LARGE SCALE STABILIZATION WITH LINEAR FEEDBACK

Name: Xiang Rao
Department: Department of Computer Science
Major Professor: Kyle A. Gallivan
Degree: Master of Science
Term Degree Awarded: Fall, 1999

This thesis analyzes the stabilization of large-scale dynamical systems via linear feedback in discrete- and continuous-time state space form. We survey the state-of-the-art for such problems and propose novel algorithms that are competitive with or superior to current algorithms. Three novel families of methods for discrete-time dynamical systems are developed based on the Discrete Riccati Difference Equation: SQR, CSQR, and Approximate Newton. For each family, particular methods are developed and analyzed with respect to efficiency and efficacy. Efficient versions of methods from the family of projection methods proposed by Saad are also developed and compared to our novel algorithms. For continuous-time systems we consider the transformation of the systems to discrete-time and the application of our methods and projection methods.

THE FLORIDA STATE UNIVERSITY

COLLEGE OF ARTS AND SCIENCES

LARGE SCALE STABILIZATION WITH LINEAR FEEDBACK

By

XIANG RAO

A Thesis submitted to the
Department of Computer Science
in partial fulfillment of the
requirements for the degree of
Master of Science

Degree Awarded:
Fall Semester, 1999

The members of the Committee approve the thesis of Xiang Rao defended on November 19 1999.

Kyle A. Gallivan
Professor Directing Thesis

Robert Van Engelen
Committee Member

Gordon Erlebacher
Committee Member

ACKNOWLEDGEMENTS

I want to thank Dr. Kyle Gallivan. He brought this research topic to me and has sponsored my two years study and research. Without his guiding, frequent discussions about the research and supervising, I could never complete this thesis. I particularly appreciate the help from Dr. Paul Van Dooren. During his three visits to Florida State University, he provided me countless precious ideas and research methods. Without his thoughts, I would still be in a maze and struggling to find a path. Special thanks to Dirac Library. I obtained there most of the references which helped this research. I also want to thank all my friends in Florida State University and Tallahasee for their long time help and support in these years.

This research was supported by the National Science Foundation under Grant No. CCR-9796315.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

This thesis analyzes the stabilization of large-scale dynamical systems via linear feedback in discrete- and continuous-time state space form. We survey the state-of-the-art for such problems and propose novel algorithms that are competitive with or superior to current algorithms. Three novel families of methods for discrete-time dynamical systems are developed based on the Discrete Riccati Difference Equation: SQR, CSQR, and Approximate Newton. For each family, particular methods are developed and analyzed with respect to efficiency and efficacy. Efficient versions of methods from the family of projection methods proposed by Saad are also developed and compared to our novel algorithms. For continuous-time systems we consider the transformation of the systems to discrete-time and the application of our methods and projection methods.

CHAPTER 1

INTRODUCTION

The topic of this thesis is the stabilization of discrete-time system

$$x_{i+1} = Ax_i + Bu_i, \tag{1.1}$$

where A and B are $n \times n$ and $n \times p$ real matrices which are known, x_i and u_i are vectors of dimension n and p respectively. The stabilization of the system requires the computation of a $p \times n$ feedback matrix F such that all eigenvalues of the so-called *closed loop matrix* $A - BF$ are inside unit circle and therefore the system defined by replacing A with $A - BF$ is stable. For small and moderate values of n, F can be computed via pole placement or the solution of a matrix equation, e.g., Riccati or Lyapunov equations [1],[2],[3],[4],[5],[6],[7],[8],[9].

The computational requirements for standard algorithms for these approaches, however, is prohibitive for large values of n. Fortunately, when n is large and $p << n$, the system matrix A and/or input matrix B are typically very sparse. Algorithms for such problems must therefore exploit this structure in order to efficiently compute a stabilizing feedback. In this thesis we survey the state-of-the-art for such problems and propose novel algorithms that are competitive with or superior to current algorithms.

Chapter 2 introduces the terminology and the basic theory governing the problem. The conditioning of the stabilization problem is also discussed in Chapter 2. Two metrics based on the matrix triple (A, B, F) are proposed to measure the conditioning of the stabilization problem and the efficacy of the stabilization algorithms. Two special forms of control systems are also analyzed in detail in order to enhance the understanding of the factors that influence conditioning.

A major contribution to solving large scale stabilization problems with a few unstable eigenvalues is Y. Saad's projection method ([10]) with which the stabilization or eigenvalue assignment is only imposed on a small size invariant subspace which contains the unstable invariant subspace of the system matrix A. Such an approach is often effective, but it can have convergence difficulties and the need for a basis of the invariant subspace can cause excess space requirements for very large systems. In this paper, we will discuss an improvement to the projection-based algorithm of Saad and we will develop efficient alternatives that address the convergence difficulties and avoid the need for an explicitly formed basis of the invariant subspace. The latter algorithms are created by exploiting the properties and free parameters in Riccati equations.

In Saad's projection algorithm ([10]), a left invariant subspace V of A (with presumably small dimension), that contains the left unstable invariant subspace of A is computed. There are two major classes of methods that can be used. The first computes the unstable eigenvalues and recovers their eigenvectors by some form of inverse iteration. The second class computes the basis directly by subspace iteration-like methods. The low-order projected system $(V'AV, V'B)$ is then stabilized and the reduced feedback F_u is lifted back to form a stabilizing feedback $F = F_uV'$ of the original system (A, B).

Methods in the first class benefit from years of sparse eigenvalue algorithm research but often require very high accuracy in the eigenvalues in order to produce the basis and hence result in more computation than necessary for stabilization.

Effective convergence is also one of the main issues of the second class of methods. The convergence of subspace iteration-like (SSI) methods which generate the sequence of approximations to the invariant subspace (1) starting from initial subspace V_0 and (2) updating V_i by extracting orthogonal basis of $A'V_{i-1}$ is usually consistent with the separation between desired eigenvalues and undesired eigenvalues in absolute value. In practice, it is often difficult to tune the parameters of such methods to converge even this quickly. They can be accelerated and some parameter sensitivity mitigated by the use of Stewart's SRR (Schur-Rayleigh-Ritz) refinement([11]). The acceleration is achieved by (1) enlarging the size of initial subspace V_0, (2) extracting the Schur vectors U_i corresponding to largest (or unstable) eigenvalues of $V_i' A' V_i$ and (3) combining $V_i U_i$ as the basis of the approximated invariant subspace.

A second source of difficulty for Saad's method is that it is, by definition, a two-phase process: find the basis then stabilize. Experience and empirical testing shows that a stabilizing feedback can often be found with approximations available long before the eigensolver would have any confidence in the basis of the unstable space. Simply computing a feedback on every iteration of the eigensolver is too expensive so a more seamless method of integrating the feedback computation with the update of the basis is needed. Finally, the major drawback with this approach is the need for a basis of the invariant subspace. Storage problems for large dynamical systems can result, therefore, it is worthwhile to look for methods that do not require the basis.

Chapter 4 discusses these projection-based approaches to stabilization. An effective version of SSI combined with SRR is proposed and analyzed. Particular attention is paid to the termination condition of the iteration including a technique that includes embedding a feedback computation of low dimension in the basic iteration.

The major results of this thesis concern a family of stabilization methods that addresses the concerns listed above, is competitive with Saad's method when Saad's method does well and is successful for many problems where Saad's method fails. Chapter 3 discusses the required theoretical foundations related to the discrete-time Riccati equation (DRE) and the discrete-time Riccati difference equation (DRDE)

$$P = A'(P - PB(R + B'PB)^{-1}B'P)A + Q \tag{1.2}$$

$$P_{i+1} = A'(P_i - P_iB(R + B'P_iB)^{-1}B'P_i)A + Q \tag{1.3}$$

where R and Q are $p \times p$ and $n \times n$ non-negative matrices and Q is usually decomposed into CC'. The most general results about DRE and DRDE convergence are given in [12]. It is shown there that under the condition of stabilizability of (A, B), there exists a stabilizable and non-negative solution P of DRE (1.2) and a stabilizing feedback F can be computed by $(R + B'PB)^{-1}B'PA$. Whether the solution of DRDE (1.3) converges to the stabilizing solution of DRE depends on properties of (A', C) or initial condition P_0.

For the purpose of stabilization, we have freedom to choose R, C and P_0. We have developed low-rank stabilization algorithms using various combinations of parameter settings for C (or Q) and P_0. Starting from non-negative P_0, P_i in the DRDE will keep the non-negative property. If Q is chosen as zero, the rank of P_i will be non-increasing. If $P_0 = 0$, $P_{i+1} - P_i$ will be non-negative and its rank will be non-increasing. The low-rank stabilization algorithms in this paper are based on these non-negative and non-increasing rank properties and square-root decomposition of these non-negative matrices.

Square root forms of iterations like the DRDE have been developed for several scenarios in the literature. The square root algorithm (SQR) is based on the DRDE with $Q = 0$ and developed in Chapter 5. The feedback generated in the limit moves the unstable eigenvalues of A, λ to their unit circle mirror images, $1/\lambda$, and leaves the stable eigenvalues unchanged. As a special case of the square root form of DRDE which was introduced in [13] for Kalman filtering, the SQR stabilization algorithm has the form

$$\begin{bmatrix} R^{1/2} & B'P_i^{1/2} \\ 0 & A'P_i^{1/2} \end{bmatrix} U_i = \begin{bmatrix} (R_i^{\epsilon})^{1/2} & 0 \\ \tilde{K}_i & P_{i+1}^{1/2} \end{bmatrix} \tag{1.4}$$

where U_i is orthogonal and the dimension of $P_i^{1/2}$ is $n \times l$, the same as $P_0^{1/2}$. Note the QR decomposition is computed for a small matrix with size $(p+l) \times p$ (the first row of (1.4)) and feedback F_i can be computed from $(R_i^{\epsilon})^{1/2}$ and $\tilde{K}_i$.

The SQR iteration can produce the same sequence of subspaces as SSI with only an additional economical QR decomposition of $P_i^{1/2}$ since the updating of $P_i^{1/2}$ has the form $P_{i+1}^{1/2} = A'P_i^{1/2}U_i^{22}$. If $P_0^{1/2}$ is taken to be the same initial subspace basis as used for SSI, SQR iteration will converge. Even if SSI does not converge, SQR will converge under conditions related to the ability of the SRR to extract a convergent subsequence of approximated unstable invariant subspace.

As noted above, the use of an effective termination check is needed in order to avoid iterating longer than necessary. The square root algorithms in the literature usually use a stopping criterion analogous to the convergence of the feedback increment $F_i - F_{i-1}$, which is determined by smallest distance of eigenvalues (especially unstable eigenvalues) of A to unit circle and would suffer slow convergence if some eigenvalues are too close to unit circle. We have developed and evaluated a more effective method which can catch stability of $A - BF_i$ earlier in the SQR iteration. This is achieved by (1) extracting an approximation to the unstable subspace V_i via SRR from $P_i^{1/2}$ and (2) comparing the eigenvalues of $V_i'(A - BF_i)V_i$ and unit circle mirror images of eigenvalues of $V_i'AV_i$. Empirical evidence indicates that this test effectively detects stabilization much earlier than other convergence tests for square root-like methods.

We have also developed a second SQR algorithm that can be useful in some circumstances where the algorithm above has difficulties and which can be used as the basis for a general DRE solver. It is based on an alternative form of the DRDE

$$P_{i+1} = (A - BF_i)'P_i(A - BF_i) + F_i'RF_i + Q \tag{1.5}$$

where $P_i = (R + B'P_iB)^{-1}B'P_iA$ and derived by decomposing every term in (1.5) into its square root form. First form the matrix

$$\left[(A - BF_i)' P_i^{1/2} \;\; F_i' R^{1/2} \;\; Q^{1/2}\right]$$

and use economical SVD to decompose it into the form $U_i S_i V_i$ where the diagonal elements of S_i are ordered with larger ones on top. The matrix $P_{i+1}^{1/2}$ is formed by first several columns of U_i and corresponding submatrix of S_i (hence this algorithm is referred to the truncated square root algorithm – TSQR). With $Q = 0$, this truncated algorithm is equivalent to the SQR of the same rank. However, by using nonzero Q, the spectral radius of $A - BF$ can be influenced and therefore the convergence of the iteration. Care must be taken with TSQR when Q is nonzero to avoid certain numerical problems. Unless the rank of $P_i^{1/2}$ is large enough, TSQR can have trouble with convergence and stability. These difficulties can be solved by limiting the iteration to the unstable left invariant subspace of A, which can be achieved by updating Q with an orthogonal iteration on each iteration of (1.5). As a result, in practice, the rank of the basis matrix propagated by TSQR is usually slightly larger than the number of unstable eigenvalues of A and that used by SQR.

The resulting SQR and TSQR methods essentially propagate a low-rank approximation to a basis of the invariant subspace and information needed to cheaply compute the stabilizing feedback. When parameters are carefully chosen the convergence is less sensitive than that of SSI/SRR and performance is as least as good and often significantly better than Saad's method. Therefore, they address the first two difficulties with Saad's method. However, since l, the rank of the (T)SQR approximation, or the rank of $P_0^{1/2}$, is at least the number of unstable eigenvalues of A, it still requires the propagation of a basis of the unstable invariant subspace.

The similarities of SQR to projection and subspace iteration-based algorithms are analyzed in Chapter 5.

A true low-rank stabilization algorithm, CSQR, that does not require the propagation of an estimate of the basis of the unstable space is developed in Chapter 6.

It is based on another well-known recurrence, the square root Chandrasekhar [13] algorithm, that uses the DRDE with $P_0 = 0$.

Let L_i be the square root of $P_{i+1} - P_i$, starting from $L_0 = C$, $R_0^\epsilon = R$ and $\tilde{K}_0 = 0$, CSQR has the form

$$\begin{pmatrix} (R_{i-1}^\epsilon)^{1/2} & B'L_{i-1} \\ \tilde{K}_{i-1} & A'L_{i-1} \end{pmatrix} U_i = \begin{pmatrix} (R_i^\epsilon)^{1/2} & 0 \\ \tilde{K}_i & L_i \end{pmatrix} \tag{1.6}$$

where U_i is orthogonal and the feedback F_i can be computed from $\tilde{K}_i(R_i^\epsilon)^{-1/2}$. The dominant computation in each iteration of CSQR is $A'L_{i-1}$, whose rank is just the rank of C. CSQR will converge for any choice of C. For the purpose of stabilization, C should satisfy the condition that (A', C) is stabilizable. Typically, C can be taken as a rank 1 matrix but in general, its rank must be at least the largest multiplicity of any unstable eigenvalue of A. Note that the dimensions of the matrices propagated do not depend on the dimension of the unstable space.

The CSQR algorithm therefore addresses all three difficulties with Saad's method and in the form above is practical for many problems. It is possible, however, to develop techniques that improve its performance significantly.

Feedback convergence is determined by the spectral radius of the closed loop matrix $A - BF$. If some eigenvalues of A are very close to unit circle, the convergence is usually slow. There are several methods which can accelerate the stabilization process and reduce the number of steps of iteration needed to compute a stabilizing feedback.

The first method is to limit the iteration to the unstable invariant subspace, which is approximated by performing preprocessing on C consisting of several steps of the power method (for any choice of C, use normalized $(A')^k C$ to replace C with a small k, usually 5 or 10). Such simple preprocessing with, e.g., $k = 10$, has been observed to save hundreds of CSQR iterations.

The second method is a deflation/restart version of CSQR. In this version, every K CSQR iterations, we compute F_K and deflate A to $A - BF_K$, then choose a new

C (usually from the previous L_K) and restart CSQR. With this method, once an unstable eigenvalue is stabilized, it will be deflated out so the new round of CSQR will only focus on the unstable eigenvalues yet to be stabilized.

The third method takes advantage of the property that CSQR tends to move the unstable eigenvalues close to unit circle very quickly if does not stabilize them. With scaling the system by a constant $0 < r < 1$, the feedback F_i from CSQR iteration based on the new system $(A/r, B/r)$ should stabilize the original system (A, B) much faster than CSQR based on (A, B). The stability of $A - BF_i$ in deflation/restart CSQR and CSQR via scaling A can be caught with $s_i = (A - BF_i) \cdots (A - BF_1)B$ which should converge to zero and usually $\|s_i\|_2 / \max_{k=1}^{i} \|s_k\|_2$ within 10^{-2} is enough to signal stability of $A - BF_i$. Unless the eigenconditioning of corresponding closed-loop matrix is very poor, both deflation/restart CSQR and CSQR via scaling A are much more effective than usual CSQR. CSQR via scaling A with r around 0.9 can save an order of magnitude on computations.

For rank l SSI (measured by rank of initial subspace), SQR and TSQR (measured by rank of $P_0^{1/2}$), CSQR (measured by rank of L_i), the computation costs of one step iteration are roughly close, $(n + p)nl + (p + l)p^2 + n(p + l)^2$ flops for dense (A, B) and $O(\rho n)l + (p + l)p^2 + n(p + l)^2$ flops for sparse (A, B) with an average nonzero count per row of ρ, while a QR decomposition on an $n \times n$ matrix needs $O(n^3)$ computations. Note the rank of subspace iteration, SQR and TSQR is at least the number of unstable eigenvalues of A, and the rank of CSQR is at most the maximum multiplicity of unstable eigenvalues of A and usually can be chosen as 1 (or 1 can be used with deflation), so CSQR will be the best choice if no knowledge about the number of unstable eigenvalues exists or this number is too large.

Chapter 7 discusses a different method to solve discrete-time Riccati equation - an approximate Newton's algorithm. We propose a truncated square root algorithm (TSQR) which uses the SVD to truncate a square root form (different from square root of Riccati difference equation in other chapters) of the approximate Newton's iteration of which the Riccati Difference Equation is a special case. The stabilization algorithm form of TSQR shares much with SQR for $Q = 0$ and has the same rank. TSQR, however, stabilizes and converges faster and requires less overall work than SQR.

Chapter 8 is dedicated to continuous-time stabilization problem

$$\dot{x} = Ax + Bu \tag{1.7}$$

where A and B are $n \times n$ and $n \times p$ real matrices and $x(t)$, $u(t)$ are vectors of dimension n and p respectively. Almost all results from discrete-time stabilization can be ported to continuous-time stabilization under a conformal transformation. Relationships between discrete-time Riccati equation and continuous-time Riccati equation are analyzed and a low-order polynomial approximation of conformal transformation to avoid matrix inverse is explored.

The final chapter of the thesis summarizes future work that we believe important to large scale stabilization algorithms.

CHAPTER 2

BASIC THEORY AND CONDITIONING

In this chapter, we introduce the fundamentals of the stabilization problem. First basic definitions are given. Theory demonstrating that feedback stabilization is possible follows. We then review and summarize results on the conditioning of stabilization problems. Most of these results are extracted and modified from papers on conditioning of eigenvalue assignments ([1], [2]). The discussion attempts to identify the factors that can cause difficulties when constructing stabilization algorithms. We propose two methods to measure the conditioning of stabilization. The first is based on Theorem 2.1 and an estimate of the minimal singular values of the eigenvector matrix of the closed-loop system. The second is the optimal stability radius of matrix triple (A, B, F). The conditioning of the stabilization problem is then investigated for two examples of standard forms of dynamical systems. Matrices A, B, C are assumed to be of dimension $n \times n, n \times p, m \times n$ respectively throughout.

2.1 Basic Definitions and Theorems

We first define the basic properties of the matrix pencils associated with a dynamical system: stability, stabilizability, detectability, controllability, observability, and reachability.

