed in the figure. The polynomials used in this plot are the 500-degree Chebyshev polynomials shown in figure 5.8 and table 5.4, they transform the 0 eigenvalue of the original matrix into 1. As the lower end of the Chebyshev interval becomes closer to 10^{-6}, the separation between between 1 and the rest of the transformed spectrum becomes larger. This observation mirrors the above observation on the value of the Chebyshev polynomial at 10^{-6}. By increasing the gap between the wanted eigenvalue and the rest, the Lanczos method or the Davidson method can resolve the transformed spectrum much easier than the original one.

If the approximate eigenvector is a linear combination of the first two exact eigenvectors, it is clear that the Chebyshev interval should be chosen to minimize $|T(10^{-6})|$. However, often the approximate eigenvector is a linear combination of a large number of eigenvectors. How to choose a to reduce the residual norm is not nearly as clear as in this trivial case. If the contributions from a few closest eigenvalues does not dominate the error in the approximate eigenvector, then it is more beneficial to choose a slightly larger than 10^{-6}, e.g., $a = 10^{-5}$ in figure 5.9. In the numerical experiment to be conducted later, we will attempt to identify an effective strategy for selecting a.

Denote the interval for the Kernel polynomial by $[a_1, b_2]$, and the 2 intervals for the 2-interval least-squares polynomial as $[a_1, b_1]$ and $[a_2, b_2]$. The interval selection schemes described earlier in this section are applied to the Kernel polynomial and the 2-interval least-squares polynomial as follows.

- Selection of a_1 and b_2,
  1. Let a_1 and b_2 be the left and right ends of the Gershgorin interval.
  2. Let a_1 and b_2 be the smallest and the largest eigenvalue of H_m.
- To simplify the selection of b_1 and a_2 when use the two-interval least-squares polynomial, we use the the following relation so that we only need to select a value for a_2, $\lambda - b_1 = a_2 - \lambda$, where λ is the current approximation of the eigenvalue. The choice of a_2 given λ and b_2 can be made using the same

techniques as selecting a for the one-interval polynomials. It is possible that b_1 is less or equal to a_1 in some cases. If this is true, we will revert back to use one-interval least-square polynomial.

5.5 Polynomials for purification

This section will use some examples to demonstrate how the above mentioned polynomials work as the purification step of an eigenvalue method. The hybrid eigenvalue method we build alternates between the Davidson method and one of the three polynomial methods mentioned above. The Davidson method is immediately restarted if the residual norm has reduced by a factor of 0.6 since last restart, otherwise the polynomial method is invoked before restarting the Davidson method. The Davidson method compute a block of Ritz values and Ritz vectors before restart. When use a polynomial method, we need to make a choice as which ones we would apply the polynomial on. Our first scenario tries to preserve the structure of the thick-restarted Davidson method. It only applies the polynomial on the first Ritz vector and appends the resulting vector at end of the input vectors. More specifically, the thick-restarted Davidson method retains a number of Ritz vectors, say, t, and a block of residual vectors corresponding to the first few Ritz pairs. The Davidson method essentially restarts at step t and tries to incorporate the residual vectors in its basis. This hybrid scheme increase the number of new vectors to be included in the basis by one.

The first example uses a matrix from the Harwell-Boeing collection called 1138BUS [37]. It has 1138 rows and 4054 nonzero elements if stored in CSR format [115]. We seek to find the 5 smallest eigenvalues starting with one initial guess of $[1, 1, \ldots, 1]^T$. As before, the maximum basis size is 20, and the tolerance on the residual norm is $10^{-12}\|A\|_F$ which is 10^{-7} in this case. Without preconditioning, the thick-restarted Davidson method can find 3 eigenpairs in 100,000 matrix-vector multiplications. The hybrid method that produced results shown in table 5.5 alternates between the unpreconditioned Davidson method and one of the three polynomials for extreme eigenvalue problems. There are 7 different choices used for a, of which 5 are based on selection scheme 1 discussed above. The row headed by λ_{p+1} refers to the selection scheme 2, and the last row, λ_γ refers to the third scheme where a is chosen to ensure that the damping factor is $1/2$. The fixed b case uses the right end of the Gershgorin interval as b, and the dynamic b refers to the case of using λ_m as b. Three columns are presented for each case, the first column, headed with "MATVEC (total)", is the total number of matrix-vector multiplications used to compute the five eigenpairs, the second column, headed with "MATVEC (Davidson)", is the number of matrix-vector multiplications used in the Davidson method, and the third one, headed with "time (sec)" is the total execution time in seconds. We run this test on an SGI Challenge with MIPS 4400 processors. The machines used have more than 512 MB real memory. Our test problem fit into the main memory.

A glance at table 5.5 reveals that let $a = \lambda_6$ and use the fixed b with the Chebyshev polynomial of the first kind converges in the least number of matrix-vector multiplications. Overall the two cases using Chebyshev polynomial with fixed b and $a = \lambda_4$ or λ_5 use the least amount of CPU time to reach convergence.

One alternative to applying a polynomial on one Ritz vector is to apply the polynomial on all the Ritz vectors. In the thick-restarted Davidson method, we save a number of Ritz vectors, x_i, and the corresponding Ax_i at restart. If we apply a polynomial on all these Ritz vectors, the corresponding Ax_i are no longer useful. This increases the number of matrix-vector multiplications used by the Davidson method. More importantly, the number of matrix-vector multiplications used to build the polynomial is significantly more in this scheme than in the previous case. Table 5.6 has the results of applying this hybrid eigenvalue method on 1138BUS. Compared with table 5.5, it is clear that many more matrix-vector multiplications are used in this scheme. In addition, table 5.6 shows that there are a number of cases where this hybrid method did not find all 5 eigenpairs in 200,000 matrix-vector multiplications. For these cases, the "MATVEC (total)" column appears as > 200000 and the "MATVEC (Davidson)" column contains a negative number that indicates the number of eigenpairs computed using 200,000 matrix-vector multiplications.

The best two cases in table 5.6 are using the Chebyshev polynomial and the least-squares polynomials with $a = \lambda_6$ and dynamic b. These two cases use the same CPU time to reach convergence. However the Chebyshev polynomial case uses slightly less matrix-vector multiplications. In this test, using dynamic b is generally better than using fixed b which is different from the case where purification is applied on one Ritz vector, see table 5.5.

The smallest number of matrix-vector in table 5.6 is 113311, the corresponding time is 35.8 seconds.

Chebyshev polynomial of the 1st kind

	fixed b			dynamic b		
a	MATVEC (total)	MATVEC (Davidson)	time (sec)	MATVEC (total)	MATVEC (Davidson)	time (sec)
λ_2	42788	3724	24.7	64917	5073	35.8
λ_3	33429	3116	19.7	42391	3925	24.8
λ_4	26750	2912	16.8	44187	4133	26.0
λ_5	25642	3051	16.8	35956	3796	22.3
λ_6	25082	3233	17.0	34697	3794	21.8
λ_{p+1}	33353	3642	21.0	37941	3906	23.2
λ_γ	26406	3896	19.2	37580	5008	25.9

Chebyshev polynomial of the 2nd kind

	fixed b			dynamic b		
a	MATVEC (total)	MATVEC (Davidson)	time (sec)	MATVEC (total)	MATVEC (Davidson)	time (sec)
λ_2	34659	3146	20.2	52048	3936	28.3
λ_3	32291	3056	19.1	54693	4386	30.3
λ_4	35616	3406	21.2	48276	4106	27.4
λ_5	37094	3676	22.3	50061	4496	29.0
λ_6	37953	3916	23.3	48977	4746	29.2
λ_{p+1}	35528	3756	22.0	41099	4096	24.8
λ_γ	44366	5596	29.8	38379	4866	25.8

least-squares polynomial on one interval

	fixed b			dynamic b		
a	MATVEC (total)	MATVEC (Davidson)	time (sec)	MATVEC (total)	MATVEC (Davidson)	time (sec)
λ_2	51457	3986	28.4	58709	4367	31.9
λ_3	48466	3836	26.9	55415	4386	30.7
λ_4	46388	3967	26.5	55263	4666	31.4
λ_5	47543	4171	27.4	54153	4696	31.0
λ_6	43891	4411	26.6	54013	5146	32.1
λ_{p+1}	42270	4106	25.3	50917	4727	29.9
λ_γ	50595	6276	33.7	40957	5136	27.4

Table 5.5: Results of using polynomial purification with the Davidson method on 1138BUS.

The smallest number of matrix-vector multiplications in table 5.5 is 25082 and the corresponding time is 17 seconds. Considering the dramatic differences in number of matrix-vector multiplications between tables 5.5 and 5.6, the differences in time is relatively small. This is because the majority of the matrix-vector multiplications are done on a large block of vectors which could be made more efficient compared to multiplying one single vector. Comparing tables 5.5 and 5.6, we see that the matrix-vector multiplications used by the Davidson method is significantly less in the second case. On the environments where the matrix-vector multiplication on a block of vectors is relatively cheap, using more block matrix-vector multiplication to reduce the number of steps taken in the Davidson method could potentially reduce the total execution time. Under this circumstance, applying the polynomials on all Ritz vectors could be a more effective alternative.