Definition 1 (Stability:) *A matrix A is stable if all of its eigenvalues are inside the unit circle for the discrete-time case, or are located in the left half complex plane (negative real part) for the continuous-time case.*

Definition 2 (Stabilizability:) *A system (A, B) is stabilizable:*

1. *if there is a matrix F such that $A - BF$ is stable,*

2. *or if $v'A = \lambda v'$ and $v \neq 0$, where λ is an unstable eigenvalue of A implies $v'B \neq 0$,*

3. *or if $(B, A - \lambda I)$ is of full rank for any unstable eigenvalue λ of A.*

Definition 3 (Detectability:) *(C, A) is detectable:*

1. *if (A', C') is stabilizable,*

2. *or if $Av = \lambda v$ and $v \neq 0$, where λ is an unstable eigenvalue of A implies $Cv \neq 0$.*

Definition 4 (Controllability:) *(A, B) is controllable:*

1. *if the matrix $(A, AB, \cdots, A^{n-1}B)$ is of rank n,*

2. *or if all matrices $(B, A - \lambda_i I)$ are full rank for all eigenvalues λ_i of A.*

A particular λ is controllable if $(A - \lambda I|B)$ is of full rank.

Note that controllability can be measured by

$$\min_{\lambda \in \mathcal{C}} \sigma_n(B, A - \lambda)$$

where σ_n is the smallest singular value of the matrix for a given value of λ.

Definition 5 (Observability:) *(C, A) is observable:*

1. *if (A', C') is controllable,*

2. *or if, $Av = \lambda v$ and $v \neq 0$ implies, that $Cv \neq 0$.*

Definition 6 (Reachability:) *A mode λ of (A, C') is reachable if $A'v = \lambda v$ and $v \neq 0$ implies $Cv \neq 0$.*

Theorem 2.1 shows that the feedback for eigenvalue assignment (pole placement) or stabilization problem can be computed directly.

Definition 7 (Image and Kernel:) *The image of a matrix A, denoted $IM(A)$, is all vectors v such that $v = Ax$ for some vector x. The kernel of a matrix A, denoted $Ker(A)$, is all vectors v such that $Av = 0$.*

Theorem 2.1 *(see [14], page 55, exercise 2.2) Let $\lambda_i, i = 1, \cdots, n$ be a symmetric set of n distinct complex numbers, and let $v_i, i = 1, \cdots, n$ be nonzero vectors such that $v_i = v_j^*$ if $\lambda_i = \lambda_j^*$, or real if λ_i real. Then there exists F such that $(A-BF)v_i = \lambda_i v_i, i = 1, \cdots, n$ iff the v_i are independent and $v_i \in (A-\lambda_i I)^{-1} IM(B)$. Furthermore, F is unique when $Ker(B) = 0$.*

Proof. (IF): Let

$$F = (B'B)^{-1}B'(A - V\Lambda V^{-1}), \tag{2.1}$$

where

$$\begin{aligned} V &= (v_1, \cdots, v_n) \\ \Lambda &= diag(\lambda_1, \cdots, \lambda_n). \end{aligned}$$

Assume λ_i is simple, then

$$\begin{aligned} (A - BF)v_i &= (A - B(B'B)^{-1}B'(A - V\Lambda V_{-1})v_i \\ &= Av_i - B(B'B)^{-1}B'(Av_i - \lambda_i v_i) \end{aligned}$$

From $v_i \in (A - \lambda_i)^{-1} IM(B)$, $v_i = (A - \lambda_i)^{-1} B\beta_i$, that is $(A - \lambda_i)v_i = B\beta_i$, we have

$$
\begin{aligned}
(A-BF)v_i &= Av_i - B(B'B)^{-1}B'(B\beta_i) \\
&= Av_i - B\beta_i \\
&= Av_i - (A-\lambda_i)v_i \\
&= \lambda_i v_i
\end{aligned}
$$

(ONLY IF): Note that $(A-BF)v_i = \lambda_i v_i$ means $v_i = (A-\lambda_i)^{-1}BFv_i$ if $A-\lambda_i$ is nonsingular, $v_i \in (A-\lambda_i)^{-1}IM(B)$. The independence of $v_1, \cdots, v_n$ comes from the fact that v_i is the eigenvector of A.

(UNIQUENESS): Assume F_1, F_2 satisfy $(A-BF)v_i = \lambda_i v_i$ for $i = 1, \cdots, n$, i.e., $A - BF = V\Lambda V^{-1}$. $B(F_1 - F_2) = 0$. Since $Ker(B) = 0$, $F_1 - F_2 = 0$, so, $F_1 = F_2$.■

If $A - \lambda_i I$ is singular, which means λ_i is an eigenvalue of A, the computation can proceed by replacing $v_i \in (A-\lambda_i)^{-1}IM(B)$ with the eigenvector corresponding λ_i. For a single input system, the solution is unique since $Ker(B) = 0$. For a multiple input system, there is more freedom. In [1] there are several algorithms to choose v_i to minimize the condition number of V . Theorem 2.1 is not a bad choice to construct feedback F for a small system. For a large scale problem, however, the computational effort is prohibitive.

While Theorem 2.1 cannot be used as the basis for an efficient algorithm for large systems, it does give very important information about conditioning of eigenvalue assignment or stabilization which we discuss later in the thesis.

If a problem has only a very few unstable eigenvalues, and if these eigenvalues and eigenvectors are easy to compute, a deflation method can be used to stabilize each of the eigenvalues in turn. The deflation method is detailed in [10].

Assume an eigen-pair (λ_1, v_1) is known and we want to move λ to σ. For simplicity, consider a single input system, i.e., $p = 1$. If we define

$$
F = \frac{(\lambda_1 - \sigma)}{(B'v_1)} v_1' \tag{2.2}
$$

then the eigenvalue move can be accomplished via feedback according to the following theorem.

Theorem 2.2 *([10]) Let A have eigenvalues $\lambda_1, \cdots, \lambda_n$ with associated left eigenvectors $v_1, \cdots, v_n$ and associated right eigenvectors $u_1, \cdots, u_n$. Then, the eigenvalues of $A - BF$ are $\sigma, \lambda_2, \cdots, \lambda_n$. The corresponding left eigenvectors are*

$$v_1, v_2 - \alpha_2 v_1, \cdots, v_n - \alpha_n v_1$$

where

$$\alpha_i = \frac{(\lambda_1 - \sigma)B' v_i}{B' v_1 (\lambda_i - \sigma)}$$

and the corresponding right eigenvectors are $\hat{u}_1, u_2, \cdots, u_n$, where

$$\hat{u}_1 = \frac{1}{\|(A - \sigma)^{-1}B\|_2}(A - \sigma)^{-1}B$$

The deflation method can be used in various ways. We can find an eigenvalue/eigenvector pair, compute F with Equation (2.2), update A to $A - BF$ and repeat this process until all the eigenvalue moves required have been accomplished. An alternative way is to use Theorem 2.2: first compute the eigenvalue/eigenvectors pairs to be moved, then repeatedly compute F with Equation (2.2) and update the eigenvectors with above theorem.

For a multiple input system, we can split B into columns $B_1, \cdots, B_p$. Any B_i such that $B_i' v_1 \neq 0$ can replace B in (2.2). We can also balance in several qualified B_is to minimize feedback norm or condition number.

If λ_1 and v_1 are complex they must be part of a complex conjugate pair and using deflation twice suffices to guarantee a final real F.

Deflation based on Equation (2.2) and Theorem 2.2 is used later in the thesis as a tool to analyze or preprocess systems with one or two very large unstable eigenvalues whose presence can slow the convergence of our stabilization algorithms. A different type of deflation is also used later in the thesis to construct a stabilization algorithm, where we update A to $A - BF$ repeatedly and the eigenvalues which have been stabilized will be filtered out so that further computations only focus on the eigenvalues yet to be stabilized.

2.2 Conditioning of Stabilization Problems

In order to develop effective algorithms it is crucial that we determine which factors of the stabilization problem and the proposed algorithms influence the conditioning of the results and the effort required to stabilize. The first determines the types of stabilization problems that are reasonable to attempt. The second indicates the appropriate form of algorithm to use for a particular problem.

For any eigenvalue assignment algorithm, we have matrices A, B and the desired eigenvalues $\Lambda = (\lambda_1, \cdots, \lambda_n)$ as input parameters and feedback F as output parameter. Small changes like $\Delta A, \Delta B$ and $\Delta\Lambda$ in the input parameters will cause change ΔF in output. Saad in [10] uses ΔF as a function of ΔA or ΔB as a measure of difficulty of the stabilization problem. While this measure is useful, especially in the context of iterative methods producing a series of estimates of F, it does not capture all important aspects of the problem.

The purpose of eigenvalue assignment is to construct a matrix which has the desired eigenvalues. Small ΔF is not enough to guarantee small perturbations in the resulting eigenvalues. Some small changes in the closed-loop matrix $A - BF$ can cause big differences in the resulting eigenvalues. It is very important, therefore, to take the eigen-conditioning of $A - BF$ into consideration. For the purpose of stabilization, any small changes in the input parameters such as A, B and/or other algorithm-specific parameters should not cause large changes in output feedback F and large eigenvalue perturbations in $A - BF$. The major concern of [2] and [1] is to find a way to reduce the eigen-condition number of the closed-loop matrix.

The stabilization problem is somewhat less restrictive than the eigenvalue assignment problem. The goal is to produce a closed-loop matrix that is stable not necessarily with a specific spectrum. Therefore, a key concern is that the closed-loop matrix that results from the algorithm be robustly stable, i.e., it does not have nearly unstable eigenvalues that may become unstable under small perturbations. It follows then that the stability radius of the closed-loop matrix is very important as a measure of an algorithm's robustness. It can be used to assess the difficulty of the problem. When the stability radius can be shown to be small, the stabilization problem is poorly conditioned. The eigen-condition number can be used to estimate a bound on the stability radius of the closed-loop matrix. We discuss methods for this estimation below.

Sun has discussed perturbations to eigenvalue assignment and Riccati equations (which are used to construct our stabilization algorithms) based on first principles of conditioning [15], [16], [17]). Unfortunately, the approach does not yield much into the influence of the structure of the problem on stabilization difficulty. The computation of the resulting condition numbers requires a large Kronecker product and a solution of eigenvalue assignment problem or Riccati equation, and, therefore, is useful only for very small systems.

2.3 Closed-Loop Eigen-conditioning

The eigen-conditioning of a matrix A can be measured by the condition number of the matrix whose columns are the normalized eigenvectors of A.

Proposition 2.1 *Let columns of matrix X be the normalized right eigenvectors of A such that $A = X\Lambda X^{-1}$ where the diagonal elements of $\Lambda = diag(\lambda_1, \cdots, \lambda_n)$ are eigenvalues of A. We then have*

$$\kappa_2(X) = \|X\|_2 \|X^{-1}\|_2 \geq \frac{\|A\|_2}{max_j|\lambda_j|}. \tag{2.3}$$

Proof. Taking the 2-norm on both sides of $A = X\Lambda X^{-1}$, we have $\|A\|_2 = \|X\Lambda X^{-1}\|_2 \leq \|X\|_2\|X^{-1}\|_2\|\Lambda\|_2 \leq \kappa_2(X) max_j|\lambda_j|$. ■

Corollary 2.2 *Let the columns of X be the normalized right eigenvectors of the closed-loop $A - BF$ of system (A, B) with feedback F and let $\lambda_j, j = 1, \cdots, n$ be the closed-loop eigenvalues. We then have*

$$\kappa_2(X) \geq (\|BF\|_2 - \|A\|_2)/max_j|\lambda_j|. \tag{2.4}$$

Corollary 2.2 shows the effect of F or BF on eigen-conditioning of closed-loop matrix $A - BF$. So, it is necessary to limit the magnitude of $\|F\|_2$ relative to $\|A\|_2$ for a given B. Note, however, that since we are dealing with a lower bound, small $\|F\|_2$ does not necessarily mean small $\kappa_2(X)$. and, therefore, it might not be useful to find the feedback with minimum norm.

A bound on the perturbation that can be made to the closed-loop system such that stability is retained is derived in ([1]).

Proposition 2.3 *If the state feedback F assigns the set of stable eigenvalues $\lambda_j, j = 1, \cdots, n$, then the perturbed closed-loop matrix $A - BF + \Delta$ remains stable for all disturbances Δ which satisfy, in continuous-time case,*

$$\|\Delta\|_2 < min_{\lambda=i\omega}\sigma_n(\lambda I - (A - BF)) \equiv \delta(F). \tag{2.5}$$

A lower bound on $\delta(F)$ that can be used as a reasonable approximation is given by

$$\delta(F) \geq min_j Re(-\lambda_j)/\kappa_2(X). \tag{2.6}$$

In the discrete-time case, the closed-loop system remains stable for disturbances Δ which satisfy

$$\|\Delta\|_2 < min_{|z|=1}\sigma_n(zI - (A - BF)) \equiv \delta(F) \tag{2.7}$$

A lower bound of $\delta(F)$ that can be used as a reasonable approximation is given by

$$\delta(F) \geq min_j(1 - |\lambda_j|)/\kappa_2(X). \tag{2.8}$$

Equation 2.8 is in fact an estimate of the stability radius of the closed-loop system. The same lower bound can be derived from the *Bauer-Fike* theorem ([18]).

Theorem 2.3 (Bauer-Fike) *If μ is an eigenvalue of $A + E$ and $X^{-1}AX = D = diag(\lambda_1, \cdots, \lambda_n)$, then*

$$\min_j |\lambda_j - \mu| \le \kappa_p(X)\|E\|_p$$

where $\|\cdot\|_p$ denotes any of the Holder norms.

For stable matrix A, the bounds on Δ given by $\kappa_2(X)$ from Proposition 2.3 and Bauer-Fike theorem are too tight.

An alternative bound can be derived from the complex Schur decomposition of $A - BF$ ([18]):

Theorem 2.4 (Schur) *Let $Q^H AQ = D + N$ be the Schur decomposition where D is diagonal and N is strictly upper triangular. It follows that*

$$\min_j |\lambda_j - \mu| \le \max(\theta, \theta^{1/q})$$

where $\theta = \|E\|_2 \sum_{k=0}^{q-1} \|N\|_2^k$ and q is the smallest integer such that $N^q = 0$.

The N in the Schur decomposition tends to give a better bound when q is small. In continuous-time case, $\kappa_2(X)$ gives the bound of

$$\|\Delta\|_2 \le \min_j Re(-\lambda_j)/\kappa_2(X)$$

and the Schur decomposition of A gives the bound

$$\|\Delta\|_2 \le \min\left\{\min_j Re(-\lambda_j), (\min_j Re(-\lambda_j))^q\right\}/(\sum_k^{q-1} \|N\|_2^k).$$

In discrete-time case, the bounds of $\|\Delta\|_2$ are

$$\|\Delta\|_2 \le \min_j(1 - |\lambda_j|)/\kappa_2(X)$$

and

$$\|\Delta\|_2 \le (\min_j(1 - |\lambda_j|)^q)/(\sum_k^{q-1} \|N\|_2^k$$

respectively.

Example 1 demonstrates a situation where the Schur decomposition of A gives better bound when q is small.

Example 1 *Take a discrete-time case where* $A = \begin{bmatrix} 0.9 & 1 \\ 0 & 0.9-\delta \end{bmatrix}$ *which has eigen condition number* $\kappa_2(X) = 2/\delta$ *where* $\delta = 10^{-6}$. *With Bauer-Fike theorem or Proposition 2.3, the estimated bound of* Δ *such that* $A+\Delta$ *remains stable is* 5×10^{-8}. *With* N *from Schur decomposition of* A*, it is easy to verify that* $q = 2$ *and then the bound for* Δ *is 0.005, which is much better than the bound from* $\kappa_2(X)$.

2.4 Condition Number of Stabilization

Analysis using $\kappa_2(X)$ can overestimate the difficulty of stabilization or eigenvalue assignment problems. Nevertheless, it is always worth checking this number if some difficulties are encountered during the stabilization process. In general, if $\kappa_2(X)$ is very small, we should have a good bound on feedback disturbance ΔF and stabilization or pole placement can typically be done with moderate effort. However, both $\kappa_2(X)$ and direct ΔF analyses share drawbacks. Both have reliability problems and they are a posteriori analyses since they require the closed-loop matrix. As a result, when very difficult problems are encountered F may not be known and therefore analysis to explain the large effort required to stabilize or place poles may not be tractable via these techniques. Even when stabilization is possible the analyses may conflict. Example 2 illustrates such a situation.

Example 2 *Let* $A = diag(2 + i/20, i = 1, \cdots, 10)$ *and* $B = (1, \cdots, 1)'$. *It is easy to verify that* (A, B) *is stabilizable and controllable. The estimated distance to uncontrollability (defined by* $\min_\lambda \sigma_n(A - \lambda I|B)$*) is 0.0034 with* $\lambda = 2.1494$. *In* *theory, we should be able to move the eigenvalues of A to anywhere we like. In practice, all of the standard algorithms fail to produce a stabilizing feedback.*

This example also shows the instability of some stabilization and eigenvalue assignment algorithms which need to do some orthogonal transformation on (A, B). The Hessenberg standard form (H, b) of (A, B), where $H = Q'AQ$ is a Hessenberg matrix, $b = Q'B$ has only nonzero element b_1 and Q is orthogonal, can be stabilized with a feedback f via different stabilization algorithms. But $A - BF$ with $F = fQ'$ is not stable. To get $H - bf$ stabilized, we have $\|f\|_2$ around 10^{11} and $\kappa_2(X)$ for $H - bf$ is around 10^{23}. If we add small disturbance to A or B, the corresponding $\|\Delta f\|_2/\|f\|_2$ is also small and it therefore fails to catch the difficulty of stabilization.

In Theorem 2.1, the independence of $v_1, \cdots, v_n$ is required. In single input case, it is equivalent to independence of $(A - \lambda_i I)^{-1}B, i = 1, \cdots, n$ after normalization. For multiple inputs system, $(A - \lambda_i I)^{-1}B,\ i = 1, \cdots, n$ must be of full rank. In above example, this independence holds in theory, but not numerically.

Definition 8 (Numerical Rank:) *For a specified accuracy tolerance ϵ, the numerical rank of a matrix A is the number of its singular values greater than ϵ.*

With the definition of numerical rank, we randomly chose a set of stable eigenvalues and assign this set to the system (A, B) of Example 2. The corresponding numerical rank of normalized $(A - \lambda_i I)^{-1}B,\ i = 1, \cdots, 10$ was found to be 7 with machine tolerance taken as ϵ. All of the numerical algorithms failing to reach a stabilizable feedback is therefore not a surprise. This example indicates the need to investigate conditioning analysis techniques that are a priori and that do not depend upon a particular stabilizer F. These techniques may therefore be more computationally intensive but it is expected that they will provide more insight into difficulties with stabilization and indicate a preference between the many stabilizers that may exist for a given problem.

Clearly, we need to consider the numerical rank of the normalized $(A-\lambda_i I)^{-1}B$, $i = 1, \cdots, n$. A metric of stabilizability based on the minimum singular value of the matrix constructed from the normalized $(A-\lambda_i I)^{-1}B$, $i = 1, \cdots, n$ is a good starting point.

Definition 9 (Condition number for stabilizability:) *The condition number for stabilizability of a system (A, B) is defined as the reciprocal of the maximum of the minimum singular value of the matrix constructed as follows. Let $\lambda_1, \cdots, \lambda_n$, from $\{theorthogonalbasisof(A-\lambda_i)^{-1}B,\ i = 1, \cdots, n\}$ where the maximum takes over all $\lambda_1, \cdots, \lambda_n$ which are stable and symmetric (in conjugate pair if complex), if $(A-\lambda_i)$ is singular, $(A-\lambda_i)^{-1}B$ is replaced by corresponding right eigenvector of A.*

Definition 10 (Condition number for stabilizability:) *The condition number for stabilizability of a system (A, B) is defined as the reciprocal of the maximum of the minimum singular value of the matrix constructed as follows. Let $\Lambda = diag(\lambda_1, \cdots, \lambda_n)$, be a diagonal matrix specifying a target spectrum which consists or real or complex conjugate eigenvalues. For each λ_i find an orthogonal basis, S_i for $(A-\lambda_i)^{-1}B$ (which is an $n \times p$ matrix). If $(A-\lambda_i I)$ is singular, $(A-\lambda_i I)^{-1}B$ is replaced by corresponding right eigenvector or principal vector of A. If λ_i is a multiple eigenvalue of A then several S_i will be so generated. Concatenate the S_i to form the matrix $S(\Lambda) = [S_1, \ldots, S_n]$. The condition number is then given by*

$$\kappa^{-1} = \max_{\Lambda} \left(\sigma_n(S(\Lambda))\right)$$

where $\sigma_n(\Lambda)$ denotes the smallest singular value of Λ and the maximum is over all matrices Λ as described above.

From Theorem 2.1, the eigenvectors V of corresponding closed-loop $A-BF$ which has the eigenvalues $\lambda_1, \cdots, \lambda_n$ should have the form SD with $D = diag(D_1, \cdots, D_n)$. The following theorem establishes the lower bound of the condition number of V ([1]).

Proposition 2.4 *The condition number of V is*

$$\kappa_2(V) \geq \|S^+\|_2 \tag{2.9}$$

where S^+ denotes the Moore-Penrose inverse of S.

We do not need to compute S^+ to get the lower bound. In fact, $\|S^+\|_2$ is the reciprocal of the smallest singular value of S, i.e.,

$$\kappa_2(V) \geq 1/\sigma_n(S)$$

Using $1/\sigma_n(S)$ to measure the difficulty of eigenvalue assignment or stabilization has at least one advantage, it is invariant under the deflation method which moves eigenvalues step by step to the target eigenvalues. For example, if we want to move the eigenvalues of A to $\Lambda = (\lambda_1, \cdots, \lambda_n)$ with feedback F, we first move to $\Gamma = (\gamma_1, \cdots, \gamma_n)$ with feedback F_1, then go to Λ with feedback F_2 from $A-BF_1$. We should have the same $\sigma_n(S)$ for both (A, B) and $(A-BF_1, B)$ which will be assigned eigenvalues Λ since only the eigenvectors of the final closed-loop are concerned.

Numerical experiments show this bound does predict difficulties for eigenvalue assignment and stabilization and is much better than other measurements such as distance to an unstablilizable system, which is defined by $\min_{Re(\lambda)>0} \sigma_n(\lambda I - A|B)$ for continuous-time system and $\min_{|\lambda|>1} \sigma_n(\lambda I - A|B)$ for discrete-time system. For the stabilization problem, the maximization should cover all symmetric stable $(\lambda_1, \cdots, \lambda_n)$ to find the largest possible $\sigma_n(S(\Lambda))$, which is equivalent to finding optimal stable eigenvalues such that the eigen-condition number of the corresponding closed-loop matrix is minimized. Although the maximum of $\sigma_n(S(\Lambda))$ is more accurate to describe stabilizability or difficulty of eigenvalue assignment, the computation of such a measurement is not trivial, especially for large scale stabilization problems. We have successfully used a sampling process to estimate the bound. We sample a long list of stable $\lambda_i, i = 1, \cdots, N$ with $N > n$ and construct S from $S_i, i = 1, \cdots, N$. The computation $\sigma_n(S)$ finds the best combinations of n columns among $S_i, i = 1, \cdots, N$. With this method using $N = 200$, we can compute the $\sigma_n(S)$ of Example 2 to be 3.9×10^{-16} and the a priori condition number for stabilizability is around 2.558×10^{15}. This clearly indicates difficulty regardless of the algorithm used.

2.5 Stability Radius of (A, B, F)

The bounds of Section 2.3 exploit the fact that the closed loop matrix has the form $A - BF$ in that they require a particular F but, the perturbation affects all elements of $A - BF$ and therefore ignores the structure. The condition number of Section 2.4 is not specific to a particular F but it also ignores the structure of the closed-loop matrix. An alternate, and in our experience superior, way to measure the conditioning of stabilization exploits the structure of the closed-loop matrix. This metric is the optimal (largest) stability radius of matrix triple (A, B, F) over all stabilizing F. The stability radius of matrix triple (A, B, F) with respect to p-norm is defined as

$$r_{\mathbb{C}}(A, B, F) = \inf_{\Delta F} \left\{ \|\Delta F\|_p : A - B(F + \Delta F) \text{ is not stable} \right\}. \tag{2.10}$$

The conditioning of stabilization can be measured by the maximum of $r_{\mathbb{C}}(A, B, F)$ over all stabilizing F.

Let $\partial\mathbb{C}_g$ denote the unit circle in discrete-time case and the imaginary axis in continuous-time case. Following the work of Van Dooren in [19], we have

$$
\begin{aligned}
r_{\mathbb{C}}(A,B,F) &= \inf_{\Delta F}\Big\{\|\Delta F\|_p : (A - B(F+\Delta F))\bigcap \partial\mathbb{C}_g \neq 0\Big\} \\
&= \inf_{\lambda\in\partial\mathbb{C}_g}\Big[\inf_{\Delta F}\big\{\|\Delta F\|_p : det(I - \Delta F(\lambda I - (A-BF))^{-1}B) = 0\big\}\Big] \\
&= \inf_{\lambda\in\partial\mathbb{C}_g}\|(\lambda I - (A-BF))^{-1}B\|_p^{-1}. \qquad (2.11)
\end{aligned}
$$

Compared to $\sigma_n(S(\Lambda))$ as a measurement of conditioning of stabilization, the estimation of $\max_F r_{\mathbb{C}}(A,B,F)$ over all stabilizing F is very costly, since, at least, we need to find a sampling of stabilizing F. Of course, this implies that for the system (A,B) in Example 2 where we cannot find a stabilizable F, it is impossible to estimate $\max_F r_{\mathbb{C}}(A,B,F)$. On the other hand, we can use $r_{\mathbb{C}}(A,B,F)$ as a measurement to compare different feedbacks F. If we have several feedbacks F computed from different algorithms, it is reasonable to choose one with largest $r_{\mathbb{C}}(A,B,F)$. The value of $r_{\mathbb{C}}(A,B,F)$ can be estimated as $\min_j \|V_j\|_p^{-1}$ by solving a sequence of linear systems of the form (from Equation (2.11))

$$(z_j I - (A-BF))V_j = B$$

where $z_j, j = 1,\cdots,N$ are sampled from unit circle for discrete-time system or imaginary axis for continuous-time system. In this form, we see that it is reasonable to investigate ways of integrating such a scheme into the algorithms that produce F in order to bias the iterations toward an F with a reasonable stability radius.

2.6 A Standard Form Where A is Symmetric

The literature on stabilization difficulty also contains analyses of model problems where the system is assumed to be in particular standard forms, e.g., [2]. In this section and the next we discuss two standard forms of single-input control system (A,B) and determine the interplay of factors that cause $\|F\|_2/\|A\|_2$ to become large and therefore increase the difficulty of stabilization or pole placement.

Assume F is a stabilizing feedback of system (A, B) and $A - BF$ has eigenvalues $\lambda_j, j = 1, \cdots, n$. For a single-input system it is possible to derive an explicit formula for the feedback F.

For any λ_j, $\|v_j\|_2 = 1$ such that $(A - BF)v_j = \lambda_j v_j$, we have

$$v_j = (A - \lambda_j I)^{-1} BFv_j. \tag{2.12}$$

Since Fv_j is scalar and $\|v_j\|_2 = 1$, if $(A - \lambda_j I)$ is nonsingular, v_j will have form

$$v_j = \alpha(A - \lambda_j I)^{-1} B \tag{2.13}$$

where $\alpha = 1/\|(A - \lambda_j I)^{-1} B\|_2$.

Substituting (2.13) for v_j in (2.12), we have

$$\begin{aligned}
\alpha(A - \lambda_j I)^{-1} B &= \alpha(A - \lambda_j I)^{-1} B(F(A - \lambda_j I)^{-1} B) && (2.14)\\
(A - \lambda_j I)^{-1} B &= (F(A - \lambda_j I)^{-1} B)(A - \lambda_j I)^{-1} B && (2.15)\\
(A - \lambda_j I)^{-1} B &= \beta_j (A - \lambda_j I)^{-1} B. && (2.16)
\end{aligned}$$

This yields a set of linear equations that determine F

$$\beta_j = F(A - \lambda_j)^{-1} B = 1, \; j = 1, \cdots, n. \tag{2.17}$$

Proposition 2.5 states this result formally.

Proposition 2.5 *To assign $\lambda_j, j = 1, \cdots, n$ to linear system (A, B) by feedback F where $A - \lambda_j I$ is nonsingular, the feedback is*

$$F = (1, \cdots, 1)[(A - \lambda_1 I)^{-1} B, \cdots, (A - \lambda_n I)^{-1} B]^{-1} \tag{2.18}$$

and the right eigenvector matrix of $A - BF$ is

$$E = (\alpha_1 (A - \lambda_1 I)^{-1} B, \cdots, \alpha_n (A - \lambda_n I)^{-1} B) \tag{2.19}$$

where the α_i are normalization factors.

Note that the nonsingularity assumption implies that the target spectrum is distinct from the original spectrum of A.

Consider a special case with $A = diag(\gamma_1, \cdots, \gamma_n)$, or more generally assume A is symmetric and therefore diagonalizable via an orthogonal transformation. Let

$$
\begin{aligned}
C_B &= [(A-\lambda_1)^{-1}B, \cdots, (A-\lambda_n)^{-1}B] &(2.20)\\
&= \begin{pmatrix} B_1/(\gamma_1-\lambda_1) & \cdots & B_1/(\gamma_n-\lambda_1) \\ \cdots & \cdots & \cdots \\ B_n/(\gamma_n-\lambda_1) & \cdots & B_n/(\gamma_n-\lambda_n) \end{pmatrix} &(2.21)\\
&= \tilde{B}(1/(\gamma_i-\lambda_j))_{i,j} &(2.22)\\
&\equiv \tilde{B}C &(2.23)
\end{aligned}
$$

where $B' = (B_1', \cdots, B_n')$, $\tilde{B} = diag(B_1, \cdots, B_n)$ and $C = (1/(\gamma_i - \lambda_j))_{i,j=1,\cdots,n}$. Using the form of the inverse of a Cauchy-Vandermonde matrix, we have $C^{-1} = -WC'H$ where $W = diag(w_1, \cdots, w_n), H = diag(h_1, \cdots, h_n)$ and

$$
\begin{aligned}
w_i &= \frac{\prod_{k=1}^n (\lambda_i - \gamma_k)}{\prod_{k=1,k\neq i}^n (\lambda_i - \lambda_k)}, &(2.24)\\
h_i &= \frac{\prod_{k=1}^n (\gamma_i - \lambda_k)}{\prod_{k=1,k\neq i}^n (\gamma_i - \gamma_k)}. &(2.25)
\end{aligned}
$$

The feedback can then be computed via the following generalization of a result in [2].

Corollary 2.6 *Let $A = diag(\gamma_1, \cdots, \gamma_n)$. The feedback $F = (F_1, \cdots, F_n)$ can be computed via*

$$F_i = \left(\sum_{j=1}^n \frac{\prod_{k=1,k\neq i}^n (\lambda_j - \gamma_k)}{\prod_{k=1,k\neq j}^n (\lambda_j - \lambda_k)} \right) \frac{h_i}{B_i} = \frac{h_i}{B_i}. \tag{2.26}$$

This result shows the interplay between the influence of B and the spectrum of A on $\|F\|_2$ and therefore the difficulty of the stabilization problem. If the eigenvalues of A are very close to each other, large h_i and F_i are expected. If some B_i are too small, a large magnitude of F_i is expected. For the discrete-time stabilization problem, Equation (2.26) yields lower and upper bounds of the feedback F. As a simple example, assuming all eigenvalues γ_k of A are unstable, then we have

$$\frac{(|\gamma_j|-1)^n}{\prod_{k=1,k\neq j}^{n}|\gamma_j-\gamma_k|B_j} \leq |F_j| \leq \frac{(|\gamma_j|+1)^n}{\prod_{k=1,k\neq j}^{n}|\gamma_j-\gamma_k|B_j} \tag{2.27}$$

The ratio of upper bound and lower bound of $|F_j|$ is $(\frac{(|\gamma_j|+1)}{(|\gamma_j|-1)})^n$. This ratio is much more significant to unstable eigenvalues which are closed to unit circle than to unstable eigenvalues with large magnitude.

Corollary 2.7 *Let $A = diag(\gamma_1, \cdots, \gamma_n)$. For any eigenvalue λ of $A - BF$, its condition number, defined by the reciprocal of absolute value of cosine between its left and right eigenvector, is*

$$cond(\lambda) = \frac{\sqrt{\sum_i \frac{|B_i|^2}{|\gamma_i-\lambda|^2}\sum_i \frac{|h_i|^2}{|B_i(\gamma_i-\lambda)|^2}}}{|\sum_i \frac{h_i}{|\gamma_i-\lambda|^2}|}. \tag{2.28}$$

Proof. In the induction of Proposition 2.5, we get the right eigenvector of $(A-BF)$ as $v = (A-\lambda)^{-1}B$. Similarly, its left eigenvector is $u = (A-\lambda)^{-T}F^T$. The condition number of λ is

$$cond(\lambda) = \frac{|u||v|}{|(u,v)|}.$$

Combine u, v and F from Equation (2.26), we get (2.28). ■

The effect of an individual B_i is also shown in Corollary 2.7. As B_i increases to a large magnitude or decreases to a small magnitude while all other components of B are unchanged, the conditioning worsens significantly. The Frobenius norm eigen-condition number of closed-loop system $A - BF$ can be computed directly from Equation (2.19) following ([2]).

Corollary 2.8

$$\kappa_F^2(X) = n\sum_{i,j}\frac{(\prod_{k=1,k\neq j}|\gamma_i-\lambda_k|^2)(\sum_l\prod_{k=1,k\neq l}|\lambda_j-\gamma_k|^2)}{(\prod_{k=1,k\neq i}|\gamma_i-\gamma_k|^2)(\prod_{k=1,k\neq j}|\lambda_j-\lambda_k|^2)}\frac{|B_j|^2}{|B_i|^2} \tag{2.29}$$

The distributions of the eigenvalues of both A and closed-loop system $A - BF$ could contribute to large condition number. If eigenvalues of open-loop A or eigenvalues of closed-loop $A - BF$ are very close to each other, i.e., if either matrix is near having an eigenvalue with a multiplicity more than 1, a large condition number is expected. Moreover, the balance of components in B could cause large condition number if $|B_i/B_j|$ is very large for some pair i and j.

Recall, that the definition and conditioning of C_B in Equation (2.20) and the matrices W and H that define its inverse depend on the fact $\lambda_i \neq \gamma_i$. In practice, it is quite common to move only part of the spectrum of A (partial pole placement or eigenvalue assignment). In particular, let Γ and Λ be eigenvalues of A and $A - BF$ and split Γ and Λ into $\Gamma_1 = \lambda_1, \cdots, \lambda_m, \Gamma_2 = \lambda_{m+1}, \cdots, \lambda_n$ and Λ_1, Λ_2 accordingly. Assume no common eigenvalues in Γ_1 and Λ_1, and $\Gamma_2 = \Lambda_2$. In order to avoid the ill-conditioning of the characterization of F using C_B in Equation (2.20) some sort of regularization is required. This is done via projection which is discussed in Chapter 4. Due to the projection onto the eigenspace of the eigenvalues that are moved and the uniqueness of single-input eigenvalue assignment feedback, F only depends only on Γ_1, Λ_1 and $B_i, i = 1, \cdots, m$. The condition number, however, depends on (A, B) and Λ.

For simplicity, assume $\lambda_i = \gamma_i$ for all $i > m$. The Frobenius norm eigen-condition number of the closed-loop matrix $A - BF$ is as follows.

Corollary 2.9

$$
\begin{aligned}
\kappa_F^2(X) \quad = \quad & n \sum_{i,j=1}^{m} \frac{\prod_{k=1,k\neq j}^{m} |\gamma_i - \lambda_k|^2}{\prod_{k=1,k\neq i}^{m} |\gamma_i - \gamma_k|^2} \frac{\prod_{k=1}^{m} |\lambda_j - \gamma_k|^2}{\prod_{k=1,k\neq j}^{m} |\lambda_j - \lambda_k|^2} \sum_{l=1}^{n} \frac{1}{|\lambda_j - \gamma_l|^2} \frac{|B_j|^2}{|B_i|^2} \\
& + n \sum_{i=1}^{m} \sum_{j=m+1}^{n} \frac{\prod_{k=1}^{m} |\gamma_i - \lambda_k|^2}{\prod_{k=1,k\neq i}^{m} |\gamma_i - \gamma_k|^2} \frac{\prod_{k=1,k\neq i}^{m} |\lambda_j - \gamma_k|^2}{\prod_{k=1}^{m} |\lambda_j - \lambda_k|^2} \frac{|B_j|^2}{|B_i|^2} \\
& + n(n - m).
\end{aligned}
$$

More insight can be gained by considering the problem where only a single eigenvalue is moved.

Corollary 2.10 *Assume only one eigenvalue γ_1 of A is moved to λ_1. It follows from Corollary 2.9 that*

$$\kappa_F^2(X) = n^2 + n|\lambda_1 - \gamma_1|^2 \sum_{l=2}^{n} \frac{1}{|\lambda_1 - \gamma_l|^2}(1 + \frac{|B_j|^2}{|B_1|^2})$$

The factors affecting the eigen-condition of λ are then clear. They are the distance the eigenvalue is moved, its proximity to the unchanged eigenvalues and the magnitude of B_1 compared to other B_j (more precisely, the cosine of the angle between B and the eigenvector of the eigenvalue γ_1.)

Note the order of the system n appears in all of the Frobenius norm bounds in this section. Corollary 2.10 indicates that the scale at which n appears in the condition numbers is $O(n)$. When this is adjusted to the Euclidean norm it should be $O(\sqrt{n})$. Therefore, the order of the system does not significantly affect the condition bounds above until the system is very large, e.g., $n \approx 10^6$.

For symmetric A the fact that diagonalization has occurred via an orthogonal transformation implies that the bounds above still hold. For a general A that is diagonalized by a nonorthogonal transformation, $A = Y\Gamma Y^{-1}$, the eigen-condition number of closed-loop system $A - BF$ can be bounded by the eigen-condition number of corresponding closed-loop system from Γ, $B' = (1, \cdots, 1)$ and Euclidean norm condition number of $Y\tilde{B}$ ([2]). When the eigen-conditioning of A is poor or B is poorly balanced, this bound is not very useful.

2.7 Hessenberg Form and Conditioning

Some eigenvalue assignment and stabilization algorithms require the transformation of the system (A, B) to its Hessenberg control form. The system (A, B) is transformed using an orthogonal matrix Q into the system $(Q'AQ, Q'B) = (H, h)$ where $H = (h_{ij})$ is a Hessenberg matrix, $e_1'h \neq 0$, and $e_j'h = 0$, $j = 2, \ldots, n$. For dense systems the computation of the transformation requires $O(n^3)$ operations and is therefore only viable for systems of small and moderate order. For such systems a key consideration is the perturbation analysis of the numerical computation and application of the transformation. Sun [15] discusses this in great detail.

We are interested in this form however, in order to gain insight into the conditioning and difficulty of eigenvalue assignment or stabilization for both dense and large sparse systems.

Arnold and Datta have proposed the recursive computation of the feedback as follows:

$$F = \alpha(0, \cdots, 0, 1)\psi(H)Q' \tag{2.30}$$

where $\alpha = 1/(h_1 \prod_{i=1}^{n-1} h_{i+1,i})$ and polynomial $\psi(x) = \prod_{i=1}^{n}(x - \lambda_i)$. From here, we can see the effect of $\prod_{i=1}^{n-1} h_{i+1,i}$ on F. This product relates directly to the distance to unstabilizability of the system. When one or more of the $h_{i+1,i}$ are small a large $\|F\|_2$ can result indicating difficulty in stabilization and placement. On the other hand, when the product stays moderate to large $\|F\|_2$ is acceptably small and stabilization and placement is easily accomplished.