The two kinds of Chebyshev polynomial and the least-squares polynomials are computed through three-term recurrences which means the last three polynomials are available without extra work. If we only give one vector v to the polynomial routines, we could get three different results, $P_k(A)v$, $P_{k-1}(A)v$ and $P_{k-2}(A)v$. Instead of only adding $P_k(A)v$ as new input vectors to the Davidson routine, we could use two or three of them. Table 5.7 shows the results of giving the first two to the Davidson routine, and table 5.8 shows

Chebyshev polynomial of the 1st kind

	fixed b			dynamic b		
a	MATVEC (total)	MATVEC (Davidson)	time (sec)	MATVEC (total)	MATVEC (Davidson)	time (sec)
λ_2	>200000	-3	-	>200000	-3	-
λ_3	>200000	-3	-	>200000	-3	-
λ_4	>200000	-3	-	160853	2937	48.8
λ_5	>200000	-3	-	180587	3171	54.4
λ_6	153702	3420	47.9	113311	2882	35.8
λ_{p+1}	>200000	-4	-	184306	3535	56.1
λ_γ	>200000	-3	-	>200000	-3	-

Chebyshev polynomial of the 2nd kind

	fixed b			dynamic b		
a	MATVEC (total)	MATVEC (Davidson)	time (sec)	MATVEC (total)	MATVEC (Davidson)	time (sec)
λ_2	192249	2466	56.2	150006	2265	44.7
λ_3	>200000	-3	-	145180	2466	43.8
λ_4	>200000	-3	-	130925	2577	40.1
λ_5	158157	2947	48.0	127278	2769	39.4
λ_6	142940	3064	44.1	128572	3019	40.2
λ_{p+1}	158868	3064	48.3	143397	3147	44.4
λ_γ	191437	4250	59.1	170904	4133	53.5

least-squares polynomial on one interval

	fixed b			dynamic b		
a	MATVEC (total)	MATVEC (Davidson)	time (sec)	MATVEC (total)	MATVEC (Davidson)	time (sec)
λ_2	176501	2528	51.1	145447	2392	42.8
λ_3	136493	2315	40.2	128051	2321	38.1
λ_4	150863	2847	45.0	125682	2427	37.5
λ_5	142009	2837	42.6	117970	2712	36.0
λ_6	132885	2964	40.3	114823	2997	35.8
λ_{p+1}	140368	3132	42.7	119850	2898	36.8
λ_γ	>200000	-3	-	142603	3774	44.4

Table 5.6: Results of applying polynomial purification on all Ritz vectors when restart.

the results of using all three of them. Overall, all three polynomials benefited from using more than one output vectors. Among them, the least-squares polynomial benefited the most. The execution time of the hybrid method decreased from table 5.5 to table 5.8 for most of the cases. Overall, the best combination for each polynomial is shown in table 5.9. In the table, the numbers 2 and 3 indicates the number of output polynomials used. Overall, the hybrid method using the last three least-squares polynomials is the most successful scheme tested. This best scheme uses dynamic b and $a = \lambda_2$. It computed the five smallest eigenvalues of 1138BUS with 16,863 matrix-vector multiplications. This is considerably better than using the Davidson method alone which only computed 3 eigenpairs in 100,000 matrix-vector multiplications.

To test the polynomials for interior eigenvalues, we have chosen to find 5 eigenvalues of PLAT1919 near 0.0001 as the test case [37]. The maximum basis size is again 20 and the tolerance on the residual norm is $10^{-12}\|A\|_F = 2 \times 10^{-11}$. The only starting vector for the Davidson method is $[1, 1, \ldots, 1]^T$ as before. The Davidson method for interior eigenvalue problem restarts with 10 Ritz vectors whose corresponding Ritz values are close to 0.0001. Since the Rayleigh-Ritz projection algorithm does not possess the same optimality property for interior Ritz values as for extreme eigenvalues, the behavior of the hybrid method behaves very differently from the extreme eigenvalue case. We have repeated all 126 different combinations

Chebyshev polynomial of the 1st kind

	fixed b			dynamic b		
a	MATVEC (total)	MATVEC (Davidson)	time (sec)	MATVEC (total)	MATVEC (Davidson)	time (sec)
λ_2	46403	3940	26.1	41621	3872	24.2
λ_3	37278	3533	21.9	32982	3215	19.5
λ_4	32428	3197	19.3	25313	2958	16.2
λ_5	28355	3093	17.6	25315	3029	16.3
λ_6	27748	3147	17.5	21865	2927	14.9
λ_{p+1}	41323	3840	24.0	34375	3603	20.9
λ_γ	29908	4277	21.0	24003	3575	17.2

Chebyshev polynomial of the 2nd kind

	fixed b			dynamic b		
a	MATVEC (total)	MATVEC (Davidson)	time (sec)	MATVEC (total)	MATVEC (Davidson)	time (sec)
λ_2	20429	2145	12.5	18824	2057	11.7
λ_3	21314	2377	13.4	18751	2139	11.9
λ_4	21329	2507	13.7	20047	2357	12.8
λ_5	20493	2540	13.5	18377	2445	12.4
λ_6	20861	2627	13.8	18487	2597	12.9
λ_{p+1}	21740	2683	14.2	17722	2487	12.3
λ_γ	25698	3747	18.3	19210	3038	14.2

least-squares polynomial on one interval

	fixed b			dynamic b		
a	MATVEC (total)	MATVEC (Davidson)	time (sec)	MATVEC (total)	MATVEC (Davidson)	time (sec)
λ_2	18868	1968	11.5	19655	2029	11.9
λ_3	19443	2157	12.2	19593	2211	12.3
λ_4	18362	2275	12.1	18183	2237	11.9
λ_5	18788	2261	12.2	18527	2357	12.3
λ_6	19076	2368	12.5	19357	2690	13.4
λ_{p+1}	18151	2337	12.1	18935	2468	12.7
λ_γ	21459	3203	15.4	18760	2897	13.7

Table 5.7: Results of polynomial purification where the last two polynomials are used.

of parameters shown in tables 5.5, 5.7 and 5.8, table 5.10 only shows the cases where the hybrid method successfully computed 5 eigenpairs around 0.0001 within 400,000 matrix-vector multiplications. In table 5.10, the column headed by N_p is the number of output vectors used and the "least-squares" refers to the 2-interval least-squares polynomial. For the Kernel polynomials we only need to determine the outer boundaries, however, we use one of the seven choices for inner boundaries of the two-interval least-squares polynomial to determine the degree of the Kernel polynomial. Using λ_γ as inner boundary indicates that the degree of the polynomial is set to 100. From this table we see that the only successful scheme for selecting the boundaries of the intervals is to use λ_γ as inner boundaries and use the boundaries of the Gershgorin interval as the outer.

The intermediate approximate solution to an interior eigenvalue can be either larger or smaller than the final solution. However for extreme eigenvalues, the approximate eigenvalues of the Davidson method approaches the exact solution from the inside of the spectrum. This difference could be the reason why the successful schemes for selecting the boundaries for the extreme eigenvalue problems are not successful schemes for selecting inner boundaries for the 2-interval least-squares polynomials. Generally, using λ_γ to compute the inner boundaries given more consistent answer to the inner boundaries. Thus it is a more

Chebyshev polynomial of the 1st kind

	fixed b			dynamic b		
a	MATVEC (total)	MATVEC (Davidson)	time (sec)	MATVEC (total)	MATVEC (Davidson)	time (sec)
λ_2	43097	3693	24.2	52168	4300	28.9
λ_3	35242	3259	20.4	40430	3610	23.0
λ_4	28483	3116	17.7	28617	3158	17.8
λ_5	27375	3152	17.3	24953	3050	16.2
λ_6	26510	3123	16.9	22794	3130	15.7
λ_{p+1}	31849	3391	19.5	34535	3750	21.3
λ_γ	29458	4328	20.8	23496	3650	17.1

Chebyshev polynomial of the 2nd kind

	fixed b			dynamic b		
a	MATVEC (total)	MATVEC (Davidson)	time (sec)	MATVEC (total)	MATVEC (Davidson)	time (sec)
λ_2	22518	2358	13.8	18056	1990	11.2
λ_3	21120	2403	13.4	17527	2102	11.3
λ_4	21761	2561	14.0	19395	2288	12.5
λ_5	21458	2660	14.0	19015	2488	12.8
λ_6	20867	2709	14.0	18284	2548	12.7
λ_{p+1}	20536	2661	13.7	18492	2570	12.8
λ_γ	26885	3960	19.1	20612	3163	15.0

least-squares polynomial on one interval

	fixed b			dynamic b		
a	MATVEC (total)	MATVEC (Davidson)	time (sec)	MATVEC (total)	MATVEC (Davidson)	time (sec)
λ_2	19974	2045	12.1	16863	1858	10.5
λ_3	19996	2059	12.1	17655	2204	11.6
λ_4	19186	2218	12.2	18591	2249	12.0
λ_5	18633	2438	12.5	17173	2300	11.6
λ_6	21009	2468	13.4	17600	2488	12.2
λ_{p+1}	20326	2468	13.1	17419	2428	12.0
λ_γ	25263	3479	17.3	22534	3129	15.5

Table 5.8: Results of polynomial purification where the last three polynomials are used.

	fixed b	dynamic b
Chebyshev I	2, $a = \lambda_6$	2, $a = \lambda_6$
Chebyshev II	2, $a = \lambda_2$	3, $a = \lambda_2$
least-squares	2, $a = \lambda_2$	3, $a = \lambda_2$

Table 5.9: The best choices for different polynomials in hybrid method.

Polynomial	N_p	inner boundaries	outer boundaries	MATVEC (total)	MATVEC (Davidson)	time (sec)
least-squares	2	λ_γ	fixed	182471	19201	267.3
least-squares	3	λ_γ	fixed	231633	23889	336.6
Kernel	2	λ_γ	fixed	182471	19201	267.4

Table 5.10: Hybrid methods that successful computed 5 eigenvalues around 0.0001 for PLAT1919.

preconditioner	MATVEC (total)	MATVEC (Davidson)	time (sec)
CG$(A-\delta I)$	12956	71	5.8
CG$(A-\delta I+xx^T)$	13356	73	6.4
CG$(A-\delta I-2x(Ax)^T)$	13823	76	6.7
CG$((I-xx^T)(A-\delta I)(I-xx^T))$	13272	72	7.0
Chebyshev II ($a=\lambda_\zeta$, fixed b)	4181866	58385	1690

Table 5.11: Finding 5 smallest eigenvalues of 1138BUS with preconditioners that only require matrix-vector multiplications.

suitable choice compare to the other schemes.

The number of matrix-vector multiplication taken to compute this desired eigenvalues is very high compared with the matrix size. Nevertheless, since we can not even compute one eigenpair around with the plain Davidson method, finding 5 of them with 182,471 matrix-vector multiplications is a significant improvement.

5.6 Polynomial preconditioning

In this section, polynomials are used as preconditioners to the Davidson method, see algorithm 4.1. The first modification was to simply replace step 2.d.iii of algorithm 4.1 with

$$Z = P(A)r.$$

This preconditioning scheme is different from all previous schemes mentioned in this thesis since it does not attempt to solve a linear system.

An advantage of using polynomial preconditioning is that it is easy. Using Krylov methods as preconditioners is similar to polynomial preconditioners. For this reason, we will also show examples of Krylov iterative methods as preconditioners in this section.

For numerical testing, we apply the preconditioners to the two test problems used before, i.e., the extreme eigenvalues of 1138BUS and interior eigenvalues of PLAT1919. The convergence tolerance and the maximum basis size for the Davidson method are the same as before. Unlike the hybrid method case where the polynomial purification step may be skipped if the Davidson iteration is converging rapidly, the preconditioning step is always executed. However, since the workspace required by the polynomial routines is the unused portion of the basis vectors, it is possible that there is not enough workspace to carry out the polynomial procedure, which happens when the basis is almost full. In this case, no preconditioning is performed, we simply copy the input vectors to the output array. In addition, to ensure that there is a reasonable eigenvalue estimate to establish the intervals for the polynomial procedures, the polynomial preconditioning routines are not invoked for the first few steps of the Davidson iteration. In the tests shown, the polynomial preconditioning procedure is not used until the basis size is 10 in the Davidson method.

Table 5.11 shows the results of computing the smallest eigenvalues of the 1138BUS matrix using the polynomial preconditioner and four CG based preconditioners. The preconditioners CG$(A-\delta I)$, CG$(A-\delta I+xx^T)$, CG$(A-\delta I-2x(Ax)^T)$ and CG$((I-xx^T)(A-\delta I)(I-xx^T))$ use the Conjugate Gradient method to compute an approximate solution of $B^{-1}r$ where the matrix B is the matrix in the parenthesis, see Chapter 3 for the origin of these matrices. The maximum number of matrix-vector multiplications is set to 200 for CG. The iteration is terminated when the residual norm is reduced by one order of magnitude or 200 matrix-vector multiplications have been used. Compared with the best results reported in tables 5.5, 5.7 and 5.8, the Davidson method with one of the four CG preconditioners takes significantly less time and fewer matrix-vector multiplications than the best hybrid method scheme. The total numbers of matrix-vector multiplications used by the four CG schemes are not very different, and the simplest of the four, CG$(A-\delta I)$, is slightly more effective than the others.

In this test, the polynomial preconditioners tested include all 42 schemes shown in table 5.5. In table 5.11 only one case is shown because it is the only case where the polynomial preconditioner scheme is successful

Chebyshev polynomial of the 1st kind

	fixed b			dynamic b		
a	MATVEC (total)	MATVEC (Davidson)	time (sec)	MATVEC (total)	MATVEC (Davidson)	time (sec)
λ_2	114642	1443	45.3	170118	2135	66.9
λ_3	55442	713	21.9	82662	1053	32.4
λ_4	27664	379	10.9	44608	599	17.6
λ_5	26264	374	10.4	41017	571	16.2
λ_6	28567	417	11.4	46002	659	18.3
λ_{p+1}	93490	1225	36.9	150120	1953	59.0
λ_γ	21970	559	9.4	34494	869	14.6

Chebyshev polynomial of the 2nd kind

	fixed b			dynamic b		
a	MATVEC (total)	MATVEC (Davidson)	time (sec)	MATVEC (total)	MATVEC (Davidson)	time (sec)
λ_2	21348	283	8.4	35999	467	14.1
λ_3	20354	279	8.0	35470	469	13.9
λ_4	22556	319	8.9	36085	491	14.2
λ_5	21372	319	8.5	34670	489	13.7
λ_6	21790	339	8.7	34690	509	13.8
λ_{p+1}	21790	339	8.7	34690	509	13.8
λ_γ	27719	701	11.8	41051	1031	17.4

least-squares polynomial on one interval

	fixed b			dynamic b		
a	MATVEC (total)	MATVEC (Davidson)	time (sec)	MATVEC (total)	MATVEC (Davidson)	time (sec)
λ_2	27165	357	10.8	40807	527	16.1
λ_3	25930	349	10.3	42045	551	16.6
λ_4	27097	377	10.8	43082	579	17.0
λ_5	26062	379	10.4	44191	627	17.5
λ_6	25785	391	10.3	44364	639	17.6
λ_{p+1}	26388	399	10.6	43359	627	17.2
λ_γ	31353	791	13.3	50745	1271	21.4

Table 5.12: Results of applying the Davidson method on 1138BUS with the modified polynomial preconditioner.

in finding all five smallest eigenvalues of 1138BUS. This indicates that appending $P(A)r$ to the basis of the Davidson method is not a good option. An obvious alternative is to apply the polynomial on the current approximate eigenvector, i.e., append $P(A)x$ to the Davidson basis. In fact, we found that a more robust approach is to append both r and $P(a)x$ to the Davidson basis. Table 5.12 shows the results of this modified preconditioning scheme.

To solve the 1138BUS problem, the most effective hybrid scheme tested used 10.5 seconds, 16,863 matrix-vector multiplications, see table 5.8. The Davidson method with the best modified polynomial preconditioner used only 8.0 seconds, 20,354 matrix-vector multiplications, see table 5.12. The modified preconditioning scheme uses less time but more matrix-vector multiplications to solve the same problem. This is because the matrix-vector multiplication is fairly inexpensive with 1138BUS. This scheme trades the number of steps used in the Davidson method with the number of matrix-vector used in the polynomial methods. Since one Davidson step in more expensive than one step in the polynomial method, the Davidson method with this modified preconditioner can reduce the execution time. However, Compared with the CG preconditioners, the modified polynomial preconditioning still uses more time and more matrix-vector multiplications.

preconditioner	MATVEC (total)	MATVEC (Davidson)	time (sec)
CG$(A-\delta I)$	591090	3090	747
CG$(A-\delta I+xx^T)$	373783	2183	600
CG$(A-\delta I-2x(Ax)^T)$	249035	1435	407
CG$((I-xx^T)(A-\delta I)(I-xx^T))$	112219	619	193

Table 5.13: The results of finding 5 interior eigenvalues of PLAT1919 with preconditioners that only require matrix-vector multiplications.

Polynomial	inner boundaries	outer boundaries	MATVEC (total)	MATVEC (Davidson)	time (sec)
least-squares	λ_3	dynamic	400963	6439	505
Kernel	λ_3	dynamic	327854	4147	414
Kernel	λ_4	dynamic	359443	4583	467
Kernel	λ_6	dynamic	390790	5003	511

Table 5.14: The modified polynomial preconditioning schemes that successfully computed the desired eigenpairs of PLAT1919.

Among the three polynomials, the Chebyshev polynomial of the second kind shows the best performance gain. Also, using fixed b is better than using dynamic b for preconditioning.

Table 5.13 shows the results of applying the Davidson method with matrix-vector multiplication based preconditioners on PLAT1919 to find 5 eigenpairs around 0.0001. Similar to table 5.11, we only show the successful cases. The preconditioning schemes that compute $P(A)r$ did not reach convergence within 400,000 matrix-vector multiplications. In this case, the four CG preconditioners uses significantly different number of matrix-vector multiplications and CPU time. The best among the four preconditioners is the Jacobi-Davidson preconditioning scheme.

Table 5.14 shows the results of using the modified preconditioning scheme on the PLAT1919 test problem. Again, we only show those cases where the preconditioned Davidson method were able to compute 5 eigenpairs within 400,000 matrix-vector multiplications. Among the cases shown in table 5.14, we see that using Kernel polynomial with dynamic outer boundaries is relatively more successful than other schemes. Compared to the most successful hybrid methods, see table 5.10, we observe that the modified polynomial preconditioner is not as effective as the hybrid method. The shortest time in table 5.10 is 267 seconds which is considerable less than the shortest time in table 5.14, 414 seconds. Similar to the 1138BUS case, the modified preconditioning scheme is effective in reducing the number of steps taken by the Davidson method. However, the reduction come with the increase in total number of matrix-vector multiplications used. Since the matrix-vector multiplication is more expensive with PLAT1919, the overall CPU time increased in this case. The best result from the CG preconditioners is again considerably better than the best results from using the five polynomials.

Note that it is probably more appropriate to call this modified preconditioning scheme a hybrid method, since it does not resemble the common form of the eigen-system preconditioning or linear system preconditioning. This hybrid method can be considered as one extreme form since a polynomial step is inserted after every step of the Davidson iteration. The hybrid scheme presented in previous section only invokes a polynomial procedure after the Davidson method has built a full basis. From the two tests performed here, we see that invoking the polynomial method more often is only beneficial if the matrix-vector multiplication is inexpensive.

5.7 Polynomial spectrum transformation

The essence of polynomial spectrum transformation technique is to replace the matrix-vector multiplication of the Davidson algorithm with a polynomial in A. However, since we have to perform Rayleigh-Ritz projection to compute the actual eigenvalue of A and to estimate parameters required to define the polynomial, the overall structure of the actual eigenvalue routine is very similar to a hybrid method. Our implementation of the polynomial spectrum transformation technique alternates between the Davidson iteration on the original eigenvalue problem and the Davidson iteration on the transformed eigenvalue problem. The iterations on the original eigenvalue problem are used to compute the eigenvalue of A after the transformed eigenvalue problem has converged and compute the approximate eigenvalues and eigenvectors to be used define the transformed eigenvalue problem. Similar to the hybrid method, if the iteration on the original problem is effective in reducing the residual norm, than we restart to continue the iteration on the original problem. However if the residual norm is decreasing slowly, or the residual norms at the end of last two iterations are both larger than the previous one, then the routine switches to the Davidson method on a transformed eigenvalue problem. The polynomials used for this routine are the same as those used in the hybrid methods and the polynomial preconditioning. The schemes for selecting the intervals are the same as in the polynomial purification case.

Similar to the preconditioning case, the polynomial routines also use the unoccupied part of the basis vectors as workspace. This means that the actual basis size for the transformed Davidson iterations is 1 or 2 less than the basis size for the Davidson iteration on the original eigenvalue problem. The two Chebyshev polynomial routines and the two least-squares polynomial routines require two extra vectors for their computation and the Kernel polynomial routine requires three extra vectors. When the basis size for the Davidson method without any transformation is 20, if the Kernel polynomial is used to perform the spectrum transformation, the actual basis size is 18, if one of the other four polynomials is used, the basis size is 19. The Davidson iteration on the transformed eigenvalue problem is terminated if the residual norm of the largest eigenvalue of the transformed matrix is less than 10^{-10}, or if the residual norm increases compared to previous iteration. Thick restart is applied to the transformed Davidson iterations as well. As before, 10 Ritz vectors corresponding to the 10 largest Ritz values are saved as starting vectors for the new Davidson iteration.