A second important factor that can be seen from this form and that can affect $\|F\|_2$ is the depth at which unstable eigenvalues of A appear in its Hessenberg form H. More specifically, if we use the Arnoldi algorithm on A starting from B to produce the Hessenberg control form, the more steps of Arnoldi we need to extract unstable eigenvalues, the larger the stabilizing feedback will be.

Both factors are relevant for large sparse matrices that must exploit projection methods of some form in order to stabilize or place eigenvalues. Encountering small $h_{i+1,i}$ values or requiring a large number of Arnoldi iterations may indicate a problem with the conditioning of the system and not algorithmic deficiencies in Arnoldi or similar algorithms. In fact, this symptoms also relate to the convergence of matrix equation-based approaches that we present as an alternative to the projection approach of Saad and its variants. The convergence of the Riccati difference equation is also reflected in the values of $h_{i+1,i}$ and the depth of the unstable eigenvalues. These observations may lead to more careful monitoring of the difficulty of the stabilization and eigenvalue assignment problem for large sparse systems.

Examples 3 and 4 demonstrate the characteristics discussed above.

Example 3 *Construct A as a 20 by 20 matrix with only diagonal and lower subdiagonal elements, all subdiagonal elements are 1, and all diagonal elements are random generated by RAND of MATLAB. Let $B = (1, 0, \cdots, 0)'$. The elements of A_i is the same as A's except $A_i(i, i) = 2$ so that A_i has only one unstable eigenvalue. Chandrasekhar algorithm which will be discussed in later sections is used to stabilize system (A_i, B). Table 2.1 shows the norms of feedbacks and the eigen-condition numbers of closed-loop with respect to i, the depth of unstable eigenvalue of A_i.*

Table 2.1. Feedback norms and eigencondition numbers for Example 3

i	1	5	10	15	20
$\|F\|_2$	148.0	410.0	3054.0	18229	80058
κ_2	4.24×10^7	7.37×10^7	1.26×10^9	1.60×10^{10}	6.56×10^{12}

Example 4 *If A is chosen such that all of A's diagonal elements are 0.5 and all lower subdiagonal elements are 1 and construct A_i and B the same way as before and assign the unstable eigenvalue of A_i to 0.9, direct computing gives the corresponding feedback $F = 1.1(1, 1.5, \cdots, (1.5)^{i-1}, 0, \cdots, 0)$. In either case, the norm of F increases drastically as i increases (or the unstable eigenvalue appears at the end).*

2.8 Summary and Future Work on Conditioning

In this chapter we have presented some of the basic theory on the stabilization and eigenvalue assignment via feedback as well as new contributions to the understanding of conditioning of those problems.

We have seen that $\|\Delta F\|/\|A\|$ and $\|\Delta F\|/\|F\|$ is not an acceptable predictor of computational difficulties for stabilization or eigenvalue assignment.

The eigen-conditioning of the closed-loop system was proposed as an alternative. The condition number $\kappa_2(X)$ was defined and bounded using results due to Van Dooren, the Bauer-Fike Theorem, and the Schur decomposition. It was demonstrated that the Schur decomposition for could be the most accurate. In general, $\kappa_2(X)$ was seen to be a conservative a posteriori analysis metric indicated the source of difficulty for a particular feedback F. As a result, it was of no use when the problem was so difficult as to not permit the production of a feedback.

An alternative that does not depend on a particular F was proposed in Section 2.4. The condition number was defined in terms of the minimum singular vectors of matrices defined by the spectrum desired in the closed-loop matrix. As a result, the computational complexity of this method is more significant than that based on $\kappa_2(X)$. It does however provide a bound on that condition number. A sampling method was proposed to compute the a priori condition estimator and experience has indicated it is a promising approach deserving of further investigation.

Finally, the robustness of the stabilization for a particular F was tested by defining a stability radius for the closed-loop matrix related to structured perturbations, i.e., changes to F rather than to the entire closed-loop matrix as in Bauer-Fike and Schur. The form of the bound was developed following work of Van Dooren. It allows the comparison of different feedbacks and provides a rigorous metric for preferring one to another. A sampling method was described to approximate the bound and discrete-time system or imaginary axis for continuous-time system. in that form, we see that it may be possible to integrate such a scheme into the algorithms that produce F in order to bias the iterations toward an F with a reasonable stability radius.

Finally, we considered single-input systems with a symmetric A or in Hessenberg control form. The former resulted in generalizations of results by [2] and their application to the stabilization problem. The latter indicated two key factors in the stabilization and eigenvalue assignment problem that may have applications in stabilization of large sparse systems.

CHAPTER 3

DISCRETE RICCATI EQUATIONS

In this chapter, we summarize some results in the literature for discrete Riccati equation and discrete Riccati difference equation that form the theoretical basis upon which we construct low-rank stabilization algorithms in later chapters. These results appear in references [20], [12] and [21].

3.1 Introduction

Consider the following (control form) discrete Riccati equation (DRE) and discrete Riccati difference equation (DRDE):

$$P = A'PA - A'PB(R + B'PB)^{-1}B'PA + Q \tag{3.1}$$

$$P_{i+1} = A'P_iA - A'P_iB(R + B'P_iB)^{-1}B'P_iA + Q \tag{3.2}$$

where A, P, P_i, B, R, Q are $n \times n$, $n \times n$,$n \times n$, $n \times p$,$p \times p$, and $n \times n$ real matrices respectively. R is positive definite, Q is semidefinite positive and usually is decomposed as CC' where C is an $n \times l$ real matrix. The non-negative solutions P and $\lim_{i\to\infty} P_i$ are of interest for stabilization. A feedback $F = (R + B'PB)^{-1}B'PA$ is associated with each P. P and F are the stabilizing solution of DRE and the stabilizing feedback respectively if $A - BF$ is stable.

In this chapter, we discuss only discrete-time systems. Most of results for the DRE are also true for the continuous-time Riccati equation (CRE) that has the form of:

$$A'P + PA - PBR^{-1}B'P + Q = 0 \tag{3.3}$$

where the continuous-time feedback is given by $F = R^{-1}B'P$.

3.2 Convergence of DRDE

The existence and uniqueness of solutions of the DRE and the convergence of the DRDE is addressed by Theorem 3.1([21]).

Theorem 3.1 *If (A, B) and (A', C) are stabilizable there exists a unique solution P to the DRE. For any initial condition $P_0 \geq 0$, the sequence P_i generated by DRDE will converge to P geometrically. Furthermore, $A - BF$ is stable where $F = (R + B'PB)^{-1}B'PA$.*

For the purpose of stabilization, only the parameters A and B in the DRE and DRDE are fixed. We have the freedom to choose $C(Q)$, R and P_0. This freedom is crucial since the conditions in Theorem 3.1 are too restrictive to support the development of low-rank algorithms for large sparse systems. The rest of this chapter explores the features of the DRE and DRDE and relaxation of the conditions in Theorem 3.1 to form the basis for the construction of low-rank stabilization algorithms in later chapters.

We start with Proposition 3.1 which states a well-known result [21]. However, we provide a new proof to gain insight into the stabilizability of the system. Proposition 3.1 holds for both discrete-time and continuous-time systems.

Proposition 3.1 *The following two statements are equivalent:*

1. *The system (A, B) is stabilizable.*

2. *Every unstable left eigenvector e of A is controllable, i.e., $e'B \neq 0$.*

Proof. First assume (A, B) is stabilizable. Let e be any unstable left eigenvector of A with eigenvalue λ. If $e'B = 0$, then $e'(A - BF) = e'A = \lambda e'$. So, λ is still unstable eigenvalue of A.

In the other direction, assume that $e'B \neq 0$ for any unstable left eigenvector e of A. For simplicity, we only prove single input case using the deflation method introduced in Chapter 2. Assume A has unstable eigenvalues λ_i, $i = 1, \cdots, k$ and corresponding left eigenvectors v_i, $i = 1, \cdots, k$. If $B'v_i \neq 0$, we can move λ_i and all other eigenvalues remain unchanged. For example, we move λ_1 by σ with $F = (\lambda_1 - \sigma)v_1'/(B'v_1)$ and keep $\lambda_2, \cdots, \lambda_k$. The corresponding left eigenvectors change to $v_1, v_2 - \alpha_2 v_1, \cdots, v_k - \alpha_k v_1$ with $\alpha_k = ((\lambda_1 - \sigma)B'v_i)/((\lambda_i - \sigma)B'v_1)$. So, for $i > 1$, we have

$$B'(v_i - \alpha_i v_1) = B'v_i \frac{\lambda_i - \lambda_1}{\lambda_i - \sigma}.$$

This vanishes only when $\lambda_i = \lambda_1$, i.e., where λ_1 is multiple eigenvalue. If there are linear independent eigenvectors for a multiple eigenvalue it is always possible to find a linear combination (and therefore another eigenvector) such that $B'v = 0$. This violates the assumption the case of multiple eigenvalues is excluded (and is in fact unstabilizable). With $B'(v_2 - \alpha_2 v_1) \neq 0$, we can move λ_2 into the stable region. By induction, we can move all other unstable eigenvalues into stable region.

For multiple-input systems, we only need to modify the deflation algorithm of Chapter 2 accordingly. ■

In this thesis, matrix relation $A \geq B$ means $A - B$ is positive semidefinite and $A > B$ means $A - B$ is positive definite. The stabilizability of (A, B) is a sufficient condition for the boundedness of the sequence P_i generated by DRDE.

Theorem 3.2 *[20] If (A, B) is stabilizable and P_0 is positive semidefinite, then the sequence P_i is uniformly bounded, i.e., there exists U such that* $0 \leq P_i \leq U$, $i \geq 0$

The stabilizability of (A, B) is not a necessary condition for the P_i to be bounded. For example, the P_i are bounded if C and P_0 do not contain the components of unstabilizable eigenvectors of A. The stabilizability of (A, B), however, cannot be relaxed if we want a stabilizing solution from the DRDE. If there exists a left unstable eigenvector e of A (Proposition 3.1) with eigenvalue λ such that $e'B = 0$ but $e'C \neq 0$, the P_i are unbounded.

For example, assume e and λ are real and $C = e$ and $P_0 = 0$, then $P_i = (1 + \lambda^2 + \cdots + \lambda^{2i})ee'$. If both $e'B$ and $e'C$ are 0, it is unlikely that the feedback F from P_∞ can stabilize λ. On the other hand, the stabilizability condition (A, B) is the only condition we need to impose on (A, B). Since our objective of this thesis is stabilization, we assume the stabilizability of (A, B) in this thesis.

In [20], Caines and Mayne derive certain monotonic properties of the DRDE. These monotonic properties form the foundation of the proof of convergence of DRDE in [20] and the more recent paper [12]. Here we discuss a monotonic property when $P_0 = 0$, that is the foundation of Chandrasekhar algorithm of Chapter 6. While the original proof in [20] is more general than that given below and helpful as a framework to find the general convergent conditions about the DRDE, Equation (3.5) (from [22]) provides a simple and clear proof. Furthermore, it is shown in Equation (3.5) that the rank of $P_{i+1} - P_i$ is nonincreasing which is critical to construction of low-rank stabilization algorithms in this thesis.

Theorem 3.3 *[20] If $P_0 = 0$, the sequence P_i generated by the DRDE increases monotonically. If (A, B) is stabilizable, $P_i \to P$ where P is a solution of the DRE.*

Proof. Let

$$\delta P_i = P_{i+1} - P_i, \; F_i = (R + B'P_iB)^{-1}B'P_iA, \tag{3.4}$$

and for arbitrary $P_0 \geq 0$ we have the recurrence,

$$\delta P_{i+1} = (A - BF_{i+1})'(\delta P_i + \delta P_i B(R + B'P_iB)^{-1}B'\delta P_i)(A - BF_{i+1}) \quad (3.5)$$

$$= (A - BF_i)'(\delta P_i - \delta P_i B(R + B'P_{i+1}B)^{-1}B'\delta P_i)(A - BF_i) \quad (3.6)$$

As we know, all $P_i \geq 0$ if $P_0 = 0$, so $R + B'P_iB > 0$. With $\delta P_0 = P_1 - P_0 = Q \geq 0$ and induction, $\delta P_i + \delta P_i B(R + B'P_iB)^{-1}B'\delta P_i \geq 0$, so does δP_i. If (A, B) is stabilizable, the convergence of P_i is derived from the uniformly bounded property from Theorem 3.2. ■

Note that the proof of Theorem 3.3 only needs $\delta P_0 \geq 0$. So, we can extend Theorem 3.3 to

Corollary 3.2 *If $\delta P_0 \geq 0$, P_i generated by DRDE increases monotonically. If (A, B) is stabilizable, $P_i \to P$ where P is a solution of DRE.*

Theorem 3.1 requires the stabilizability of (A', C) to guarantee the existence and uniqueness of the stabilizing solution. This condition can be relaxed but, more constraints on the initial guess P_0 are needed to achieve a stabilizing solution. For some special cases, such as Chandrasekhar algorithm which uses Theorem 3.3 and its corollary where initial guess $P_0 = 0$ or $\delta P_0 \geq 0$, the stabilizability of (A', C) is necessary for a stabilizing solution and its equivalent condition in Proposition 3.1 could be a guide to choose C. For example, if A has an unstable eigenvalue with multiple eigenvectors, the dimension of C should not be less than this multiplicity.

3.3 The DRE and Generalized Eigensolvers

In the following theorems, the phrase "a positive semidefinite P is a strong solution of the DRE" means that all eigenvalues of corresponding $A - BF$ with $F = (R + B'PB)^{-1}B'PA$ are inside or on the unit circle. Theorem 3.4 lists the most recent results about the DRE from [12]. Similar results about the continuous-time Riccati equation are proved in the later paper [23].

Theorem 3.4 ***A)*** *The strong solution of the DRE exists and is unique iff (A, B) is stabilizable.*

B) *The strong solution is the only nonnegative definite solution of the DRE iff (A, B) is stabilizable and (A', C) has no unreachable mode outside the unit circle.*

C) *The strong solution coincides with the stabilizing solution iff (A, B) is stabilizable and (A', C) has no unreachable modes on the unit circle.*

D) *The stabilizing solution is positive definite iff (A, B) is stabilizable and (A', C) has no unreachable mode inside or on the unit circle.*

Theorem 3.5 *[12] If either $P_0 > 0$ or $P_0 \geq P$ where P is the strong solution of the DRE, then the stabilizability of (A, B) and the nonexistence of unreachable modes of (A', C) on the unit circle are necessary and sufficient conditions for*

$$\lim_{i \to \infty} P_i = P$$

(exponentially fast), where P is the unique stabilizing solution of the DRE.

While Theorem 3.5 is the most recent and most general result about the initial condition of the DRDE, the conditions on P_0 in this theorem are too restrictive. First, a full rank P_0 is not useful when constructing a low-rank algorithm for large scale systems. Second, it is very difficult to find a P_0 such that $P_0 - P$ is nonnegative given that P is not known. The reason that these conditions are imposed on P_0 is that proof becomes easy with the help from some analytical methods, monotonic properties and maximum properties among different DRDEs introduced in [12].

We have seen that if (A, B) is stabilizable, starting from $P_0 = 0$, the P_i of the DRDE will be non-decreasing and bounded, thus convergent. Later in Chapter 5, we will further relax the condition on P_0 for the DRDE with $Q = 0$ so that we can build a low-rank algorithm to solve this special DRDE.

Besides the DRDE, another way to solve DRE is to solve its equivalent generalized eigenvalue problem [24]

$$Mz = \lambda Lz \tag{3.7}$$

with

$$M = \begin{pmatrix} A & 0 \\ -Q & I \end{pmatrix} \text{ and } L = \begin{pmatrix} I & BR^{-1}B' \\ 0 & A' \end{pmatrix}.$$

We need only solve the stable part of above generalized eigenvalue problem

$$MU = LUS \tag{3.8}$$

where $U = \begin{pmatrix} U_1 \\ U_2 \end{pmatrix}$ is $2n \times n$ and S is the $n \times n$ "Jordan canonical form" corresponding to all λ_i with $|\lambda_i| < 1$. More generally, S can be any real matrix which has all these λ as its eigenvalues and U can be any matrix constructed from the corresponding stable eigenvectors, such as Schur vectors. This generalized eigenvalue method cannot be used to solve a large scale DRE since all n eigenvectors are required. The form of Equation 3.7 is very useful when used to solve a small DRE or when used to analyze the properties of the DRE, e.g., to prove Theorem 3.4. Some researchers prefer the Schur method to compute U ([25]) but, the numerical advantage from the Schur method is not so convincing for ill-conditioned problems since the Schur method only orthogonalizes U, not its row partitions U_1 and U_2. It is the conditioning of U_1 and U_2 affects the accuracy of the computed solution of DRE $P = U_2U_1^{-1}$. The solutions from generalized eigenvalue problem can be summarized as follows.

Theorem 3.6 ***A)*** *All solutions of the DRE are of the form* $P = V_2V_1^{-1}$, *where* $\begin{pmatrix} V_1 \\ V_2 \end{pmatrix}$ *is a full rank* $2n \times n$ *matrix whose columns are linear combinations of* n *generalized eigenvectors and generalized principal vectors of (3.7).*

B) *If the columns of* U *are the general eigenvectors or the general principal vectors of (3.7) and* U_1 *is invertible where* $U' = (U_1', U_2')$ *then* $P = U_2U_1^{-1}$ *is a solution of* *the DRE with $P = P' \geq 0$ and U_1 is the right eigenvector matrix of closed-loop matrix $A - BF$.*

C) *Let P be a positive semidefinite solution to the DRE, then*

$$\begin{aligned}\Lambda(S) &= \textit{spectrum of closed loop } A - BF \\ &= \Lambda(A - B(R + B'PB)^{-1}B'PA) \\ &= \Lambda(A - BR^{-1}B'(A')^{-1}(P - Q)) \textit{ if } A \textit{ is invertible.}\end{aligned}$$

Proof. The proof can be extracted from [26]. ■

When using a generalized eigenvalue solver to solve the DRE, we need to construct U_1 and U_2 from its n stable eigen(principal) vectors. What will happen if any error is introduced? Assume $\tilde{U_1} = U_1 + E_1$, $\tilde{U_2} = U_2 + E_2$ and $\tilde{P} = \tilde{U_2}\tilde{U_1}^{-1}$. With the following result from [27]

$$(A + E)^{-1} = A^{-1} + \Delta_1 = A^{-1} - A^{-1}EA^{-1} + \Delta_2 \tag{3.9}$$

where $\delta = \|E\|_2 \ll \sigma_{min}(A)$, $\|\Delta_1\|_2 \leq \delta/\sigma_{min}(\sigma_{min} - \delta)$ and $\|\Delta_2\|_2 \leq \delta^2/\sigma^2_{min}(\sigma_{min} - \delta)$, we have,

$$\begin{aligned}\tilde{P} - P &= \tilde{U_2}\tilde{U_1}^{-1} - U_2U_1^{-1} \\ &= (U_2 + E_2)(U_1 + E_1)^{-1} - U_2U_1^{-1} \\ &= (U_2 + E_2)(U_1^{-1} - U_1^{-1}E_1U_1^{-1} + \Delta_2 - U_2U_1^{-1}) \\ &= -U_2U_1^{-1}E_1U_1^{-1} + U_2\Delta_2 + E_2U^{-1} - E_2U_1^{-1}E_1U_1^{-1} + E_2\Delta_2\end{aligned}$$

Assume $\|E_1\|_2 < \delta,$, $\|E_2\|_2 < \delta$, and throw away the second order errors, we have

$$\|\tilde{P} - P\|_2 \leq (\|P\|_2 + 1)\|U_1^{-1}\|_2\delta + O(\delta^2) \tag{3.10}$$

Let $\hat{P}$ be the computed solution([28]), and $R_P = \hat{P}\tilde{U}_1 - \tilde{U}_2$ then we have

$$\begin{aligned}\hat{P} - \tilde{P} &= (\hat{P}\tilde{U}_1 - \tilde{P}\tilde{U}_1)\tilde{U}_1^{-1} \\ &= (\hat{P}\tilde{U}_1 - \tilde{U}_2)\tilde{U}_1^{-1} \\ &= R_P\tilde{U}_1^{-1}.\end{aligned}$$

Therefore, the numerical accuracy of a solution P to Discrete Riccati Equation with generalized eigenvalue solver depends on (1) the magnitude of the exact solution, (2) the magnitude of U_1^{-1} and its computed version $\tilde{U}_1^{-1}$ and (3) the conditioning of U_1 and $\tilde{U}_1$. In theory, we can scale B or R and Q by a constant such that $\|P\|$ is close to one, therefore the numerical difficulties to compute P will come from U_1 and $\tilde{U}_1$ – their conditioning dominate the accuracy of the computed solution P of DRE via general eigenvalue solver. It is not a surprise that the corresponding DRE and stabilization problem share the same difficulty when U_1 is poorly conditioned because of the relationships between U_1 and the eigenvectors of closed-loop matrix $A - BF$ (Theorem 3.6). Because of the dependence of the accuracy of computed P on U_1, the ability to improve the accuracy of computed P with the scaling techniques introduced in [28] is very limited for poorly conditioned problem where U_1 is ill-conditioned, whether we use the generalized eigenvalue solver or the DRDE.

3.4 Fixed Point View of the DRDE

The convergence of the DRDE is shown in Equation (3.5). We can also apply nonlinear fixed point theory to analyze the convergence of the DRDE since it is just the fixed point method applied to the DRE. Let $\mathcal{R}(X) = A'(X - XB(R + B'XB)^{-1}B'X)A + Q$ be a mapping from set of positive semidefinite matrices into set of positive semidefinite matrices. The DRE can be thought of as finding a fixed point of $\mathcal{R}$ and the DRDE is the iterative fixed point algorithm such as $P_{i+1} = \mathcal{R}(P_i)$. The first derivative $\mathcal{P}'(X_0)$ of a nonlinear mapping $\mathcal{P}$ at X_0 is defined as

$$\lim_{t \to 0} \frac{\mathcal{P}(X_0 + tX) - \mathcal{P}(X)}{t} = \mathcal{P}'(X_0)(X). \tag{3.11}$$

We also have the standard theorem [29]

Theorem 3.7 *Suppose the mapping $\mathcal{P}$ maps a set $\Omega \subset \mathcal{X}$ into $\mathcal{X}$, and that it has a derivative at every point of a convex closed subset $\Omega_0 \subset \Omega$. If*

1. $\mathcal{P}(\Omega_0) \subset \Omega$

2. $\sup_{X\in\Omega_0} \|\mathcal{P}'(X)\| = \alpha < 1$

then $\mathcal{P}$ *has a unique fixed point in* Ω_0.

With the definition of the first derivative of $\mathcal{P}$, for any $\epsilon > 0$, we can find a $\delta > 0$ such that, whenever $\|X_2 - X_1\| < \delta$, we have

$$\|\mathcal{P}(X_2) - \mathcal{P}(X_1) - \mathcal{P}'(X_1)(X_2 - X_1)\| \leq \|X_2 - X_1\| \tag{3.12}$$

When applied to a fixed point iteration $X_{i+1} = \mathcal{P}(X_i)$ with solution X, Equation (3.12) implies

Proposition 3.3

$$X_{i+1} - X_i = \mathcal{P}'(X_{i-1})(X_i - X_{i-1}) + o(X_i - X_{i-1}) \tag{3.13}$$

$$X_{i+1} - X = \mathcal{P}'(X)(X_i - X) + o(X_i - X) \tag{3.14}$$

Applying the definition of the first derivatives of a nonlinear equation on $\mathcal{R}$ defined for the DRE, we have

$$\mathcal{R}'(X)(S) = A'_X S A_X$$

where $A_X = A - BF_X$ and $F_X = (R + B'XB)^{-1}B'XA$. Equations (3.13) and (3.14) imply

$$P_{i+1} - P_i = A'_{P_{i-1}}(P_i - P_{i-1})A_{P_{i-1}} + o(P_i - P_{i-1}) \tag{3.15}$$

$$P_{i+1} - P = A'_P(P_i - P)A_P + o(P_i - P) \tag{3.16}$$

With suitable norm (spectral norm) on above two equations, we have

Proposition 3.4 *The sequence P_i of DRDE converges with the rate of $\rho^2(A - BF)$, where $\rho(X)$ denotes spectral radius of X and $F = (R + B'PB)^{-1}B'PA$ with P as a solution of the DRE.*

The convergence of feedback F_i can be obtained from the last part of Theorem 3.6 and the linear relation between F and P. A similar relation between F_i and P_i can be shown. These relations provide a new way to compute feedback F for the DRE and F_i for the DRDE without computing inverse of $R+B'PB$ or $R+B'P_iB$ respectively. The price is the computation of $F_A = R^{-1}B'(A')^{-1}$. When using DRDE, we can update the feedback with

$$F_{i+1} = F_i + F_A \delta P_i. \tag{3.17}$$

Note that this implies we need not have P or P_i to find the feedback. It can be computed from the increment produced by a recurrence. This is a useful relationship when constructing low-rank stabilization algorithms. We can expect F_i converges with the same speed as P_i.

3.5 Stopping Criterion for the DRDE

For any iteration, there is a very important question: when can we, or should we stop? Assume F is a desired feedback such that $A-BF$ is stable, the spectral radius of $A-BF$ is ρ, and the Euclidean norm eigen-condition number is $\kappa_2(X)$ where X is the eigenvector matrix of $A-BF$. Assume $\tilde{F}$ is a computed feedback from any iterative algorithm. The Bauer-Fike Theorem can be used to derive a sufficient condition that $A-B\tilde{F}$ is stable and which we can use as a stopping criterion.

Proposition 3.5 *A sufficient condition that $A-B\tilde{F}$ is stable is*

$$\|B(F-\tilde{F})\|_2 < (1-\rho)/\kappa_2(X). \tag{3.18}$$

For example, assume B is normalized, $\rho = 0.99$ and $\kappa_2(X) = 10^3$, we can use 10^{-5} as a stopping criterion. Since it is not likely that the eigen-condition number and spectral radius of closed-loop $A-BF$ are known in advance, we can use eigen-condition number and spectral radius of $A-BF_i$ as estimates when computing the right side of Equation (3.5).

CHAPTER 4

STABILIZATION VIA PROJECTION

In this chapter, we discuss the projection method – a low-rank algorithm to solve the eigenvalue assignment problem for large sparse dynamical systems where only a few of its eigenvalues must be assigned, proposed by Saad in [10]. The projection method as discussed in [10] is more of a family of algorithms that a single method in that there are many ways to implement the general strategy that Saad advocates. In this chapter, we discuss some options such as subspace iteration (SSI) and its acceleration via Stewart's refinement [11]. We also propose a novel algorithm which embeds a feedback-computing process on the reduced (projected) system into the subspace iteration so that the stability of corresponding closed-loop system can be caught as early as possible.

4.1 Projection Methods

When the system matrix A is dense and of small or moderate order, many algorithms are available to solve the eigenvalue assignment problem. These algorithms (e.g., [1],[3], [4], [5], [6], [7]) require at least $O(n^3)$ operations due to their dependence on matrix operations like a full-rank QR decomposition and transformations of the system to Schur form or Hessenberg control form. For large scale systems, as pointed out by Saad in [10], using these algorithms to solve the eigenvalue assignment problem is not only computational prohibitive, but also untrustworthy. Saad in [10] conjectures that assigning the eigenvalues of a system of order 50 may be the upper limit of what can be reliably achieved with standard double precision and these classical algorithms. On the other hand, for the stabilization problem where A only has a few unstable eigenvalues, it is only necessary to move those unstable eigenvalues into the stable region thereby reducing the complexity. The idea of only working with the unstable eigenvalues is the key idea when designing low-rank algorithms for large scale stabilization known as projection methods. For the purpose of large scale stabilization or the assignment of a few eigenvalues, Saad in [10] introduced the following general projection algorithm:

Algorithm 1 Partial Eigenvalue Assignment by Projection [10]

1. *Compute the k eigenvalues to be assigned and the associated partial Schur decomposition $A'Q = QR$ where Q is an $n \times k$ matrix whose columns form an orthonormal basis of the left invariant subspace associated with the k eigenvalues to be assigned and R is a $k \times k$ upper quasi-triangular matrix.*

2. *Solve the projected inverse eigenvalue problem, that is, to find f such that $R' - (Q'B)f$ has the expected eigenvalues.*

3. *Lift the feedback to the original space, $F = fQ'$.*

Although Algorithm 1 uses eigenvectors and Schur decomposition, Saad [10] points out that only an orthonormal basis Q of the invariant subspace is needed and R which represents A' in that basis is not necessarily quasi-triangular. When applied to large scale stabilization problem where A has only a few unstable eigenvalues, the above algorithm becomes:

Algorithm 2 Stabilization by Projection

1. *Compute a matrix Q_u whose columns form an orthonormal basis of left invariant subspace of A, which is a superspace of the left unstable subspace of A.*

2. *Project the system (A,B) onto Q_u and stabilize $(Q_u' A Q_u, Q_u' B)$, that is, to find a stabilizer f such that $Q_u' A Q_u - (Q_u' B) f$ is stable.*

3. *Lift f back to the original space to get the stabilizer $F = fQ'$ of the original system (A,B).*

The correctness of above partial eigenvalue assignment and stabilization algorithm via projection can be easily verified from the following Lemma:

Lemma 1 *Let $Q = [Q_s \ \ Q_u]$ be a real orthogonal matrix such that columns of Q_u form a superspace of left unstable invariant subspace of A and*

$$Q'AQ = \begin{bmatrix} Q_s' A Q_s & Q_s' A Q_u \\ Q_u' A Q_s & Q_u' A Q_u \end{bmatrix} = \begin{bmatrix} A_s & A_{12} \\ 0 & A_u \end{bmatrix}. \tag{4.1}$$

Let $Q'B = \begin{pmatrix} B_s \\ B_u \end{pmatrix}$ accordingly. If F_u stabilizes A_u with $A_u - B_u F_u$, then $F = (0 \ \ F_u)Q' = F_u Q_u'$ stabilizes A by $A - BF$.

The cost of Algorithm 1 or Algorithm 2 is dominated by the first step which computes a left invariant subspace of A containing desired eigenvalues. The efficiency of the algorithms depend on the method chosen to compute the invariant subspace basis. Some numerical methods to compute invariant subspace are summarized in [10].

4.2 Invariant Subspace Computation

To compute the invariant subspace, same as described in first step of Algorithm 1, a natural idea is to compute the corresponding left eigenvectors of A and form the orthonormal basis of the invariant subspace with Schur vectors extracted by Gram-Schmidt orthogonalization of these eigenvectors. The algorithms to compute invariant subspace in [30] and [31] need the Schur decomposition of A, so these algorithms also need to compute the eigenvectors.

There are two major classes of methods that can be used for large systems. The first computes the unstable eigenvalues and recovers their eigenvectors by some form of inverse iteration. The second class computes the basis directly by subspace iteration-like methods.

An example of the first class is Arnoldi process which is a projection process onto Krylov space $K_m = span(v_1, A'v_1, \cdots, (A')^{m-1}v_1)$, see Lehoucq and Sorensen [32] and the references for recent results. The invariant space can then be recovered from the eigenvalues via inverse iteration. Methods in the first class benefit from years of sparse eigenvalue algorithm research but often require very high accuracy in the eigenvalues in order to produce the basis and hence result in more computation than necessary for stabilization.

An example of the second class of methods is subspace iteration (SSI), or orthogonal iteration (see [11], [18] and [10]). The most general form of subspace iteration originated from Bauer's *Treppeniteration* to compute the (right) approximate invariant subspace and can be described as follows.

Algorithm 3 Subspace Iteration (SSI)*[11] [10]*

1. *Choose an initial orthonormal matrix* V_0.
2. *Generate a sequence of subspaces (orthonormal matrices)* V_i *from QR decomposition of* AV_{i-1}: $AV_{i-1} = V_i R_i$.

Assume A has eigenvalues $\lambda_1, \cdots, \lambda_n$ which are ordered so that $|\lambda_1| \geq |\lambda_2| \geq \cdots |\lambda_n|$. If the rank of V_0 is m, the convergence of V_i is determined by the ratio $|\lambda_{m+1}/\lambda_m|$ where the convergence is measured by the distance between the V_i and invariant subspace of A formed by eigenvectors with m largest eigenvalues ([18]). The distance between two subspaces S_1 and S_2 is defined by

$$dist(S_1, S_2) = \|P_1 - P_2\|_2$$

where P_i is the orthogonal projection onto S_i. (If V is an orthonormal basis of a subspace S, the unique orthogonal projection onto S is $P = VV'$).

For stabilization and partial eigenvalue assignment, we can compute the left invariant subspace of A with Algorithm 3 by replacing A with A'. We will not compute the above distance to measure convergence. Instead, for any orthogonal matrix $Q = [Q_1 \;\; Q_2]$ which decomposes A to

$$\begin{pmatrix} Q_1'AQ_1 & Q_1'AQ_2 \\ Q_2'AQ_1 & Q_2'AQ_2 \end{pmatrix},$$

we shall say that Q_2 has converged if the residual $Q_2'AQ_1$ is smaller than a prespecified tolerance. After simplifying $[Q_1 \quad Q_2]'(A'Q_2 - Q_2(Q_2'A'Q_2))$ to $[Q_1'A'Q_2 \;\; 0]'$ and $[Q_1 \quad Q_2]'(AQ_2 - Q_2(Q_2A'Q_2))$ to $[Q_1'AQ_2 \;\; 0]'$, we have $\|Q_2'AQ_1\|_2 = \|A'Q_2 - Q_2(Q_2'A'Q_2)\|_2$ and $\|Q_1'AQ_2\|_2 = \|AQ_2 - Q_2(Q_2'AQ_2)\|_2$. So, we can use the subspace residual $\|A'Q_2 - Q_2(Q_2'A'Q_2)\|_2$ to measure convergence. This is also used by Stewart in [11] and is similar to what is often used in Arnoldi process.

4.3 Conditioning

There are very few results on conditioning of subspace decomposition. The most general result is the following theorem [33]:

Theorem 4.1 *Let*

$$Q'AQ = \begin{pmatrix} T_{11} & T_{12} \\ 0 & T_{22} \end{pmatrix},$$

and $Q = [Q_1 \; Q_2]$. *Define*

$$sep(T_{11}, T_{22}) = min\{\|T_{11}X - XT_{22}\|_F : \; \|X\| = 1\}.$$

For any matrix E with partition as following:

$$Q'EQ = \begin{pmatrix} E_{11} & E_{12} \\ E_{21} & E_{22} \end{pmatrix}.$$

If $\delta = sep(T_{11}, T_{22}) - \|E_{11}\|_2 - \|E_{22}\|_2 > 0$ *and*

$$\|E_{21}\|_2(\|T_{12}\|_2 + \|E_{12}\|_2) < \delta^2/4,$$

then there exists a P *such that* $\|P\|_2 \leq 2\|E_{21}\|_2/\delta$, *and the columns of* $\tilde{Q} = (Q_1 + Q_2 P)(I + P'P)^{-1/2}$ *form an orthonormal basis for a subspace that is invariant for* $A + E$.

Theorem 4.1 indicates what could contribute to ill-conditioning and reduce the ability of algorithms to compute invariant subspace and eigenvalues: the separation between the spectrum of T_{11} and the spectrum of T_{22}, the magnitude of T_{12}. In order to get an accurate invariant subspace, we need to find a way to decompose the space into two well-separated submatrices. For stabilization purposes, it is possible that we cannot separate the unstable invariant subspace well enough, so, we need to find an invariant subspace as superspace of the unstable invariant subspace and this larger invariant subspace should be better separated from other parts. More importantly, the dimension of the larger subspace cannot be too large. Unless A is symmetric, usually $\|T_{12}\|_2$ is not zero, which could determine eigen-conditioning of A. If $\|T_{12}\|_2$ is very large, $\|E_{21}\|_2 < \delta^2/\|T_{12}\|_2$ has to be very small. For subspace iteration, we need to check both the subspace residual $\|AV_i - V_i(V_i'AV_i)\|_2$ and $\|A'V_i - V_i(V_i'A'V_i)\|_2$. The larger $\|A'V_i - V_i(V_i'A'V_i)\|_2$ is, the smaller the tolerance for the subspace residual $\|AV_i - V_i(V_i'AV_i)\|_2$ should be.

The Bauer-Fike Theorem (Theorem 2.3) can also be used to derive a useful result. Assume Q_u is an orthonormal basis of the approximated left invariant subspace of A containing unstable invariant subspace and $Q = [Q_s \;\; Q_u]$ is an orthogonal matrix. Let F_u be a stabilizing feedback of the projected system $(Q_u'AQ_u, Q_u'B)$. Let

$$\tilde{A} = \begin{bmatrix} Q_s'AQ_s & Q_s'AQ_u \\ 0 & Q_u'AQ_u \end{bmatrix}$$

and $F = F_u Q_u'$. It is very easy to verify that

$$\tilde{A}_F = \tilde{A} - [Q_s \;\; Q_u]'B(F[Q_s \;\; Q_u]) = \begin{bmatrix} Q_s'AQ_s & Q_s'AQ_u - Q_s'BF_u \\ 0 & Q_u'AQ_u - Q_u'BF_u \end{bmatrix}$$

is stable. The difference between $\tilde{A}_F$ and

$$Q'AQ - Q'BFQ = \left[\begin{array}{cc} Q_s'AQ_s & Q_s'AQ_u - Q_s'BF_u \\ Q_u'AQ_s & Q_u'AQ_u - Q_u'BF_u \end{array} \right]$$

is $Q_u'AQ_s$. So, the stability of $Q'AQ - Q'BFQ$ and its equivalent $A - BF$ depends on (1) $Q_uA'Q_s$, or its equivalent subspace residual $A'Q_u - Q_u(Q_u'A'Q_u)$, (2) the norm of $Q_s'AQ_u - Q_s'BF_u$ or its equivalent $AQ_u - Q_u(Q_u'AQ_u)$, (3) the eigen-condition number $\kappa_2(X, \tilde{A}_F)$ of $\tilde{A}_F$, and (4) the spectral radius $\rho(\tilde{A}_F)$ of $\tilde{A}_F$.

Assume we can estimate $\kappa_2(X, \tilde{A}_F)$ and $\rho(\tilde{A}_F)$. If $\|A'Q_u - Q_u(Q_u'A'Q_u)\|_2$ is within $(1 - \rho(\tilde{A}_F))/\kappa_2(X, \tilde{A}_F)$, $A - BF$ will be stable, which can be verified by Proposition 2.3. For example, if $\rho(\tilde{A}_F) = 0.99$ and $\kappa_2(X, \tilde{A}_F) = 10^3$, we have to choose subspace residual tolerance at least 10^{-5}.

For partial eigenvalue assignment and stabilization via projection, it is worthwhile to explore the relationship between the eigen-condition of corresponding reduced closed loop and original closed loop. Define the condition number of eigenvalue λ of A as $cond_A(\lambda) = 1/\cos(u, v)$ where u and v are the left and right eigenvector of A with eigenvalue λ. It is easy to verify the following relationship:

Proposition 4.1 *Let*

$$Q'AQ = \left(\begin{array}{cc} A_{11} & A_{12} \\ 0 & A_{22} \end{array} \right).$$

Assume $A_{22} - B_2F_2$ and A_{11} do not have common eigenvalues and let $F = [0 \;\; F_2]\, Q'$. For any eigenvalue λ of $A - BF$:

(1) If λ is an eigenvalue of $A_{22} - B_2F_2$ and u_2, v_2 are the left and right eigenvectors, then

$$cond_{A-BF}(\lambda) = cond_{A_{22}-B_2F_2}(\lambda)\sqrt{v_2'v_2 + v_1'v_1}/\|v_2\|_2 \tag{4.2}$$

where $v_1 = (A_{11} - \lambda)^{-1}(B_1F_2 - A_{12})v_2$.