We again use 1138BUS as the test matrix for extreme eigenvalue problem to show how the polynomial spectrum transformation technique works. The same residual tolerance and the maximum basis size are used as before. Table 5.15 shows the results of using the Davidson method with the polynomial spectrum transformation. This table shows all 42 different combinations of the parameters. Among the 7 schemes for selecting a, the most successful one in this case is $a = \lambda_\gamma$. Using the dynamic b is more effective for the least-squares polynomial, while using the fixed b is more effective for the two kinds of Chebyshev polynomials. Overall, the spectrum transformation using the least-squares polynomial with $a = \lambda_\gamma$ and $b = \lambda_m$ is the most effective choice shown in table 5.15.

Majority of the 42 cases shown in table 5.15 are significantly better than unpreconditioned Davidson method on the same eigenvalue problem. However, the best hybrid schemes presented in tables 5.5, 5.7 and 5.8, use less time and fewer matrix-vector multiplications to solve the same problem as the best results from the polynomial spectrum transformation technique. In general, the spectrum transformation technique present also use more time than the modified polynomial preconditioning scheme shown in the previous section.

For interior eigenvalue problems, we were unable to have the polynomial spectrum transformation scheme to converge for the PLAT1919 problem within 400,000 matrix-vector multiplications. For this reason, we switched to the matrix PLAT362 which has similar origin as the matrix PLAT1919. We again seek to find 5 eigenvalues near 0.0001 with residual tolerance $10^{-12}||A||_F$. For comparing the effectiveness of the polynomial spectrum transformation scheme, we also repeated the test with the four CG based preconditioners. The results with CG preconditioners are shown in table 5.16. The cases where the polynomial spectrum transformation scheme reached convergence within 400,000 matrix-vector multiplications are shown in table 5.17.

The best time shown in table 5.17 is 6.9 seconds to find 5 interior eigenvalues of PLAT362. The time using the CG($A-\delta I+xx^T$) to solve the same problem is 1.7 seconds. It is clear that using CG preconditioner with the Davidson method is more effective in this case. In fact, the Davidson method with three out of the

Chebyshev polynomial of the 1st kind

	fixed b			dynamic b		
a	MATVEC (total)	MATVEC (Davidson)	time (sec)	MATVEC (total)	MATVEC (Davidson)	time (sec)
λ_2	93744	184	36.2	84072	284	32.8
λ_3	50942	184	20.4	>100000	-4	-
λ_4	41640	214	16.6	44137	334	17.9
λ_5	43112	224	17.3	77326	444	31.5
λ_6	37250	224	15.6	35022	364	14.7
λ_{p+1}	101372	224	39.7	72885	424	29.0
λ_γ	32405	184	13.3	35260	244	14.5

Chebyshev polynomial of the 2nd kind

	fixed b			dynamic b		
a	MATVEC (total)	MATVEC (Davidson)	time (sec)	MATVEC (total)	MATVEC (Davidson)	time (sec)
λ_2	42374	174	17.0	>100000	-4	-
λ_3	42973	294	17.1	102114	584	40.3
λ_4	42695	304	17.2	73266	604	29.2
λ_5	43636	224	17.7	74095	474	29.8
λ_6	46964	374	19.2	>100000	-4	-
λ_{p+1}	45537	314	18.5	>100000	-3	-
λ_γ	40665	284	17.0	49180	284	20.3

least-squares polynomial on one interval

	fixed b			dynamic b		
a	MATVEC (total)	MATVEC (Davidson)	time (sec)	MATVEC (total)	MATVEC (Davidson)	time (sec)
λ_2	41438	184	16.5	63111	294	24.6
λ_3	38788	214	15.3	36202	214	14.3
λ_4	41564	204	16.6	57664	284	22.8
λ_5	41096	254	16.6	36215	295	14.8
λ_6	43322	294	17.6	52316	264	21.4
λ_{p+1}	42966	294	17.5	63078	304	25.5
λ_γ	34198	184	14.1	31310	234	12.8

Table 5.15: Results of using polynomial spectrum transformation with the Davidson method on 1138BUS.

preconditioner	MATVEC (total)	MATVEC (Davidson)	time (sec)
CG$(A-\delta I)$	9238	52	2.3
CG$(A-\delta I+xx^T)$	6201	37	1.7
CG$(A-\delta I-2x(Ax)^T)$	16244	88	4.4
CG$((I-xx^T)(A-\delta I)(I-xx^T))$	619819	3419	173.3

Table 5.16: Results of finding 5 interior eigenvalues of PLAT362 with CG preconditioners.

Polynomial	inner boundaries	outer boundaries	MATVEC (total)	MATVEC (Davidson)	time (sec)
least-squares	λ_3	dynamic	400963	6439	505.0
Kernel	λ_2	dynamic	253583	1393	58.6
Kernel	λ_3	dynamic	147595	878	34.4
Kernel	λ_4	dynamic	28427	341	6.9
Kernel	λ_6	dynamic	198166	1454	46.8
Kernel	λ_γ	dynamic	201929	991	46.8

Table 5.17: The polynomial spectrum transformation schemes that successfully computed the desired eigenpairs of PLAT362.

preconditioner	MATVEC (total)	MATVEC (Davidson)	time (sec)
NONE	3758	3758	421
SOR	1489	1489	297
CG$(A-\delta I)$	51648	307	1272
CG$(A-\delta I+xx^T)$	44054	254	1122
CG$(A-\delta I-2x(Ax)^T)$	46610	266	1209
CG$((I-xx^T)(A-\delta I)(I-xx^T))$	>1,800,000	-9	-

Hybrid methods with polynomials for extreme eigenvalues

Polynomial	N_p	a	b	MATVEC (total)	MATVEC (Davidson)	time (sec)
Chebyshev I	1	λ_5	fixed	4139	3881	432
Chebyshev II	1	λ_γ	fixed	4272	3850	432
least-squares	1	λ_2	fixed	4272	3850	432

Hybrid methods with polynomials for interior eigenvalues

Polynomial	N_p	inner boundaries	outer boundaries	MATVEC (total)	MATVEC (Davidson)	time (sec)
least-squares	1	λ_γ	fixed	4272	3850	432
Kernel	1	λ_2	fixed	4272	3850	432

Table 5.18: MATVEC and time used to find 60 smallest eigenvalues of the Si_6 test problem.

four CG preconditioners can solve the PLAT362 problem in less than 6.9 seconds.

Comparing table 5.13 with table 5.16 we see that the same preconditioners performs quite differently on two similar eigenvalue problems. This dramatic difference could be due to the special nature of the interior eigenvalue problem or due to some features of the preconditioners. At this moment, we don't yet know which one in the main cause.

5.8 Computing a large number of extreme eigenvalues

The ultimate goal of this study is to use polynomials to improve the efficiency of computing a large number of extreme eigenvalues. To see how the polynomial schemes perform for this task we decided to conduct some tests on the matrix generated from the Si_6 test case, see table 4.1. We compute 60 smallest eigenvalues from the matrix starting with the lone initial guess $[1, 1, \ldots, 1]^T$. To reduce the number of test cases, we have decided to use the hybrid schemes discussed in section 5.5. There are a total of 210 possible test cases with 5 different polynomials, see tables 5.5, 5.7, 5.8, and 5.10. We have chosen to show only the cases with shortest execution time for each of the 5 polynomials in table 5.18. As references, we also show the matrix-vector

multiplication and time required to solve the same problem with a small number of preconditioners. The execution time shown here is gathered from a SGI challenge workstation. On the eigenvalue problem tested, the Davidson method with SOR preconditioner is the only case where the execution time is less than the unpreconditioned Davidson method. The four CG preconditioners are able to reduce the number of iterations spent by the Davidson method, however, the preconditioning steps was too expensive to reduce the overall execution time. The hybrid methods used more matrix-vector multiplications than the unpreconditioned Davidson method. Because the matrix-vector multiplication is relatively expensive, the average number of non-zero elements per row is about 37 for the Si_6 matrix, more CPU time is used by the hybrid methods than the plain Davidson method.

There could be many reasons for the outcome of this test. For example, the hybrid methods may be not effective for computing many eigenpairs, or, the hybrid schemes may not work well with the locking strategy, or, simply the test problem may be a bad example to use. However this one example does not negate the previous observations that polynomial methods can be used effectively to solve eigenvalue problems. What this example does remind us is that continued research is required to improve the robustness of the hybrid schemes.

5.9 Summary

In this chapter, we studied how to use polynomials to facilitate the Davidson method in computing eigenpairs of large sparse matrices. Among the three techniques tested, the purification technique is the most versatile one. The hybrid methods that alternate between the Davidson iteration and the polynomial purification schemes are effective. Using polynomials as preconditioners for the Davidson method is found to be ineffective on the test problems. The polynomial spectrum transformation scheme has sound theoretical basis but in the limited number of tests conducted, we were unable to surpass the hybrid scheme. Overall, the effectiveness of all three different techniques of using polynomials is sensitive to how parameters are selected.

The most successful hybrid method for extreme eigenvalue problems in our tests is the one that alternates between the Davidson method and the least-squares polynomial. For this polynomial, the most successful scheme for selecting the interval to build polynomials for extreme eigenvalue problem is to select the interval as $[\lambda_2, \lambda_m]$. In the numerical tests, we also see that saving lower degree polynomials can also enhance the performance of the hybrid methods, see table 5.9.

The test results for interior eigenvalue problems are not as extensive as the extreme eigenvalue case. This is not because we have conducted fewer tests, but because there are fewer cases where the interior eigenvalue problem was solved successfully. This underscores the need for better selection schemes for the parameters of the polynomials used or better polynomials for interior eigenvalue problems. Based on the available results, we see that the hybrid methods with the least-squares polynomial or the Kernel polynomial in the purification step solved the same test problem in about the same time. The best scheme for selecting the inner boundaries for the two-interval least-squares polynomial is to set the boundaries far enough to ensure a constant damping factor. The outer boundaries for the the least-squares polynomial and the Kernel polynomial can simply be the end points of the Gershgorin interval. The experiments show that it is more effective if the two highest degree polynomials generated by the polynomial routine are used, see table 5.10.