(2) If λ *is an eigenvalue of* A_{11} *and* v_1, u_1 *are the left and right eigenvectors, then*

$$cond_{A-BF}(\lambda) = cond_{A_{11}}(\lambda)\sqrt{u_1'u_1 + u_2'u_2}/\|u_1\|_2 \tag{4.3}$$

where $u_2 = ((A_{22} - B_2F_2)' - \lambda)^{-1}(B_1F_2 - A_{12})'u_1$.

This proposition shows that the eigen-condition of corresponding closed loop of system (A, B) could be much worse than the eigen-condition of the closed loop of reduced system if (1) some eigenvalues of reduced system are too close to the rest eigenvalues of the system, or (2) B_1 are much larger than B_2 in magnitude.

4.4 Discrete-Time Subspace Iteration

While the idea of stabilization via projection can be used to stabilize both discrete-time and continuous-time systems, usually, the eigenvalue solver or SSI described in Algorithm 3 produces eigenvalues with largest magnitude, that can only be used for a discrete-time system. For a continuous-time system, we need to apply a conformal transformation to A, which will be discussed at the end of this chapter and in Chapter 8. The general algorithm to stabilize a discrete-time system via projection and SSI is summarized as follows.

Algorithm 4 Stabilization of discrete-time system via projection and SSI

1. *Randomly choose a vector* v, *use the Arnoldi method on with* (A', v),*to get an* $n \times m$ *orthonormal matrix* V_0. *Here* m *is a pre-specified integer which could be slightly larger than the number of unstable eigenvalues of* A.

2. *For* i *from 1 to some specified number* N,
   compute the QR decomposition $A'V_{i-1} = V_iR_i$

Compute $\|A'V_i - V_i(V_i'A'V_i)\|_2$.

If $\|A'V_i - V_i(V_i'A'V_i)\|_2$ *reaches prespecified tolerance, stop, else continue.*

If tolerance is not reached after $i = N$*, go back to step 1 and restart with a new* v *taken as a combination of* $A'V_i - V_i(V_i'A'V_i)$.

3. *Compute* $A_u = V_i'AV_i$ *and* $B_u = V_i'B$*, and stabilize* (A_u, B_u) *to get stabilizable feedback* F_u.

4. *Lift feedback to* $F = F_uV_i'$.

As previously mentioned, the major cost of stabilization algorithm via projection is computing the unstable invariant subspace of A. The advantage of Algorithm 4 for large sparse A is that the computation of invariant subspace with SSI only involves matrix multiplication of form $A'V$ and a low-order QR decomposition. The disadvantage also comes from subspace iteration. We need to find m, the number of unstable eigenvalues of A. Without any prior knowledge, this can be an error prone process.

The convergence of SSI is restricted by $|\lambda_m/\lambda_{m+1}|$. In case there exists any unstable eigenvalues which cannot be separated from stable eigenvalues, the ratio $|\lambda_m/\lambda_{m+1}|$ will be close to 1 and slow convergence is expected. Even if the stable and unstable space are well-separated, a wrong guess of m could be still costly. For example, if the ratio $|\lambda_m/\lambda_{m+1}|$ is 1 or very close to 1, SSI will have trouble converging. Of course, even if SSI as a whole has trouble converging, some eigenvalues/eigenvectors could still converge with the acceptable faster rate and those eigenvectors can be extracted with Schur-Rayleigh-Ritz (SRR) refinement proposed by G. W. Stewart as a method to accelerate SSI. During SSI iteration, the convergence of the subspace spanned by first r largest eigenvectors in V_i is $O(|\lambda_{m+1}|/|\lambda_r|)$. If $r \neq m$, even if m is guessed wrong, we can still extract the subspace spanned by first r eigenvectors.

Algorithm 5 SRR refinement*[11]*

In step i of SSI (see Algorithm 3 or Algorithm 4),

1. *Compute the Schur decomposition of $H_i = V_i' A V_i$ such that $T_i = Y_i' H_i Y_i$ such that the diagonal elements of T_i appear in descending order of absolute value. Compute the eigenvalues of H_i, as $\mu_1, \cdots, \mu_m$ in descending order of absolute value.*

2. *For some k such that $k \leq m$ and $|\mu_k| > |\mu_{k+1}|$, take $Y_i^{(k)}$ as the first k columns of Y_i, and compute $V_i^k = V_i Y_i^k$.*

3. *Compute the residual $\|AV_i^k - V_i^k((V_i^k)' AV_i^k)\|_2$.*

Depending on the purpose of SRR, k can be chosen with different methods. For example, if we want to compute l largest eigenvalues of A, we only need to choose $k = l$, take first l columns of Y_i to form V_i^k and compute eigenvalues of $(V_i^k)' AV_i^k$ only when residual from V_i^k is small enough (note we do not need to compute eigenvalues of T_i in the first step). For stabilization purposes, r should be taken as the number of unstable eigenvalues of T_i (with A' replacing A). The computational cost of SRR (except the residual) is equivalent to one step of SSI plus solving small eigenvalue problem and Schur decomposition, which means the cost is at least doubled if we perform SRR on every step of SSI. Of course, SRR is usually not needed to use on each step of SSI and some savings can be realized by exploiting this fact.

4.5 The Hybrid Subspace Iteration Method

Whatever method we use to compute invariant subspace, since we usually can only get an approximate subspace, we need to determine the accuracy tolerance of the subspace residual or estimated eigenvalues used to terminate the iteration. In Chapters 5 and 6 we observe that stabilization usually occurs far earlier than any convergence measurements for the eigenvalues or subspaces. Hence, the two-phase form of the algorithms – find the basic and stabilize the projected system – is less than optimal computationally.

We, therefore, propose a stabilization algorithm that embeds a low-order stablization algorithm in some steps of SSI with SRR of the following form

Algorithm 6 SSI + SRR + Feedback Computation

1. *Randomly choose a vector v, use Arnoldi method on (A', v) to get an $n \times m$ orthonormal matrix V_0. Here m is a pre-specified integer which should be at least the number of unstable eigenvalues of A.*

2. *For i from 1 to some specified number N,*

   (a) *Compute the QR decomposition to update V_i: $A'V_{i-1} = V_iR_i$.*

   (b) *In any of a set of specified steps, run Stewart's SRR refinement Algorithm 5 to extract the approximate unstable invariant subspace Q_i from V_i, or more generally, form an approximate subspace Q_i from V_i and Schur vectors of $V_i'A'V_i$ with the eigenvalues of $V_i'A'V_i$ outside a circle of radius r (for example, $r = 0.9$).*

   (c) *Following the above SRR step, compute a stabilizing feedback F_u of the reduced system $(Q_i'AQ_i, Q_i'B)$ and lift back to feedback $F_i = F_uQ_i'$ in the original space. If some stopping criterion (discussed below) reaches a specified tolerance, stop.*

The efficiency of Algorithm 6 depends on the choice of the stabilization algorithm used for the reduced system and the convergence/stabilization criterion. One choice is to build a DRE with parameters $(Q_i'AQ_i, Q_i'B)$ and to solve it with a generalized eigenvalue method (see Chapter 3). If $Q_i'AQ_i$ has eigenvalues close to unit circle, we should scale $Q_i'AQ_i$ (see Chapters 6 and 5 for scaling techniques). The choice of stopping criterion needs further research. Here we can use feedback difference

$F_i - F_{i-1}$ or $(A - BF_i) \cdots (A - BF_1)B$. The latter $(A - BF_i) \cdots (A - BF_1)B$ only works when spectral radius of $A - BF_i$ is not close to 1. It is worth pointing out that the algorithm and parameters used should be consistent with whatever reduced order stabilization algorithm is chosen. For example, if we choose the DRE to compute the reduced order feedback, we need to specify global R and Q and project them onto Q_i to form the reduced order DRE.

4.6 Experimental Results

Example 5 shows the efficiency of Algorithm 6. The computation of SRR and feedback via reduced system in one step of iteration is costly, so we only perform them occasionally. However, when they are performed they are so done in at least two consecutive steps so we can calculate the feedback increment to check convergence.

Example 5 *A is a 100×100 matrix generated by RANDN of MATLAB and is scaled. The spectrum is displayed in Figure 5.7. B and V are generated by RAND as random vectors. The rank of SSI, m, is chosen as 4, 10 and 20 for each experiment. Algorithm 6 was run for 400 steps. Here, in each step of iteration, we extract the unstable subspace Q_i, compute subspace residual $A'Q_i - Q_i(Q_i'A'Q_i)$, build the DRE with $Q_i'AQ_i/0.9$, $Q_i'B/0.9$, $R = I$ and $Q = 0$, compute the feedback F_i from the DRE and record both the feedback increment and $(A - BF_i) \cdots (A - BF_1)B$. Figure 4.1 shows that as the rank of SSI increases, the number of iterations stabilization reduces significantly. Rank 20 SSI/SRR stabilize the system in 30 step (equivalent to 150 steps of rank 4 SSI/SRR, ignore the computation of stopping criterion) while rank 4 SSI/SRR uses 310 steps. Figure 4.2, Figure 4.3 and Figure 4.4, show the three convergence statistics as a function of iteration number. The rank 20 SSI/SRR converges much faster than rank 10 and rank 10 SSI/SRR converges much more faster than rank 4 SSI/SRR in measure of computation costs. This example verifies that SSI with SRR does benefit from enlarging rank of initial subspace.*

While Example 5 shows that SRR will benefit from higher rank SSI, embedding SRR and a stabilization process on reduced system in each step of iteration is very costly. Fortunately, extracting unstable subspace and computing feedback at step i is independent of previous computations of unstable subspace and feedback. So, we only need to use SRR and compute feedback when we expect the approximated unstable invariant subspace from SSI, or a subspace of the unstable invariant subspace if a deflation method is used, has converged to some accuracy. Stabilization via projection with SSI has another advantage: we will gain information about the eigenvalue distribution of A and its reduced system A_u, which will give us clues how to stabilize the reduced system. For example, if some eigenvalues are too close to unit circle, we should drag them away from unit circle a little farther. If we want to embed a small size stabilization for reduced system, that means, we will compute feedback more frequently than usual projection algorithm which compute feedback only after invariant subspace is computed, iterative algorithm is not recommended for small size stabilization.

Example 6 illustrates a potential problem when using SRR and how we can solve it.

Example 6 *Let*

$$A = \begin{pmatrix} 2 & 0 & 0 \\ 0 & 1 & -6 \\ 0 & 0 & -1 \end{pmatrix}$$

and start from

$$Q_0 = \begin{pmatrix} 1 & 0 \\ 1 & 1 \\ 1 & -1 \end{pmatrix}.$$

After 100 iterations of SSI with $m = 2$ *while trying to estimate the dominant eigenvalue/vector pair, we get*

$$Q_{100} = \begin{pmatrix} 1 & 0 \\ 0 & 0.7071 \\ 0 & -0.7071 \end{pmatrix}, \quad Q'_{100}AQ_{100} = \begin{pmatrix} 2 & 0 \\ 0 & 3 \end{pmatrix}.$$

Therefore, applying the SRR method to extract the dominant eigenvector yields $(0, 0.7071, -0.7071)$*, which is not correct. This error arises from* $|\lambda_2|/|\lambda_3| = 1$*. If we check the subspace residual for* $Q_{100}^{(1)} = (0, 0.7071, -0.7071)'$*, we have* $\|AQ_{100}^{(1)} - Q_{100}^{(1)}((Q_{100}^{(1)})'AQ_{100}^{(1)})\|_2 = 4.0$ *which clearly indicates a problem and that the eigenvector should not be trusted.*

The solution to this is to use more than one SRR result to determine the dominant eigenvector(s), i.e., to decide what $l < m$ *should be used as a basis. (Recall that* m *is assumed to be an overestimate.). After one more SSI step we get*

$$Q_{101} = \begin{pmatrix} 1 & 0 \\ 0 & 0.9899 \\ 0 & 0.1414 \end{pmatrix}, \quad Q'_{101}AQ_{101} = \begin{pmatrix} 2 & 0 \\ 0 & 0.12 \end{pmatrix}.$$

The estimate of eigenvalue 2 is common to both SSR analyses while the estimate of the second eigenvalue changes drastically. Identifying the intersection of the spectrums of two or more successive SRR analysis tends to provide a reliable estimate of l *and the associated eigenvalue/vector pairs. Indeed, the second Schur decomposition gives us desired order and the subspace residual* $\|AQ_{101}^{(1)} - Q_{101}^{(1)}((Q_{101}^{(1)})'AQ_{101}^{(1)})\|_2$ *is at machine precision.*

4.7 Continuous-Time Systems

Projection algorithms can be used to stabilize both discrete time and continuous time systems. We, however, have only discussed algorithms to compute approximate discrete time unstable invariant subspaces. To stabilize a continuous time system, we need to transform to a discrete time system, i.e., transform A into a matrix $\tilde{A}$ such that the eigenvalues of A in right half plane are transformed just to outside of unit circle. To compute the continuous time unstable invariant subspace, Saad proposes a set of algorithms, based on polynomial iterations such as Chebyshev iteration (see [10] and its references). The disadvantage of these algorithms is the need for information about the distribution of the eigenvalues of A.

Another transformation choice is $\tilde{A} = (I - \alpha A)^{-1}(I + \alpha A)$ which, in fact, is the basis of algorithms which compute continuous-time unstable eigenvalues. For example, in a recent paper [34] to compute rightmost eigenvalues, Meerbergen and Roose use the Arnoldi process and an approximation M_c of $(I - \alpha A)^{-1}$ to compute the largest eigenvalues of $\tilde{A}$. Other algorithms to compute continuous-time invariant subspace include the sign function method (see [35], [8] and their references for details). Its applicability to large sparse systems is still open. Algorithms to stabilize continuous-time system are discussed in Chapter 8.

4.8 Summary

In this chapter, we have discussed the family of projection methods to stabilize large sparse systems with a few unstable eigenvalues. Projection algorithms need to compute an invariant unstable subspace, that can be computed directly from eigenvalue/eigenvector algorithms, as Saad used in his examples in [10], or computed from SSI. The main disadvantage of both methods to compute invariant subspace is the difficulty in determing the accuracy tolerance required. In SSI, larger eigenvalues/eigenvectors converge first and faster, but the convergence as a whole is determined by the worst case. If we filter out (deflate) the eigenvalues/eigenvectors which have converged and concentrate on the tough ones, the total computations could be reduced significantly, especially if only very few (for example 1 or 2) eigenvalues are hard to separate from stable ones. Further research to adapt recent techniques from eigenvalue solvers to accelerate subspace iteration is needed.

In this chapter, we also tried to combine SRR and embed a low-order stabilization algorithm to compute feedback during the iteration. While this SSI/SRR/Feedback algorithm can catch stabilization much earlier and can benefit from enlarging rank of SSI, the computation of SRR and feedback in each step of iteration is very costly. In Chapter 5, we develop an alternative method that combines SSI and feedback computing much more naturally with a computation that insignificantly larger than SSI without SRR and feedback computation.

Another problem for projection methods is that the rank of computation is at least the number of unstable eigenvalues. Since this translates directly into computations and space (due to the need to store a basis) the methods do not scale well beyond a very few unstable eigenvalues for large system order. In Chapter 6, we will see a true low-rank stabilization algorithm whose rank is independent of the number of unstable eigenvalues.

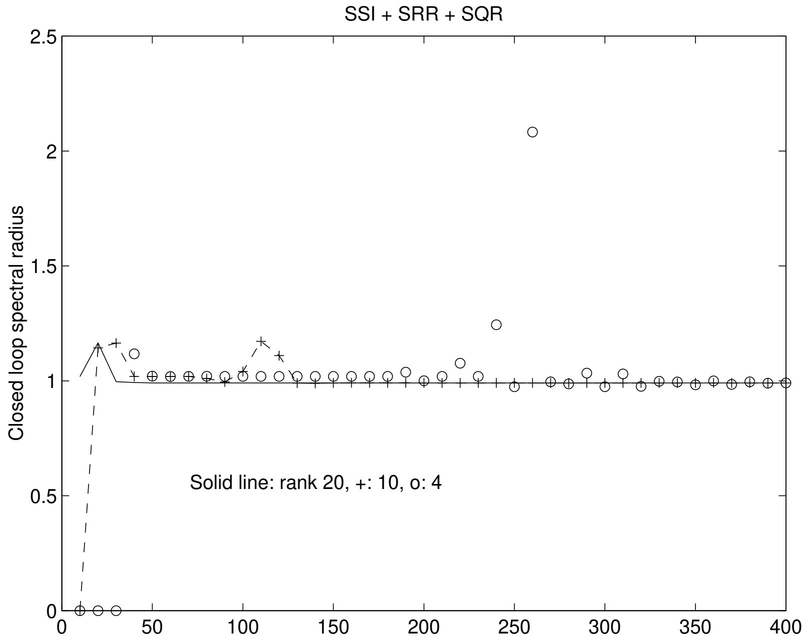

Figure 4.1. SSI+SRR+SQR: Spectral radii of $A - BF_i$ with different rank SSI, Example 5