For the three eigenvalue problems tested in this chapter, the Davidson method with CG preconditioners was able to reach convergence in less time than the best of the three types of polynomial schemes. This can be partly blamed on the weakness of the interval selection schemes tested.

The results from the Si_6 test case, see table 5.18, show another aspect of the CG preconditioners and the hybrid method. In this test problem, the CG preconditioners are too expensive for a good overall performance. The polynomial schemes are ineffective as well. But because the Davidson iterations are effective in improving the quality of the solutions, the polynomial purification step is rarely invoked which automatically limits the damage that might be incurred by constructing a high-degree polynomial. In comparison, the hybrid methods are more effective than the CG preconditioner in this case. However, the two different schemes both take more time to find the desired solution than the unpreconditioned Davidson method.

In conclusion, if only matrix-vector multiplication is available, using a Krylov method as preconditioner for the Davidson method is an effective approach. Hybrid method with polynomial purification or polynomial

spectrum transformation could be effective in some cases. For lack of good strategies of selecting parameters, the polynomial based schemes are not competitive against Krylov methods on the problems tested.

Chapter 6

Summary and Future Work

6.1 Summary

Our main goal in this thesis was to study ways to improve algorithms for large sparse eigenvalue problem. In particular, the eigenvalue problems that require a large number of eigenvalues and corresponding eigenvectors, for example, finding 1000 eigenpairs of a $100,000 \times 100,000$ matrix. These problems are challenging because of two reasons. First, the matrices involved are too large for traditional QR approaches which were designed for dense matrices. Second, the effective algorithms for sparse eigenvalue value problems are designed to find only a few eigenpairs. We addressed this problem from three different prospectives: seeking a sound preconditioned algorithm for multiple eigenpairs, improving the preconditioning scheme, and studying the use of polynomials in enhance the flexibility of the eigenvalue routine.

Seeking a sound preconditioned algorithm for multiple eigenpairs. After evaluating eigenvalue algorithms such as the QR method, the Lanczos algorithm, the Arnoldi method, and the Davidson method, we come to the conclusion that the Arnoldi method and the Davidson method are the two most promising algorithms for the task of find any eigenvalues and eigenvectors of large sparse matrices. In addition, we also studied three different variants of these two methods. Without preconditioning, it is clear that the five methods are identical to each other if there is only one initial starting vector to all five methods, see section 2.4. If they also restart with same Ritz vectors, they will continue to produce the same basis and the same approximate solutions. For this reason, the cheaper the algorithm is per step, the less time it will take to find the same solution in exact arithmetic. The simplest algorithm among the five is the Arnoldi method. Therefore, the Arnoldi method is the best if no preconditioning would be used. For symmetric matrices, if there is only one initial guess, the Lanczos algorithm should be preferred.

The five methods behave differently when more than one initial guesses are provided or when finite precision arithmetic is used. Among the five methods compared, the Davidson method is the most stable one because it has the least amount of cancelation error when building it basis. If fact, the new vector added to the basis is orthogonal to the basis without preconditioning. Because of this, there are cases where the Davidson method is preferred even though it is more expensive per step. For example, if the wanted smallest eigenvalues have significantly smaller separation than the largest eigenvalues, based on the experiences with the Lanczos method, we know that the largest eigenvalues will converge quickly and cause orthogonality problem in the basis. The same orthogonality problem can occur in the Arnoldi method though it is less likely than in the Lanczos method, but the more stable Davidson method will be able to find the solution without encounter the same difficulty. The same is true if there are interior eigenvalues that converge very easily, as in the case of BCSSTK01.

With preconditioning, the five methods behave very differently from each other. The Davidson preconditioning is regarded as an inexact-Newton method which means that it could work well when the preconditioner is good. A theoretical study of preconditioned eigenvalue algorithms has not been conducted. We took the approach of comparing the methods through numerical experiment. Based on our experimental results, the Davidson method is by far the most effective one with all tested preconditioners.

Improving preconditioning scheme. The preconditioning scheme of the Davidson method approximately solves a nearly singular equation, $(A - \lambda I)y = r$, where r is the current residual vector, and λ is the current approximate eigenvalue. In the Rayleigh quotient iteration method for eigenvalue problems, inversion of the same matrix was also required. From the study of Rayleigh quotient iteration, researchers have noticed many problems when attempting to invert the matrix $(A - \lambda I)$ [36]. However, our experiment shows that the Davidson method is still very effective. Based on this observation, we believe that the inexact Newton approach is a sound basic preconditioning strategy though this particular Jacobian matrix may have some undesirable properties.

We studied three different Newton iterations for eigenvalue problems: the augmented Newton method of Peters and Wilkinson [98], the constrained Newton method based on the framework presented in [142], and the Jacobi-Davidson recurrence based on the Jacobi-Davidson preconditioning scheme [125]. The augmented Newton scheme produces a Jacobian matrix that is larger than the size of the original matrix A. In attempting to reduce the size of this linear system, we realized that there are a number of variations which are equivalent to each other but have different Jacobian matrices. From this analysis, we see that the Olsen preconditioning scheme can be regarded as a Schur form of the augmented Newton scheme.

The Jacobian matrices of the different variations of Newton method for eigenvalue problems can be used as preconditioners in three ways.

- Develop specialized incomplete factorization scheme for the Jacobian matrix of the augmented Newton scheme.
- Olsen preconditioning scheme which can be viewed as performing a regular preconditioner $M \approx (A-\lambda I)$ twice to improve the quality of the approximate solution.
- There are three Newton schemes whose Jacobian matrices can not be easily factorized. In this case, a Krylov iterative solver or an approximate inverse can be used to approximately solve the preconditioning equation.

We have performed experiment on most of the techniques mentioned above. For the limited number of test cases studied, preconditioners based on Krylov iterative solvers are easy to use and fairly effective. There are four different linear systems that could be used with Krylov iterative solvers. They share the same right-hand side, but have different iteration matrices. The four matrices are: $J_R = (A-\lambda I)$, $J_I = (A-\lambda I+xx^T)$, $J_C = (A - \lambda I - xx^T(A + A^T))$, and $J_J = (I - xx^T)(A - \lambda I)(I - xx^T)$. Each of these four different linear systems seems to have its own advantage. The condition number of J_R is very small when λ is far from any exact eigenvalue. J_R could potentially run into problems if λ is very close to an accurate solution. Since during most of the iterations, λ is far from an accurate solution, the Davidson preconditioning scheme is effective. It is possible for J_C to have large condition number when λ is far from convergence even though it is shown to be well behaved near convergence. The matrix J_J is designed to be singular which means a Krylov method could find a solution to the linear system, but a different kind of linear system solver may fail to do so. In addition, it is generally harder to construct a preconditioner for a singular linear system. Overall, using J_I is the most prudent choice. In our tests, the condition number of J_I is almost as small as that of J_R when λ is far from any exact eigenvalue, and the condition number of J_I does not grow without a bound like that of J_R when λ converges toward a simple eigenvalue.

Using polynomials. Using polynomials for eigenvalue problems has a few advantages: they are fairly easy to add to an projection based eigenvalue routine; they can improve the scalability of the eigenvalue routine on parallel machines by reducing the number of dot-products needed. The main obstacle which prevents the effective use of the polynomial schemes is that the user has to determine a certain number of parameters in order to use them. In this thesis, we studied 14 different choices of selecting parameters for 5 different polynomials. We tested three different strategies of using these 5 polynomials. Overall, we saw that using polynomials with the Davidson method can reduce the total number of matrix-vector multiplications significantly compared with the unpreconditioned Davidson method. The performances of all three schemes are very sensitive to the choice of the parameters. Among the three different ways of using the polynomials, the purification scheme is the most successful one on the two test problems. The most successful parameter selection schemes are different for the three different ways of using polynomials.

The most successful scheme for extreme eigenvalue problem is a hybrid method that uses the three highest degree least-squares polynomials computed on the interval $[\lambda_2, \lambda_m]$. For interior eigenvalue problems, the hybrid schemes with the two-interval least-squares polynomial and the Kernel polynomial use almost the same amount of time in the best cases.

Since Krylov iterative solvers also build solutions in terms of polynomials of A, we compared the results of using them as preconditioners to other polynomial schemes. The particular iterative method used is the Conjugate Gradient (CG) method. The CG preconditioner is much more effective on the problems tested than the polynomial preconditioning schemes. In terms execution time, the best results from using polynomials purification scheme can be comparable to the average case of the CG preconditioning.

In conclusion, we identified the Davidson method to be the most effective choice for finding multiple eigenpairs of large sparse matrices. We have found different preconditioning schemes that can avoid many pitfalls of the Davidson preconditioning scheme. Among the preconditioners which use matrix-vector multiplications, our tests show that Krylov methods are effective and easy to use.

6.2 Future work

Overall we see preconditioned methods as an effective approach to solve eigenvalue problems. We would continue to pursue the study of new preconditioning schemes and new preconditioners for eigenvalue problems to further enhance the performance of eigenvalue methods. In the immediate future, we see two different directions.

The first is to build a high quality eigenvalue software. During this study, we have developed a significant number of codes to prove the concept and to conduct tests. One way of extending this research is to build a coherent software package out of the existing codes. In building this package, a few issues need to be addressed to ensure stable performance of the software. For example, a more effective use of the large eigenvector array, a more effective thick restarting scheme, a better dynamic scheme for switching between the Davidson iteration and the polynomial methods are needed. We would like also to use this eigenvalue code in a variety of applications. Through the use of PVM and MPI data communication libraries, we have been able to easily port the eigenvalue code onto parallel environments. In constructing this software package, we should continue to improve the structure of the program so that it requires little or no additional work to run the eigenvalue packages efficiently on most parallel architectures.

The second front of research would be to extend the preconditioning techniques to other types of eigenvalue problem. For example, most preconditioning techniques can be trivially extended to Hermitian eigenvalue problems and unsymmetric eigenvalue problems. Through our study here, we see that one of the most robust preconditioning schemes is the CG-based preconditioner. It has been noticed that by using a dynamic tolerance on the residual of the linear system, along with a dynamic limit on the number of matrix-vector multiplications used by CG, one could significantly reduce the execution time of the overall eigenvalue solution process. The search for this dynamic tolerance scheme could be conducted through study of underlying dynamics.

There are also many questions we have raised during the study of hybrid methods for eigenvalue problem. They are interesting challenges as well.

Appendix A

Acknowledgement

I would like to express my sincere gratitude to my research adviser Professor Yousef Saad for his support and guidance through out my PhD study. Many thanks to Edmond Chow, Dr. Andreas Stathopoulos, Dr. Xiaodun Jin, and Dr. Andrew Anda, for their friendship and for their constant inspiration in life and in research. My graduate school life would not be the same without them.

I would also like to thank all the professors who have taught me and given me advice and suggestions through out my study at the University of Minnesota. Due to their effort, the past several years have been very fruitful time for me.

Bibliography

[1] M. Alfaro, J. S. Dehesa, F. J. marcellan, J. L. Rubio de Francia, and J. Vinuesa, editors. *Orthogonal Polynomials and Their Applications*. Spriger-Verlag, Berlin, Germany, 1988. Proceedings of an International Symposium held in Segovia, Spain, Sept. 22-27, 1986.

[2] E. L. Allgower and K. Georg. *Numerical Continuation Methods - An Introduction*. Spriger-Verlag, Berlin, Germany, 1990.

[3] E. L. Allgower and K. Georg. Continuation and path following. *Acta Numerica*, pages 1–64, 1993.

[4] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Du Croz, A. Greenbaum, S. Hammarling, A McKenney, S. Ostrouchov, and D. Sorensen. *LAPACK User's Guide*. SIAM, Philadelphia, PA, 1992.

[5] W. E. Arnoldi. The principle of minimized iteration in the solution of the matrix eigenvalue problem. *Quart. Appl. Math.*, 9:17–29, 1951.

[6] S. F. Ashby, T. A. Manteuffel, and J. S. Otto. A comparison of adaptive Chebyshev and least squares polynomial preconditioning for Hermitian positive linear systems. *SIAM J. Sci. Statist. Comput.*, 13:1–29, 1992.

[7] O. Axelsson and I. Gustafsson. Preconditioning and two-level multigrid methods for arbitrary degree of approximation. *Math. Comput.*, 40:219–242, 1983.

[8] O. Axelsson and P. S. Vassilevski. A survey of multilevel preconditioned iterative methods. *BIT*, 29(4), 1989.

[9] Owe Axelsson. A survey of preconditioned iterative methods for linear systems of equations. *BIT*, 25:166–187, 1985.

[10] Owe Axelsson. *Iterative Solution Methods*. Press Syndicate of the University of Cambridge, Cambridge, UK, 1994.

[11] Z. Bai, D. Day, and Q. Ye. ABLE: an adaptive block Lanczos method for non-Hermitian eigenvalue problems. Technical Report 95-04, Univeristy of Kentucky, Department of Mathematics, 1995.

[12] Z. Bai and G. W. Stewart. SRRIT — a FORTRAN subroutine to calculate the dominant invariant subspace of a nonsymmetric matrix. Technical Report UMIACS TR-92-61, University of Maryland at College Park, College Park, MD, 1992.

[13] Zhaojun Bai. Progress in the numerical solution of the nonsymmetric eigenvalue problem. *Numerical Linear Algebra with Applications*, 2:219–234, 1995.

[14] J. G. L. Booten, H. A. van der Vorst, P. M. Meijer, and H. J. J. te Riele. A preconditioned Jacobi-Davidson method for solving large generalized eigenvalue problems. Technical Report NM-R9414, Mathematisch Centrum, Centrum voor Wiskunde en Informatica, Amsterdam, 1994.

[15] John P. Boyd. A Chebyshev polynomial interval-searching method ("lanczos economization") for solving a nonlinear equation with application to the nonlinear eigenvalue problem. *J. Comput. Phys.*, 118:1–8, 1995.

[16] J. H. Bramble, A. V. Knyazev, and J. E. Pasciak. A subspace preconditioning algorithm for eigenvector/eigenvalue computation. Technical Report 66, Center for Computational Mathematics, University of Colorado at Denver, 1995.

[17] C. Brezinski and M. Redivo-Zaglia. Hybrid procedures for solving linear systems. *Numer. Math.*, 67(1):1–19, 1994.

[18] A. Chapman and Y. Saad. Deflated and augmented Krylov subspace techniques. Technical Report UMSI 95/181, Minnesota Supercomputing Institute, Univ. of Minnesota, 1995.

[19] F. Chatelin. *Eigenvalues of Matrices*. Wiley, 1993.

[20] J. R. Chelikowsky, N. Troullier, and Y. Saad. Finite-difference-pseudopotential method: electronic structure calculations without a basis. *Phys. Rev. Lett.*, 72:1240–3, 1994.

[21] J. R. Chelikowsky, N. Troullier, K. Wu, and Y. Saad. Higher order finite difference pseudopotential method: an application to diatomic molecules. *Phys. Rev. B*, 50:11355–11364, 1994.

[22] J. R. Chelikowsky, N. R. Troullier, X. Jing, D. Dean, N. Binggeli, K. Wu, and Y. Saad. Algorithms for the structural properties of clusters. *Computer Physics Communications*, 85:325–335, 1995.

[23] T. S. Chihara, editor. *An Introduction to Orthogonal Polynomials*. Gordon and Breach Science Publishers, New York, London, Paris, 1978.

[24] E. Chow and Y. Saad. Approximate inverse preconditioners via sparse-sparse iterations. *SIAM J. Sci. Comput.*, To appear, 1996.

[25] M. L. Cohen and J. R. Chelikowsky. *Electronic Structure and Optical Properties of Semiconditors*. Springer-Verlag, New York, Berlin, Heidelberg, 2nd edition, 1989.

[26] M. Crouzeix, B. Philippe, and M. Sadkane. The Davidson method. *SIAM J. Sci. Comput.*, 15:62–76, 1994.

[27] J. Cullum and R. A. Willoughby. *Lanczos Algorithms for Large Symmetric Eigenvalue Computations: Theory*, volume 1 of *Progress in Scientific Computing; v. 3*. Birkhauser, Boston, 1985.

[28] J. Cullum and R. A. Willoughby. *Lanczos Algorithms for Large Symmetric Eigenvalue Computations: Programs*, volume 2 of *Progress in Scientific Computing; v. 4*. Birkhauser, Boston, 1985.

[29] J. W. Daniel, W. B. Gragg, L. Kaufman, and G. W. Stewart. Reorthogonalization and stable algorithms for updating the Gram-Schmidt QR factorization. *Math. Comp.*, 30(136):772–795, October 1976.

[30] Ernest R. Davidson. The iterative calculation of a few of the lowest eigenvalues and corresponding eigenvectors of large real-symmetric matrices. *J. Comput. Phys.*, 17:87–94, 1975.

[31] Ernest R. Davidson. Super-matrix methods. *Computer Physics Communications*, 53:49–60, 1989.

[32] Ernest R. Davidson. Monster matrices: their eigenvalues and eigenvectors. *Computers in Physics*, 7(5):519–522, 1993.

[33] Philip J. Davis. *Interpolation and Approximation*. Dover Publications, New York, 1975.

[34] J. J. Dongarra. Improving the accuracy of computed singular values. *SIAM J. Sci. Statist. Comput.*, 4:712–719, 1983.

[35] J. J. Dongarra, S. Hammarling, and J. H. Wilkinson. Numerical considerations in computing invariant subspaces. *SIAM J. Matrix Anal. Appl.*, 13(1):145–161, 1992.

[36] J. J. Dongarra, C. B. Moler, and J. H. Wilkingson. Improving the accuracy of computed eigenvalues and eigenvectors. *SIAM J. Numer. Anal.*, 20:23–45, 1983.

[37] I. S. Duff, R. G. Grimes, and J. G. Lewis. Sparse matrix test problems. *ACM Trans. Math. Soft.*, pages 1–14, 1989.

[38] A. Edelman, T. Arias, and S. T. Smith. Conjugate Gradient and Newton's method on the Grassmann and Stiefel manifolds, 1996. Submitted to SIAM J. Matrix Anal. Appl.

[39] B. Fischer and R. W. Freund. On adaptive weighted polynomial preconditioning for Hermitian positive matrices. *SIAM J. Sci. Comput.*, 15(2):408–426, March 1994.

[40] C. Froese Fischer. Multitasking the Davidson algorithm for the large, sparse eigenvalue problem. *International Journal of Supercomputer Applications*, 3:28–53, 1990.

[41] D. R. Fokkema, G. L. G. Sleijpen, and H. A. van der Vorst. Jacobi-Davidson style QR and QZ algorithms for the partial reduction of matrix pencils. Technical Report nr. 941, Department of mathematics, Universiteit Utrecht, 1996.

[42] B. Fornberg and D. M. Sloan. A review of pseudospectral methods for solving partial differential equations. *Acta Numerica*, pages 203–267, 1994.

[43] Bengt Fornberg. High-order finite differences and the pseudospectral method on staggered grids. *SIAM J. Numer. Anal.*, 27:904–918, 1990.

[44] Bengt Fornberg. Rapid generation of weights in finite difference formulas. In *Numerical analysis 1989 (Dundee, 1989)*, pages 105–121. Longman Sci. Tech., Harlow, 1990.

[45] Roland W. Freund. Qusai-Kernel polynomials and convergence results for qusai-minimal residual iterations. In Braess and Schumack, editors, *Numerical Methods of Approximation Theory*, pages 77–95. Birkhauser, Basel, 1992.

[46] Roland W. Freund. Qusai-Kernel polynomials and their use in non-Hermitian matrix iterations. *Journal of Computational and Applied Mathematics*, 34:135–158, 1992.

[47] Albert T. Galick. *Efficient solution of large sparse eigenvalue problems in microelectronic simulation.* PhD thesis, University of Illinois at Urbana-Champaign, 1993.

[48] A. George and J. W. H. Liu. *Computer Solution of Large Sparse Positive-Definite Systems.* Prentice-Hall, New Jersey, 1981.

[49] G. H. Golub, editor. *Fast Poisson solvers*, pages 319–370. Math. Assoc. America, Washington, DC, 1984.

[50] G. H. Golub and D. P. O'Leary. Some history of the Conjugate Gradient and Lanczos algorithms: 1948-1976. *SIAM Review*, 31:50–102, 1989.

[51] G. H. Golub and C. F. van Loan. *Matrix Computations.* The John Hopkins University Press, Baltimore, MD 21211, 1989.