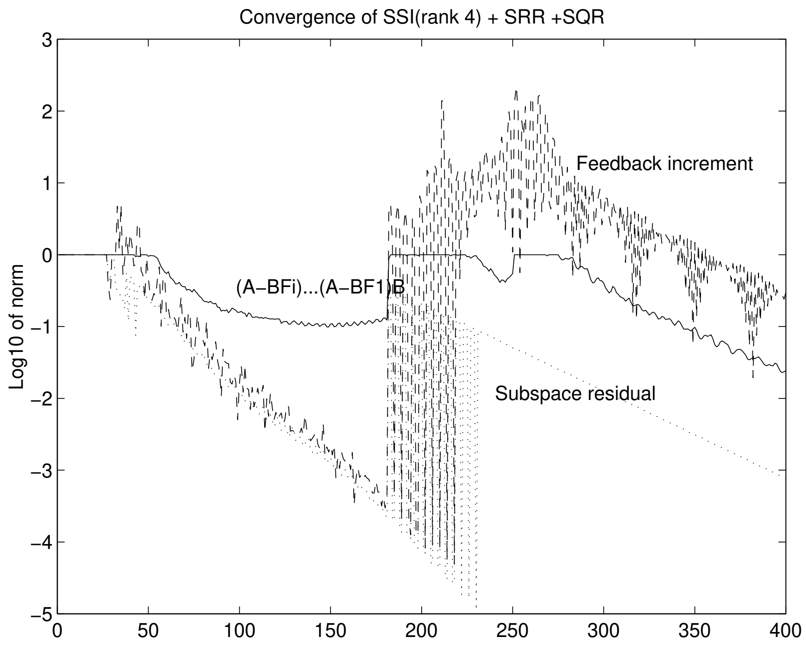

Figure 4.2. Rank 4 SSI+SRR+SQR: Stabilization and convergence, Example 5

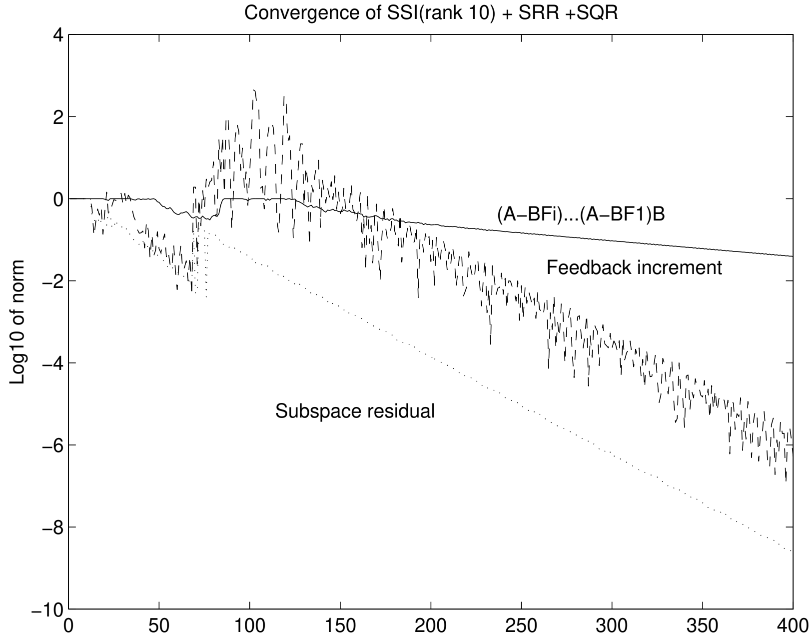

Figure 4.3. Rank 10 SSI+SRR+SQR: Stabilization and convergence, Example 5

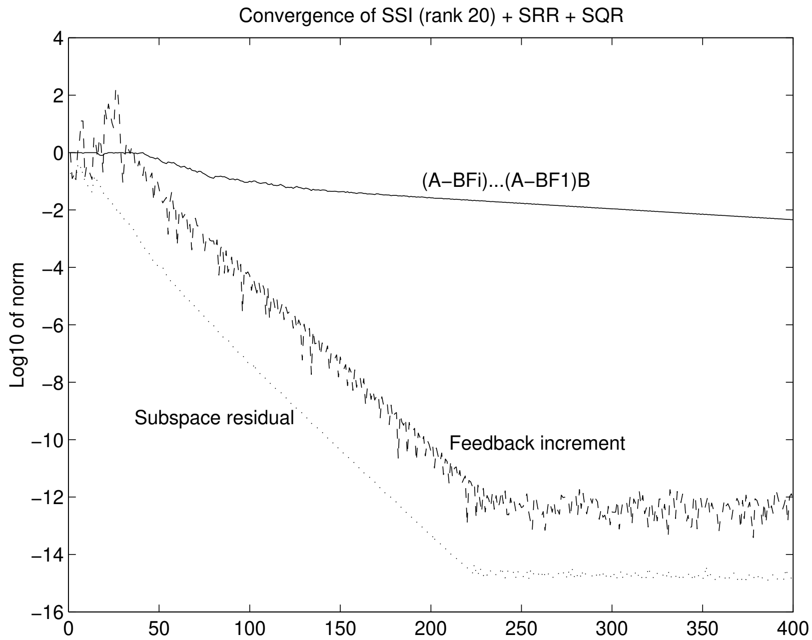

Figure 4.4. Rank 20 SSI+SRR+SQR: Stabilization and convergence, Example 5

CHAPTER 5

SQUARE ROOT ALGORITHM (SQR)

In this chapter, we discuss a low-rank stabilization algorithm, SQR, based on a square root form of the DRDE with $Q = 0$. Features of the new algorithm, especially the feature to keep stable eigenvalues unchanged and relationship with SSI, are analyzed. We also develop an effective stopping criterion that detects the stability of the closed-loop matrix $A - BF_i$ much earlier than feedback convergence. Finally, some enhancements to SQR are discussed.

Directly implementing the DRDE to solve stabilization problems for large scale systems suffers two disadvantages. One is the possible loss of positive semidefiniteness of P_i, which causes trouble with the inversion of $R + B'P_iB$ and potential numerical blow-up of P_i. The second is the update P_i and F_i that requires computations on matrices of order n that may in fact be dense. The symmetry of P_i can be kept with processing such as $0.5(P_i + P_i') \rightarrow P_i$, but it is not clear whether the positive semidefiniteness of P_i can be kept.

In [13], Morf and Kailath propose a square root algorithm to implement the DRDE. This algorithm takes advantage of the positive semidefiniteness of P_i, R, Q and expresses them as products of square roots, i.e., any positive semidefinite matrix X can be written as $X^{1/2}X^{T/2}$.

Let

$$\begin{aligned} R_i^{\epsilon} &= R + B'P_iB \\ \tilde{K}_i &= A'P_iB(R_i^{\epsilon})^{-T/2}. \end{aligned}$$

The DRDE can then be written as:

$$\begin{bmatrix} R^{1/2} & B'P_i^{1/2} & 0 \\ 0 & A'P_i^{1/2} & Q^{1/2} \end{bmatrix} U = \begin{bmatrix} (R_i^{\epsilon})^{1/2} & 0 & 0 \\ \tilde{K}_i & P_{i+1}^{1/2} & 0 \end{bmatrix} \tag{5.1}$$

where U is orthogonal.

Equation (5.1) solves the first problem of the direct implementation of DRDE. Equation (5.1) computes $P_i^{1/2}$ so that the positive semidefiniteness of P_i is maintained. To eliminate $Q^{1/2}$ from the left side of Equation (5.1), the QR decomposition involved is applied to a matrix with size of $(p+n+m) \times (p+n)$ where m is the rank of Q. Therefore $Q^{1/2}$ is an $n \times m$ matrix and $P_i^{1/2}$ is in triangular form. U is an $(p+n+m) \times (p+n+m)$ matrix and $P_i^{1/2}$ is an $n \times n$ matrix.

For large scale stabilization problems, the number of computations in Equation (5.1) is still too large. We can however explore the freedom of some parameters such as Q and P_0 to derive low-rank algorithms. In this chapter, we develop a low-rank stabilization algorithm from Equation (5.1) with the parameter choices of $Q = 0$ and low-rank P_0. In the next chapter, we develop another low-rank square root stabilization algorithm based on the DRDE with the parameter choices of low rank Q and $P_0 = 0$.

The low-rank algorithm SQR is based on the property that the rank of P_i is non-increasing during iterations of DRDE if $Q = 0$.

Proposition 5.1 *If $Q = 0$ in the DRDE, then*

$$rank(P_{i+1}^{1/2}) \leq rank(P_i^{1/2}),\ i = 0, 1, \cdots \tag{5.2}$$

Proof. Rewrite the DRDE with $Q = 0$ as

$$P_{i+1} = A'P_i^{1/2}(I - P^{T/2}B(R + B'P_iB)^{-1}P_i^{1/2})P_i^{T/2}A. \tag{5.3}$$

The inequality

$$rank(P_{i+1}) \leq rank(P_i^{1/2})$$

follows directly from Equation (5.3) and hence

$$rank(P_{i+1}^{1/2}) \leq rank(P_i^{1/2})$$

is proved. ∎

5.1 The Basic SQR Algorithm

Based on Proposition 5.1, a low-rank stabilization algorithm can be derived from the general square root algorithm (5.3). We assume $Q = 0$ in the DRE and the DRDE. (This assumption is maintained for the remainder of the chapter.) Choose P_0 such that $rank(P_0) = l$, or equivalently $P_0^{1/2}$ as an $n \times l$ matrix of full rank. Equation (5.1) becomes:

$$\begin{bmatrix} R^{1/2} & B'P_i^{1/2} \\ 0 & A'P_i^{1/2} \end{bmatrix} U_i = \begin{bmatrix} (R_i^{\epsilon})^{1/2} & 0 \\ \tilde{K}_i & P_{i+1}^{1/2} \end{bmatrix}. \tag{5.4}$$

The practical form of the algorithm consists of two stages:

$$\begin{bmatrix} R^{1/2} \ B'P_i^{1/2} \end{bmatrix} U_i = \begin{bmatrix} (R_i^{\epsilon})^{1/2} \ 0 \end{bmatrix} \quad \text{(QR stage)} \tag{5.5}$$

$$\begin{bmatrix} 0 \quad A'P_i^{1/2} \end{bmatrix} U_i = \begin{bmatrix} \tilde{K}_i \ P_{i+1}^{1/2} \end{bmatrix} \quad \text{(update stage)} \tag{5.6}$$

where U_i is orthogonal.

We will also modify another square root iteration for the DRDE in Chapter 7 given by Equations (7.8) and (7.11). It is a version used in the literature of reduced-rank Kalman filtering. However, if it were used for the stabilization situation in this chapter it would be at a disadvantage compared to SQR. The alternate form uses $\left[(A - BF_i)'P_i^{1/2} \ \ F_i'R^{-1/2}\right]$ on each iteration and must compute F_i from $(R+B'P_iB)^{-1}B'P_iA$. SQR does not compute the feedback on every iteration. When F_i is required it is computed from $\tilde{K}_i$ and $(R^{\epsilon})^{-1/2}$.

The iteration of Equation (5.5) and Equation (5.6) is much cheaper than the form that is not low-rank. The first step is to compute $(A', B')P_i^{1/2}$. We assume that (A', B') is an $n \times (n+p)$ sparse matrix with μ nonzero elements. (If p is small then B may in fact be dense but as p grows we assume that it becomes sparse.) This requires $O(\mu l)$ operations. The QR factorization of a $(p+l) \times p$ matrix to compute U_i and $(R_i^\epsilon)^{1/2}$ requires $O((p+l)p^2)$ operations. Computing $\begin{bmatrix} 0 & A'P_i^{1/2} \end{bmatrix} U_i = \begin{bmatrix} \tilde{K}_i \; P_{i+1}^{1/2} \end{bmatrix}$ by exploiting the zeros in the leading block requires $O(n(p+l)l^2)$ operations. (Note $P_{i+1}^{1/2}$ need not be in triangular form.)

The matrices $(R_i^\epsilon)^{1/2}$, $\tilde{K}_i$ and P_i must be stored in addition to A and B requiring $p^2/2 + n(p+l)$ locations. (F_i can be computed from $(R_i^\epsilon)^{1/2}$ and $\tilde{K}_i$.)

5.2 Convergence of SQR

From Theorem 3.4 in Chapter 3, we know that for DRE $Q = 0$, if A has no eigenvalue on unit circle, there exists a unique stabilizing and positive semidefinite solution P. Whether the corresponding DRDE converges to this stabilizing solution P depends on the initial guess P_0. Two sufficient conditions from Theorem 3.5 are that P_0 is full rank or is larger than the stabilizable solution P (P_0 is larger than P means $P_0 - P$ is nonnegative). Both conditions are difficult to guarantee in practice. Below we explore the DRE and DRDE to relax the conditions on P_0 to construct a low-rank stabilization algorithm from the basic SQR iteration.

Theorem 5.1 *Assume (A, B) is stabilizable and P is the stabilizing solution of the DRE with $Q = 0$. Let $A - BF$ be the corresponding closed-loop matrix. For any λ and v such that $Av = \lambda v$ we have:*

1. *If $|\lambda| > 1$, Pv is a left eigenvector of $A - BF$ with eigenvalue $1/\lambda$.*
2. *If $|\lambda| < 1$, then $Pv = 0$ and v is a right eigenvector of $A - BF$ with eigenvalue λ.*

Proof. Rewrite the DRE with $Q = 0$ as

$$P = A'P(A - BF)$$

where $F = (R + B'PB)^{-1}B'PA$ and multiply by v' on both sides. We have

$$\begin{aligned} v'P &= (Av)'P(A - BF) \\ &= \lambda v'P(A - BF) \end{aligned}$$

It follows that either $v'P = 0$ or Pv is the left eigenvector of $(A - BF)$ with eigenvalue $1/\lambda$. Since $A - BF$ is stable, then the latter condition can only hold if $|1/\lambda| < 1$ which implies $|\lambda| > 1$, i.e., the condition only holds for unstable eigenvalues of A.

If $|\lambda| < 1$ we must therefore have $v'P = Pv = 0$. We also have

$$\begin{aligned} (A - BF)v &= (A - B(R + B'PB)^{-1}B'PA)v \\ &= \lambda(v - B(R + B'PB)^{-1}B'Pv) \\ &= \lambda v \end{aligned}$$

Therefore, all stable eigenvalues of A are eigenvalues of $A - BF$ with the same right eigenvectors. ■

Theorem 5.1 shows that a stabilizing solution (P or F) of the DRE will keep all stable eigenvalues of A unchanged and move the unstable eigenvalues of A to their reciprocals inside unit circle. The implication of Theorem 5.1 is that the rank of the stabilizing solution P of the DRE should be the same as the number of unstable eigenvalues of A. Proposition 5.1 tells us that the rank of P_i in DRDE is non-increasing. So, in order to get a stabilizing solution P or stabilizing feedback F, the rank of P_0 in the DRDE should be at least the same as the number of unstable eigenvalues of A, which determines efficiency of the SQR just as it did for the projection algorithms, e.g., SSI, of the Chapter 4.

The relationship to SSI is, in fact, stronger than the observation above. Equation (5.6) indicates that $P_i^{1/2}$ is updated in a power method-like manner, just as in SSI – $P_{i+1}^{1/2} = A'P_i^{1/2}U_i^{22}$ where U_i^{22} is a nonsingular $l \times l$ matrix. The subspaces from SSI and SQR are related by the following theorem.

Theorem 5.2 *Let $P_i^{1/2}$, $i = 0, 1, \cdots$, be generated by Equations (5.5) and (5.6) and let Q_i be an orthogonal basis of the $IM(P_i^{1/2})$. The subspace $IM(Q_i)$ is the same as subspace obtained from SSI on A' with the initial subspace $IM(P_0^{1/2})$.*

Proof. From Equations (5.5) and (5.6), we have $P_{i+1}^{1/2} = A' P_i^{1/2} U_i^{22}$ where U_i^{22} is the $l \times l$ trailing principal submatrix of U_i. It follows that

$$P_{i+1}^{1/2} = (A')^{i+1} P_0^{1/2} U_0^{22} \cdots U_i^{22}.$$

On the other hand, let $\tilde{Q}_0$ be an orthogonal basis of $IM(P_0^{1/2})$ and apply SSI – $A'\tilde{Q}_i = \tilde{Q}_{i+1} R_{i+1}$. By induction, we have

$$(A')^{i+1} \tilde{Q}_0 = \tilde{Q}_{i+1} R_{i+1} \cdots R_0$$

and

$$\tilde{Q}_{i+1} = (A')^{i+1} \tilde{Q}_0 R_0^{-1} \cdots R_{i+1}^{-1}.$$

The QR decomposition of $P_{i+1}^{1/2}$ yields

$$Q_{i+1} = (A')^{i+1} Q_0 U_0^{22} \cdots U_i^{22} R_{i+1}^{-1}.$$

Therefore, the SQR and SSI iterations on A' with the same initial basis generate the same subspace sequence. ■

From Theorem 5.2, intuition would indicate that the conditions on P_0 for the DRDE to converge should have much in common with those used for SSI.

Theorem 5.3 *Let V be an orthogonal basis of $P_0^{1/2}$. If the subspace spanned by V is left invariant subspace of A, then P_i generated by SQR with initial $P_0^{1/2}$ will converge provided that (A, B) is stabilizable and A has no eigenvalues on unit circle. Moreover, if V contains the left unstable invariant subspace of A, SQR will converge to a stabilizing solution.*

Proof. $P_0^{1/2}$ can be written in the form $P_0^{1/2} = VR_0$ with R_0 is of full rank. From the update stage of SQR iteration, if V is left invariant to A, all $P_i^{1/2}$ generated by SQR will have form $P_i^{1/2} = VR_i$ and the $P_i = VR_i(VR_i)'$, $i = 0, 1, \cdots$, satisfy the DRDE

$$P_{i+1} = A'P_iA - A'P_iB(R + B'P_iB)^{-1}B'P_iA.$$

Multiplying by V' and V on both sides yields that the $p_i = R_iR_i'$, $i = 0, 1, \cdots$ satisfy

$$p_{i+1} = (V'AV)'p_i(V'AV)' - (V'AV)'p_i(V'B)(R + (V'B)'p_i(V'B))^{-1}(V'B)'p_i(V'AV).$$

Therefore p_i is the solution of DRDE with parameters $(V'AV, V'B)$ which are generated from projection of (A, B) on V. Since (A, B) is stabilizable and A has no eigenvalues on unit circle, $(V'AV, V'B)$ is stabilizable and $V'AV$ has no eigenvalues on unit circle. $p_0 = R_0R_0' > 0$ implies p_i will converge to a stabilizing solution p of the projected DRDE. Therefore $P_i = Vp_iV'$ will converge to $P = VpV'$.

F can be computed from $F = R^{-1}B'(A')^{-1}P$. Now assume $A'V = V\Lambda$ where Λ is some full rank square matrix. It follows that

$$\begin{aligned} F &= R^{-1}B'V\Lambda^{-1}P \\ &= R^{-1}(V'B)'\Lambda^{-1}VpV' \\ &= R^{-1}(V'B)'((V'AV)')^{-1}pV' \\ &= fV' \end{aligned}$$

where f is the suboptimal feedback from the projected DRDE. From the earlier discussion of projection methods, we know that if V contains left unstable invariant subspace of A and f stabilizes the projected system, $F = fV'$ will stabilize the original system (A, B). Therefore the converged solution from SQR is a stabilizing solution. ■

Considering that SSI and SQR can generate the same subspace sequence, if SQR starts from $P_0^{1/2}$ and SSI starts from V_0 which is an orthogonal basis of $P_0^{1/2}$, intuitively, from above proposition, convergence of SSI should imply the convergence of SQR. There are some details that are not yet clear however. In the case of SSI, the phase of the algorithm iterates until a good approximation of the unstable invariant subspace basis is available. The projection and stabilization in the second phase of the algorithm assumes an exact basis is available. In order to show that the approximate basis also stabilizes one must appeal to the perturbation results in Chapter 2 and deduce how close the approximation has to be yield a nontrivial stability radius.

The SQR algorithm can be viewed for the purpose of convergence discussion as a two phase algorithm as well. Theorem 5.2 indicates that SSI and SQR produce the same spaces, therefore, at some point SQR will have a very good approximation of the invariant unstable subspace basis that SSI would pass to its projection phase. If this basis was exact then we know from Theorem 5.3 that convergence to a stabilizer is assured. It is not yet clear how to modify Theorem 5.3 to take into account an initial space that is only an approximation to the invariant unstable space, i.e., the assumption of containment is too strong. Unfortunately, with $Q = 0$, certain monotonic properties that are the key to the proofs of existing convergence theorems for the DRDE are not provable. Further theoretical work is needed to provide rigorous proof of this claim and to find a more general condition on P_0 under which the DRDE will converge.

However, the heuristic arguments above and our experience with SQR and its modifications indicate that the iteration does indeed converge where SSI and its hybrids converge and often converges when they do not. The theory built in this section provides the necessary background to make the intuitive leap that SQR does in fact converge given mild conditions on P_0.

The reason SSI diverges or converges very slowly is that A has eigenvalues λ_m and λ_{m+1} which are poorly separated in absolute value and SSI cannot get these components to converge or vanish. If columns of $P_0^{1/2}$ are generated via Arnoldi, poor separation of $|\lambda_l|$ and $|\lambda_{l+1}|$ will cause slow convergence or divergence of SSI starting from the orthogonal basis of $P_0^{1/2}$. This does not necessarily mean slow convergence or divergence of SQR. The key difference between the SSI iteration and SQR is given in Theorem 5.1. It shows that the DRDE removes the components of $P_i^{1/2}$ in the direction of the stable eigenvectors of A, with the largest stable ones leaving fastest. Assuming $P_i^{1/2}$ from SQR does not blow up, the components of stable eigenvalues of A in $P_i^{1/2}$ will vanish with at least the rate of the largest stable eigenvalues. SSI and its hybrids have no such mechanism guaranteed. As a result, when SSI converges we expect SQR to converge (with SQR possibly converging faster) and in cases where SSI does not converge or converges slowly we would expect SQR to potentially converge. We state this as an unproven conjecture.