[52] B. S. Grabow, J. M. Boyle, J. J. Dongarra, and C. B. Moler, editors. *Matrix Eigensystem Routines — EISPACK Guide Extension.* Number 51 in Lecture Notes in Computer Science. Springer, New York, 2nd edition, 1977.

[53] R. G. Grimes, J. G. Lewis, and H. D. Simon. A shifted block Lanczos algorithm for solving sparse symmetric generalized eigenproblems. *SIAM J. Matrix Anal. Appl.*, 15(1):228–272, 1994.

[54] Martin H. Gutknecht. A completed theory of the unsymmetric Lanczos process and related algorithms, part I. *SIAM J. Matrix Anal. Appl.*, 13(2):594–639, 1992.

[55] Martin H. Gutknecht. A completed theory of the unsymmetric Lanczos process and related algorithms, part II. *SIAM J. Matrix Anal. Appl.*, 15(1):15–58, 1994.

[56] B. Harrod, A. Sameh, and W. J. Harrod. Trace minimization algorithm for the generalized eigenvalue problems. In Garry Rodrigue, editor, *Parallel Processing for Scientific Computing (Los Angeles, CA, 1987)*, pages 77–81, Philadelphia, PA, 1989. SIAM.

[57] Michael A. Heroux. A reverse communication interface for "matrix-free" preconditioned iterative solvers. In C. A. Brebbia, D. Howard, and A. Peters, editors, *Applications of Supercomputers in Engineering II*, pages 207–213, Boston, 1991. Computational Mechanics Publications.

[58] X. Jing, N. R. Troullier, J. R. Chelikowsky, K. Wu, and Y. Saad. Vibrational modes of silicon nanostructures. *Solid State Communication*, 96(4):231–235, 1995.

[59] X. Jing, N. R. Troullier, D. Dean, N. Binggeli, J. R. Chelikowsky, K. Wu, and Y. Saad. *Ab Initio* molecular dynamics simulations of Si clusters using the higher-order finite-difference-pseudopotential method. *Phys. Rev. B*, 50:12234–12237, 1994.

[60] Xiaodun Jing. *Theory of the electronic and structural properties of semiconductor clusters.* PhD thesis, University of Minnesota, Minneapolis, Minnesota, 1995.

[61] Claes Johnson. *Numerical Solution of Partial Differential Equations by Finite Eqlement Method.* Cambridge University Press, Cambridge; New York, 1987.

[62] Wayne Joubert. A robust GMRES-based adaptive polynomial preconditioning algorithm for nonsymmetric linear systems. *SIAM J. Sci. Comput.*, 15(2):427–439, March 1994.

[63] Charles Kittel. *Quantum Theory of Solids.* John Wiley & Sons, New York, 1963.

[64] Charles Kittel. *Introduction to Solid State Physics.* John Wiley & Sons, New York, 1986.

[65] A. V. Knyazev and A. L. Skorokhodov. Preconditioned gradient-type iterative methods in a subspace for partial generalized symmetric eigenvalue problems. *SIAM J. Numer. Anal.*, 31:1226–1239, 1994.

[66] L. Yu. Kolotilina and A. Yu. Yeremin. Factorized sparse approximate inverse preconditionings I. theory. *SIAM J. Matrix Anal. Appl.*, 14(1):45–58, 1993.

[67] J. D. Kress, S. B. Woodruff, G. A. Parker, and R. T. Pack. Some strategies for enhancing the performance of the block Lanczos method. *Computer Physics Communications*, 53:109–115, 1989.

[68] V. Kumar, A. Grama, A. Gupta, and G. Karypis. *Introduction to Parallel Computing: design and analysis of algorithms.* The Benjamin/Cummings Publishing Company, Redwood City, California, 1994.

[69] C. Lanczos. An iterative method for the solution of the eigenvalue problem of linear differential and integral operators. *Journal of Research of the National Bureau of Standards*, 45:255–282, 1950.

[70] R. B. Lehoucq. Restart an Arnoldi reduction. Technical Report MCS-P591-0496, Argonne National Laboratory, Argonne, IL, 1996. Submitted to SIAM Journal on Scientific Computing.

[71] R. B. Lehoucq and K. Meerbergen. The inexact rational Krylov sequence method. Technical Report MCS-P612-1096, Argonne National Laboratory, Argonne, IL, 1997.

[72] R. B. Lehoucq and D. Sorensen. Deflation techniques for an implicitly restarted Arnoldi iteration. Technical Report TR94-13, Department of Computational and Applied Mathematics, Rice University, Houston, Texas, 1994. Accepted for publication in SIAM Journal on Matrix Analysis and Applications.

[73] Richard B. Lehoucq. *Analysis and implementation of an implicitly restarted Arnoldi iteration.* PhD thesis, Rice University, 1995.

[74] G. C. Lo and Y. Saad. Iterative solution of general sparse linear systems on clusters of workstations. Technical Report UMSI 96-117, Minnesota Supercomputer Institute, University of Minnesota, Minneapolis, MN, 1996.

[75] Karl Meerbergen. *Robust methods for the calculation of rightmost eigenvalues of nonsymmetric eigenvalue problems.* PhD thesis, K. U. Leuven, Heverlee, Belgium, 1996.

[76] J. A. Meijerink and H. A. van der Vorst. An iterative solution method for lineat systems of which the coefficient matrix is a symmetric M-matrix. *Math. Comp.*, 31:148–162, 1977.

[77] R. B. Morgan. Davidson's method and preconditioning for generalized eigenvalue problems. *J. Comput. Phys.*, 89:241–245, 1990.

[78] R. B. Morgan. Computing interior eigenvalues of large matrices. *Lin. Alg. Appl.*, pages 289–309, 1991.

[79] R. B. Morgan. Generalizations of Davidson's method for computing eigenvalue of large nonsymmetric matrices. *J. Comput. Phys.*, 101:289–291, August 1992.

[80] R. B. Morgan and D. S. Scott. Generalizations of Davidson's method for computing eigenvalues of sparse symmetric matrices. *SIAM J. Sci. Statist. Comput.*, 7:817–825, 1986.

[81] R. B. Morgan and D. S. Scott. Preconditioning the Lanczos algorithm for sparse symmetric eigenvalue problems. *SIAM J. Sci. Comput.*, 14:585–593, 1993.

[82] R. B. Morgan and M. Zeng. Estimates for interior eigenvalues of large nonsymmetric matrices, 1996.

[83] Ronald B. Morgan. On restarting the Arnoldi method for large nonsymmetric eigenvalue problems, 1996.

[84] K. W. Morton and D. F. Mayers. *Numerical Solution of Partial Differential Equations.* Cambridge University Press, Cambridge; New York, 1994.

[85] S. G. Mulyarchik, S. S. Bielawski, and A. V. Popov. Efficient computational schemes of the Conjugate Gradient method for solving linear systems. *J. Comput. Phys.*, 110:210–211, 1994.

[86] C. W. Murray, S. C. Racine, and E. R. Davidson. Improved algorithms for the lowest eigenvalues and associated eigenvectors of large matrices. *J. Comput. Phys.*, 103(2):382–389, 1992.

[87] R. Natarajan and D. Vanderbilt. A new iterative scheme for obtaining eigenvectors of large, real-symmetric matrices. *J. Comput. Phys.*, 82:218–228, 1989.

[88] Roy L. Nersesian. *Computer simulation in financial risk management : a guide for business planners and strategists.* Quorum Books, New York, 1991.

[89] Michael Oettli. *The Homotopy Method Applied to the Symmetric Eigenproblem.* PhD thesis, Swiss Federal Institute of Technology Zurich, Zurich, Switzerland, 1995.

[90] J. Olsen, P. Jorgensen, and J. Simons. Passing the one-billion limit in full configuration- interaction (FCI) calculations. *Chemical Physics Letters*, 169:463–472, 1990.

[91] C. C. Paige, B. N. Parlett, and H. A. van der Vorst. Approximate solutions and eigenvalue bounds for Krylov subspaces. *Numerical Linear Algebra with Applications*, 2:115–134, 1995.

[92] G. A. Parker, W. Zhu, Y. Huang, D. K. Hoffman, and D. J. Kouri. Matrix pseudo-spectroscopy: iterative calculation of matrix eigenvalues and eigenvectors of large matrices using a polynomial expansion of the Dirac delta function. *Computer Physics Communications*, 96(1):27–35, 1996.

[93] B. N. Parlett. The Rayleigh quotient iteration and some generalizations for nonnormal matrices. *Math. Comp.*, 28:679–693, 1974.

[94] B. N. Parlett. Two monitoring schemes for the Lanczos algorithm. In R. Glowinski and J. L. Lions, editors, *Computing Methods in Applied Science and Engineering, V*, pages 27–34. North-Holland Publishing Company, 1982.

[95] Beresford N. Parlett. *The symmetric eigenvalue problem.* Prentice-Hall, Englewood Cliffs, NJ, 1980.

[96] Beresford N. Parlett. Towards a black box Lanczos program. *Computer Physics Communications*, 53:169–179, 1989.

[97] Kamran Parsaye and Mark Chignell. *Intelligent database tools & applications : hyperinformation access, data quality, visualization, automatic discovery*. Wiley, New York, 1993.

[98] G. Peters and J. H. Wilkinson. Inverse iteration, ill-conditioned equations and Newton's method. *SIAM Review*, 21(3):339–360, July 1979.

[99] S. Petiton, Y. Saad, K. Wu, and W. Ferng. Basic sparse matrix computations on the CM-5. *International Journal of Modern Physics C*, 4(1), 1993.

[100] Philip Pfaff. *Financial modeling*. Allyn and Bacon, Needham Heights, Mass., 1990.

[101] Theodore J. Rivlin. *Chebyshev Polynomials*. Wiley, New York, 2nd edition, 1990.

[102] A. Ruhe. Implementation aspects of band Lanczos algorithms for computation of eigenvalues of large sparse symmetric matrices. *Math. Comput.*, 33:680–687, 1979.

[103] A. Ruhe. Rational Krylov sequence methods for eigenvalue computation. *Lin. Alg. Appl.*, 58:391–405, 1984.

[104] Heinz Ruthishauser. Computational aspects of F. L. Bauer's simultaneous iteration method. *Numer. Math.*, 13:4–13, 1969.

[105] Heinz Ruthishauser. Simultaneous iteration method for symmetric matrices. *Numer. Math.*, 16:205–223, 1970.

[106] Y. Saad. Analysis of augmented Krylov subspace techniques. Technical Report UMSI 95/175, Minnesota Supercomputing Institute, Univ. of Minnesota, 1995.

[107] Y. Saad and K. Wu. Design of an iterative solution module for a parallel matrix library (P_SPARSLIB). *Applied Numerical Mathematics*, 19:343–357, 1995. Was TR 94-59, Department of Computer Science, University of Minnesota.

[108] Y. Saad, K. Wu, and S. Petiton. Sparse matrix computations on the CM-5. In R. Sincovec, D. Keyes, M. Leuze, L. Petzold, and D. Reed, editors, *Proceedings of Sixth SIAM Conference on Parallel Processing for Scientific Computing*, pages 414–420, Philadelphia, 1993. SIAM.

[109] Yousef Saad. On the rate of convergence of the Lanczos and the block-Lanczos methods. *SIAM J. Numer. Anal.*, 17:687–706, 1980.

[110] Yousef Saad. Variations on Arnoldi's method for computing eigenelements of large unsymmetric matrices. *Lin. Alg. Appl.*, 34:323–346, 1980.

[111] Yousef Saad. Iterative solution of indefinite symmetric linear systems by methods using orthogonal polynomials over two disjoint intervals. *SIAM J. Numer. Anal.*, 20(4):784–811, 1983.

[112] Yousef Saad. Chebyshev acceleration techniques for solving nonsymmetric eigenvalue problems. *Math. Comp.*, 42:567–588, 1984.

[113] Yousef Saad. Practical use of polynomial preconditioning for the Conjugate Gradient method. *SIAM J. Sci. Statist. Comput.*, 6:865–881, 1985.

[114] Yousef Saad. Preconditioning techniques for nonsymmetric and indefinite linear systems. *Journal of Computational and Applied Mathematics*, 24:89–105, 1988.

[115] Yousef Saad. SPARSKIT: A basic toolkit for sparse matrix computations. Technical Report 90-20, Research Institute for Advanced Computer Science, NASA Ames Research Center, Moffet Field, CA, 1990. Software currently available at <ftp://ftp.cs.umn.edu/dept/sparse/>.

[116] Yousef Saad. *Numerical Methods for Large Eigenvalue Problems*. Manchester University Press, 1993.

[117] Yousef Saad. Highly parallel preconditioners for general sparse matrices. In G. Golub, A. Greenbaum, and M. Luskin, editors, *Recent advances in Iterative Methods*, pages 165–199. Springer-Verlag, 1994.

[118] Yousef Saad. ILUT: a dual threshold incomplete ILU factorization. *Numerical Linear Algebra with Applications*, 1:387–402, 1994. Technical Report 92-38, Minnesota Supercomputer Institute, University of Minnesota, 1992.

[119] Yousef Saad. *Iterative Methods for Sparse Linear Systems*. PWS publishing, Boston, MA, 1996.

[120] Miloud Sadkane. Block-Arnoldi and Davidson methods for unsymmetric large eigenvalue problems. *Numer. Math.*, 64(2):195–211, 1993.

[121] Miloud Sadkane. A block Arnoldi-Chebyshev method for computing the leading eigenpairs of large sparse unsymmetric matrices. *Numer. Math.*, 64(2):181–193, 1993.

[122] A. H. Sameh and J. A. Wisniewski. A trace minimization algorithm for the generalized eigenvalues problem. *SIAM J. Numer. Anal.*, 19(6):1243–1259, 1982.

[123] V. Simoncini and E. Gallopoulos. A hybrid block GMRES method for nonsymmetric systems with multiple right-hand sides. Technical Report 1378, Center for Superconputing Research and Development, University of Illinois at Urbana-Champaign, Urbana, IL, 1994.

[124] G. L. G. Sleijpen, A. G. L. Booten, D. R. Fokkema, and H. A. van der Vorst. Jacobi-Davidson type methods for generalized eigenproblems and polynomial eigenproblems: Part I. Technical Report nr. 923, Department of mathematics, Universiteit Utrecht, 1995.

[125] 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(2), 1996.

[126] G. L. G. Sleijpen and H. A. van der Vorst. The Jacobi-Davidson method for eigenvalue problems and its relation with accelerated inexact newton schemes, 1996.

[127] B. T. Smith, J. M. Boyle, B. S. Garbow, Y. Ikebe, V. C. Klema, and C. B. Moler, editors. *Matrix eigensystem Routines — EISPACK Guide.* Number 6 in Lecture Notes in Computer Science. Springer, New York, 2nd edition, 1976.

[128] Gordon D. Smith. *Numerical Solution of Partial Differential Equations: Finite Difference Methods.* Oxford University Press, 1985.

[129] D. Sorensen, R. Lehoucq, P. Vu, and C. Yang. ARPACK: an implementation of the Implicitly Restarted Arnoldi iteration that computes some of the eigenvalues and eigenvectors of a large sparse matrix. Available from ftp.caam.rice.edu under the directory pub/people/sorensen/ARPACK, 1995.

[130] D. C. Sorensen and C. Yang. A truncated RQ-iteration for large scale eigenvalue calculations. Technical Report TR96-06, Department of Computational and Applied Mathematics, Rice University, Houston, Texas, 1996.

[131] D. S. Sorensen. Implicit application of polynomial filters in a K-step Arnoldi method. *SIAM J. Matrix Anal. Appl.*, 13(1):357–385, 1992.

[132] G. Starke and R. S. Varga. A hybrid Arnoldi-Faber iterative method for nonsymmetric systems of linear equations. *Numer. Math.*, 64(2):213–240, 1993.

[133] A. Stathopoulos, Y. Saad, and C. F. Fischer. Robust preconditioning of large, sparse, symmetric eigenvalue problems. *Journal of Computational and Applied Mathematics*, 64:197–215, 1995.

[134] A. Stathopoulos, Y. Saad, and K. Wu. Dynamic thick restarting of the davidson and the implicitly restarted arnoldi methods. Technical Report UMSI 96/123, University of Minnesota Supercomputer Institute, 1996.

[135] A. Stathopoulos, Y. Saad, and K. Wu. Thick restarting of the Davidson method: an extension to implicit restarting. In T. Manteuffel, S. McCormick, L. Adams, S. Ashby, H. Elman, R. Freund, A. Greenbaum, S. Parter, P. Saylor, N. Trefethen, H. van der Vorst, H. Walker, and O. Wildlund, editors, *Proceedings of Copper Mountain Conference on Iterative Methods*, Copper Mountain, Colorado, 1996.

[136] Andreas Stathopoulos and Charlotte Fischer. Reducing synchronization on the parallel Davidson method for the large, sparse, eigenvalue problem. In *Supercomputing '93*, pages 172–180, Los Alamitos, CA, 1993. IEEE Comput. Soc. Press.

[137] Andreas Stathopoulos and Charlotte Fischer. A Davidson program for finding a few selected extreme eigenpairs of a large, sparse, real, symmetric matrix. *Computer Physics Communications*, 79(2):268–290, 1994.

[138] Eduard L. Stiefel. *Kernel Polynomials in Linear Algebra and Their Numerical Applications.* National Bureau of Standards, Washington, D. C., 1955.

[139] J. Stoer and R. Bulirsch. *Introduction to Numerical Analysis.* Springer-Verlag, New York, 1980.

[140] Bjarne Stroustrup. *The C++ Programming Language.* Addison-Wesley Publishing, 2nd edition, 1991.

[141] Gabor Szegö. *Orthogonal Polynomials.* American Mathematical Society, Providence, Rhode Island, 4th edition, 1975.

[142] R. A. Tapia. Newton's method for optimization problems with equality constraints. *SIAM J. Numer. Anal.*, 11:874–886, 1974.

[143] R. A. Tapia. Newton's method for problems with equality constraints. *SIAM J. Numer. Anal.*, 11:174–196, 1974.

[144] R. A. Tapia. Diagonalized multiplier methods and quasi-Newton methods for constrained optimization. *Journal of Optimization Theory and Applications*, 22:135–194, 1977.

[145] James. W. Thomas. *Numerical Partial Differential Equations*. Springer, New York, 1995.

[146] Charles H. Tong. *Parallel preconditioned Conjugate Gradient methods for elliptic partial differential equations*. PhD thesis, University of California, Los Angeles, CA, 1990.

[147] N. Troullier, J. R. Chelikowsky, and Y. Saad. Calculating large systems with plane waves: is it a N^3 or N^2 scaling problem? *Solid State Communications*, 93:225–230, 1995.

[148] N. R. Troullier and J. L. Martins. Efficient pseudopotentials for plan-wave calculations. *Phys. Rev. B*, 43:1993–1997, 1991.

[149] M. B. van Gijzen. A polynomial preconditioner for the GMRES algorithm. *Journal of Computational and Applied Mathematics*, 59:91–107, 1995.

[150] J. H. van Lenthe and P. Pulay. A space-saving modification of Davidson's eigenvector algorithm. *J. Comput. Chem.*, 11:1164–1168, 1990.

[151] A. F. Voter, J. D. Kress, and R. N. Silver. Linear-scaling tight binding from a truncated-moment approach. *Phys. Rev. B*, 53(19):12733–12741, 1996.

[152] T. Washio and K. Hayami. Parallel block preconditionering based on SSOR and MILU. *Numerical Linear Algebra with Applications*, 1:533–553, 1994.

[153] D. S. Watkins and L. Elsner. Convergence of algorithms of decomposition type for the eigenvalue problems. *Lin. Alg. Appl.*, 143:19–47, 1991.

[154] J. H. Wilkinson. *Algebraic Eigenvalue Problem*. Monographs on Numerical Analysis. Oxford Science Publications, New York, 1965.

[155] K. Wu, Y. Saad, and A. Stathopoulos. Preconditioned Krylov subspace methods for eigenvalue problems. In T. Manteuffel, S. McCormick, L. Adams, S. Ashby, H. Elman, R. Freund, A. Greenbaum, S. Parter, P. Saylor, N. Trefethen, H. van der Vorst, H. Walker, and O. Wildlund, editors, *Proceedings of Copper Mountain Conference on Iterative Methods*, Copper Mountain, Colorado, 1996.