Conjecture 1 *Let V_0 be an orthogonal basis of $P_0^{1/2}$. If SSI on A' starting with V_0 converges, the corresponding SQR will converge. If SSI does not converge due to eigenvalues λ_m and λ_{m+1} that are stable eigenvalues of A, the corresponding SQR will converge.*

SQR is a numerically stable implementation of the DRDE, so it should converge with the same rate as the DRDE. Equation (3.15) and Equation (3.16) suggest the convergence rate of SQR depends on the convergence rate of the reduced DRDE that is obtained by projecting the DRDE to the subspace spanned by $P_\infty^{1/2}$ and the convergence of subspace sequence extracted from $P_i^{1/2}$. If the subspace sequence converges to an unstable invariant subspace of A', the convergence rate of SQR is determined by the convergence rate of corresponding SSI and the closed-loop spectral radius of the reduced DRDE, whichever is slower. Otherwise, the convergence rate of SQR is determined by the closed-loop spectral radius of $\rho(A - BF)$. From Equation (3.17) of Chapter 3, the feedback F_i will converge at the same rate as P_i in the DRDE and SQR.

5.3 Termination Criteria for the SQR Algorithm

For a large scale system, it is impossible to compute the closed-loop spectral radius as a stopping criterion of the SQR algorithm. Feedback convergence (measured by difference of feedback in two consecutive steps of iteration) is easy to compute but, based on empirical observations it is not reliable. It often indicates stabilization far too late in the iterations especially when some eigenvalues of A are very close to unit circle.

Theorem 5.1 and Theorem 5.2 indicate new methods to terminate SQR for large scale systems. Using the DRE with $Q = 0$ to find a stabilizing F is equivalent to projecting the system (A, B) onto the left unstable invariant subspace of A and then to finding a stabilizer. If V is a left unstable invariant subspace of A and F is produced by the DRE with $Q = 0$, we know from the discussion about projection method in Chapter 4, the orthogonal matrix $\tilde{V} = (V_s\ V)$ transforms F and B as follows

$$\begin{aligned} F\tilde{V} &= (0\ F_u) \\ \tilde{V}'B &= (B_s\ B_u)' \end{aligned}$$

Therefore, A and $A - BF$ can be decomposed into

$$\begin{aligned} \tilde{V}'A\tilde{V} &= \begin{pmatrix} A_s & A_{12} \\ & A_u \end{pmatrix} \\ \tilde{V}'(A - BF)\tilde{V} &= \begin{pmatrix} A_s & A_{12} - B_sF_u \\ & A_u - B_sF_u \end{pmatrix}. \end{aligned}$$

The subscripts s and u indicate stable and unstable portions of the system respectively.

It is clear that V should be the left invariant subspace of closed-loop $A - BF$ into which all unstable eigenvalues of A are moved. If V only contains stable eigenvalues of $A - BF$, and it is large enough to contain all unstable eigenvalues of A, we can expect that we have stabilized A. Assume V_i is obtained via QR decomposition and

Stewart's refinement from $P_i^{1/2}$ such that all eigenvalues of $V_i'A'V_i$ are unstable. If V_i converges and all eigenvalues of $V_i'(A-BF_i)V_i$ are stable, we can conclude that we have stabilized the system. We propose that the convergence of eigenvalues or spectral radius of $V_i'(A-BF_i)V_i$ as a new stabilization stopping criterion.

Of course, the stability of $V_i'(A-BF_i)V_i$ is not enough to guarantee the stability of $A-BF_i$, but the eigenvalue convergence plus stability of $V_i'(A-BF_i)V_i$ seems to be enough in practice. High accuracy of the eigenvalues convergence of $V_i'(A-BF_i)V_i$ is unnecessary and 2 to 3 digits should be enough unless some unstable eigenvalues of A are too close to unit circle. The difficulty here is that we cannot compute the accuracy of eigenvalues or spectral radius of $V_i'(A-BF_i)V_i$ as estimates of the spectral radius or eigenvalues of $V_\infty'(A-BF_\infty)V_\infty$. In practice, we compare the eigenvalues or spectral radius of $V_i'(A-BF_i)V_i$ in a window of small size, say 10. When doing so, we typically use more accuracy than mentioned above, e.g., 3 to 4 digits.

Empirical evidence has shown another test based on our theory to be superior to those mentioned above. Examining the difference between the reciprocals of the eigenvalues of $V_i'A'V_i$ and eigenvalues of $V_i'(A-BF_i)'V_i$ when the rank of V_i reaches the rank of unstable invariant subspace of A seems to provide a robust termination criterion.

Definition 11 (Stabilization criterion related to V_i) *Let V_i be extracted from $P_i^{1/2}$ during SQR iterations as approximated unstable invariant subspace of A' and F_i is feedback from SQR. Let ME_i be the set containing the reciprocals (inside unit circle) of eigenvalues of $V_i'A'V_i$, and E_{A-BF_i} be the set containing eigenvalues of $V_i'(A-BF_i)V_i$, ordered with absolute value from small to large. Let ΔV_i be the norm of the difference of ME_i and E_{A-BF_i}. When the rank of V_i reaches the number of unstable eigenvalues of A, or dose not change any more during SQR iterations, we will use ΔV_i as a stabilization criterion for SQR algorithm.*

Note ΔV_i is effective only when rank of V_i becomes stable during SQR. Fortunately, checking the rank of V_i is straightforward. The operation count for extracting V_i or projecting $A - BF_i$ or A on V_i is the same as that of one step SQR. So, it is too expensive to compute this stopping criterion ΔV_i for each iteration. Instead, after each K steps of SQR, we extract V_i, check its rank, and compute ΔV_i if rank of V_i is stable. If d is the smallest distance between the reciprocal of the unstable eigenvalues of A and the unit circle, we do not need V_i reach high accuracy. If $d > 0.02$, tolerance of 10^{-2} is enough for ΔV_i to catch stability of $A - BF_i$. If $d > 0.002$, the tolerance of 10^{-3} should be enough. When the stabilization process with SQR is very slow, we need other techniques such scaling A or variants of the algorithm discussed in Chapter 6 to accelerate and detect convergence.

The following examples investigate the efficiency of SQR when applied to a system which is well-conditioned and easy to stabilize and another system which is less well-conditioned, hard to be stabilized.

Example 7 *The system matrix A is constructed by randomly generating a 100×100 matrix with MATLAB RANDN and scaling it to $\tilde{A}$ so that the spectral radius of $\tilde{A}$ is 0.9.*

The 100×1 matrix B is generated randomly with RAND as is a 1×100 matrix $\tilde{F}$. Construct $A = \tilde{A} + B\tilde{F}$. All eigenvalues of A (dots in Figure 5.1) are well separated from unit circle and only two are unstable. The norm $\|\tilde{F}\|_2 = 9.9734$ and eigen-condition number of $\tilde{A}$ is $\kappa_2(X, \tilde{A}) = 110.8009$. So the system (A, B) should be well-conditioned and easy to stabilize. Figure 5.3 and Figure 5.4 show the results of rank 2 SQR. Figure 5.5 and Figure 5.6 show the result of rank 3 SQR. $P_0^{1/2}$ is randomly generated with RAND. The spectral radius of $A - BF_i$ from both rank 2 and rank 3 SQR converges within 7 iterations, which is shown in Figure 5.3 and Figure 5.5. With the new stabilization criterion ΔV_i related to projections of $A - BF_i$

and A on V_i, Figure 5.2 catches stabilization of both Rank 2 and Rank 3 SQR at step 7 with tolerance 10^{-2}, while residuals from SSI and $\|F_i - F_{i-1}\|_2$ have not reached 10^{-2} (see Figure 5.4 and Figure 5.6). Furthermore, Figure 5.1 shows the spectrum of $A - BF_5$ (the curve of $+$ symbols) converges within 5 iterations. We also see that the only eigenvalues moved are the two unstable ones of A. Figure 5.4 and Figure 5.6 show the convergence of the feedback and V_i with SQR, SSI convergence and another stabilization criterion $(A - BF_i) \cdots (A - BF_1)B$. In Figure 5.4, the ranks of SQR and SSI are taken to be the order of the unstable space and the subspace sequence (orthogonal basis) from $P_i^{1/2}$ converges to an unstable invariant subspace of A'. We see feedback of SQR and the SSI residual converge with the same rate. In Figure 5.6, we use the wrong rank for the subspace, and SSI has trouble converging and needs Stewart's refinement with significant additional computational cost. The feedback of SQR still converges with a rate consistent to closed-loop spectral radius as predicted.

Example 8 *In this example, A is a 100×100 matrix generated by RANDN of MATLAB and is scaled so that A has the spectrum described in Figure 5.7. B is a 100×1 matrix generated by RAND of MATLAB. Figure 5.7 shows there are plenty of eigenvalues, both stable and unstable, which are very close to the unit circle. It is, therefore, expected that stabilization with SQR or SSI will be very costly.*

Figure 5.8 to Figure 5.12 show the efficiency of SSI and SQR of rank 4 (correct order) and rank 5 (incorrect order). Figure 5.9 and Figure 5.11 show about 300 steps of SQR are taken before the closed-loop spectral radius of $A - BF_i$ starts to converge. With ΔV_i, Figure 5.8 catches stability of $A - BF_i$ at about step 350 for Rank 4 SQR and step 280 for rank 5 SQR with tolerance 10^{-2}, while both $\|F_i - F_{i-1}\|_2$ from SQR, subspace residuals for unstable subspace V_i and SSI reach accuracy 10^{-2} to 10^{-4} and hence do not indicate convergence. Figure 5.7 shows that the closed-loop spectrum of $A - BF_i$ converges at step 300 of SQR and only unstable eigenvalues of A are moved.

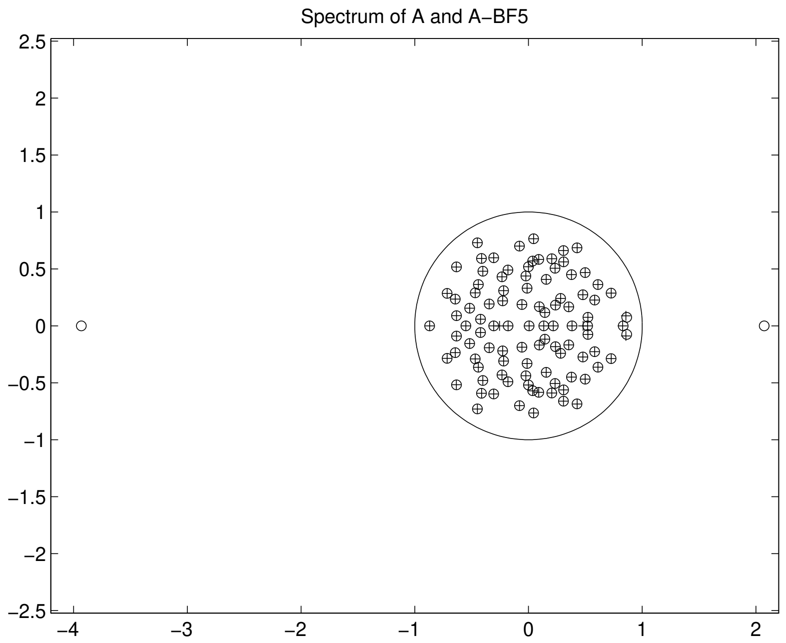

Figure 5.1. SQR: Spectrums for Example 7

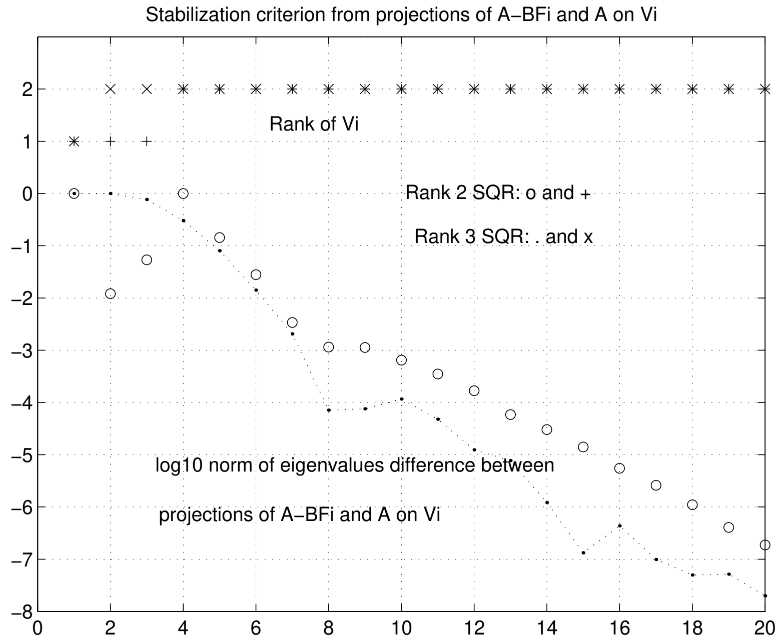

Figure 5.2. SQR: $\log_{10}(\Delta V_i)$, Example 7

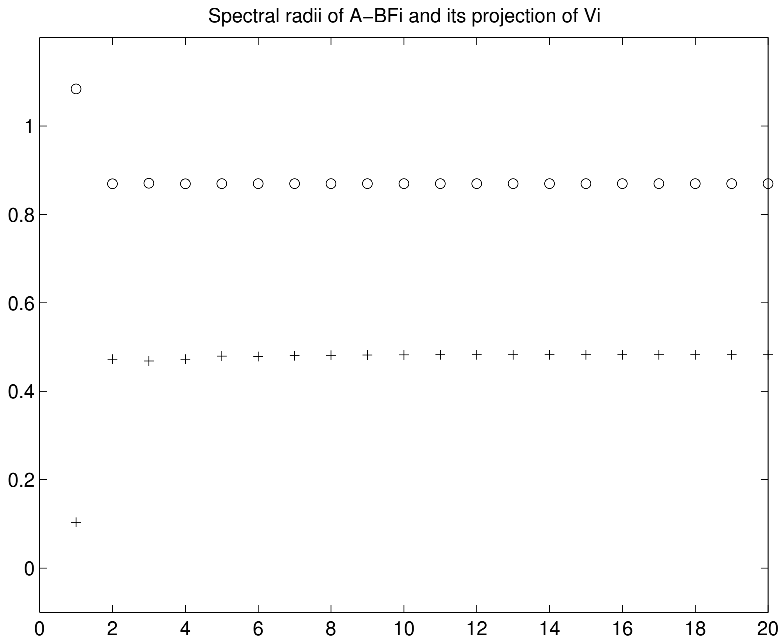

Figure 5.3. SQR: $A - BF_i$ (o) and $V_i'(A - BF_i)V_i$ (+), Example 7

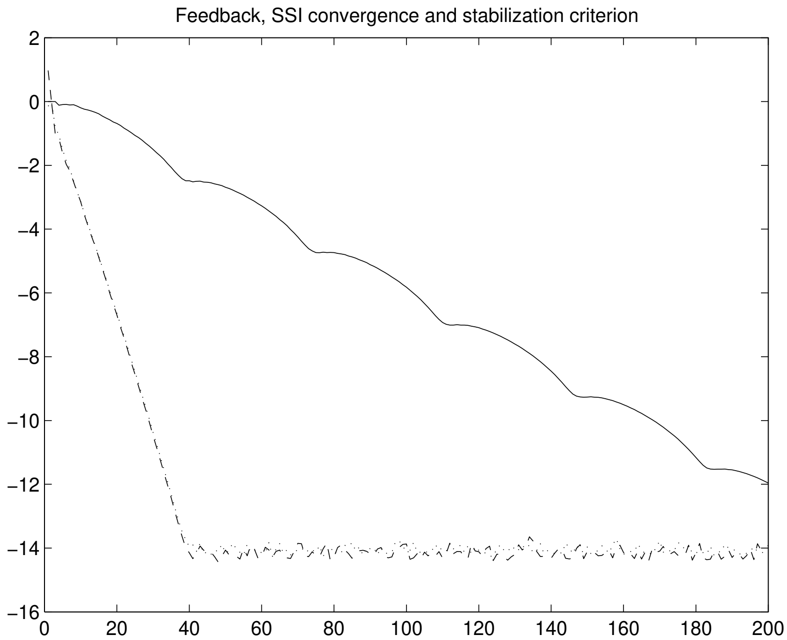

Figure 5.4. SQR: Feedback (- -), SSI (.) and $\prod(A - BF_i)B$ (–), Example 7

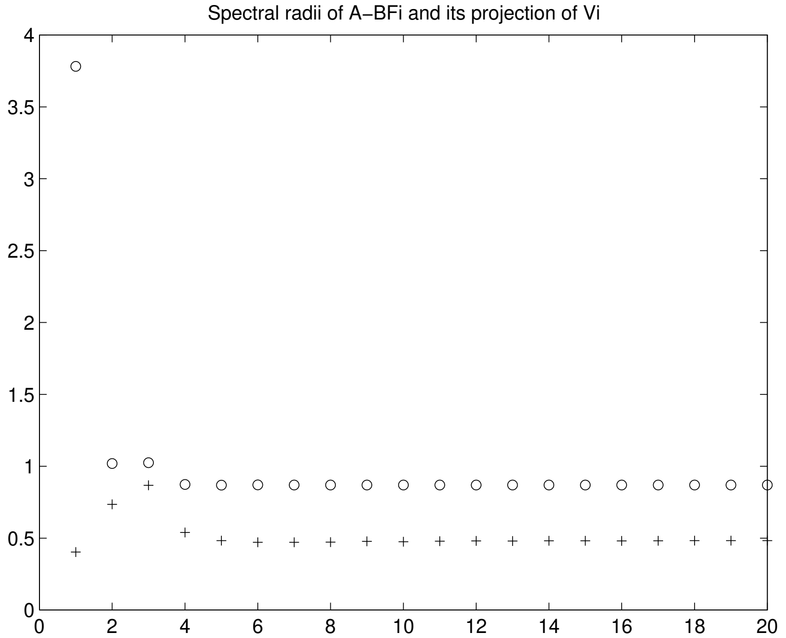

Figure 5.5. SQR: $A - BF_i$ (o) and $V_i'(A - BF_i)V_i$ (+) , Example 7

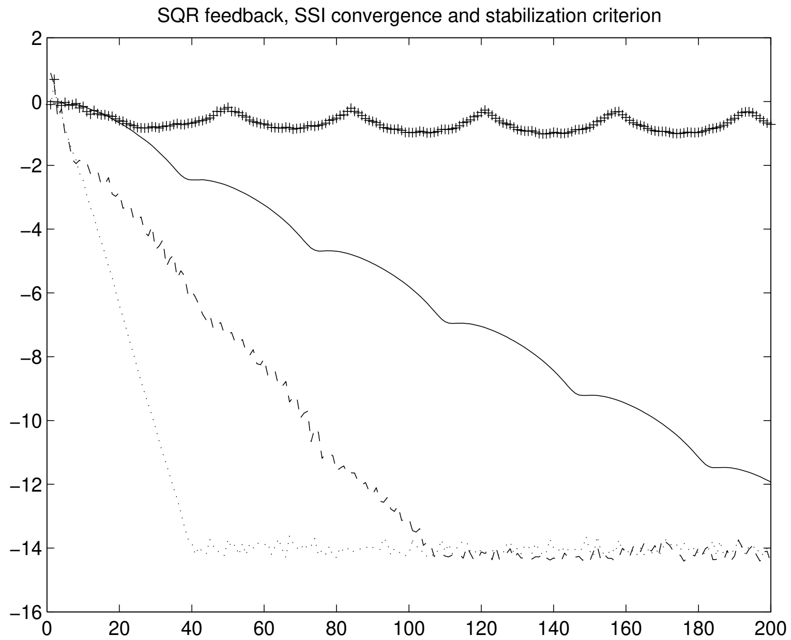

Figure 5.6. SQR: Feedback (- -), V_i (.), SSI (+) and $\prod(A - BF_i)B$ (–) , Example 7

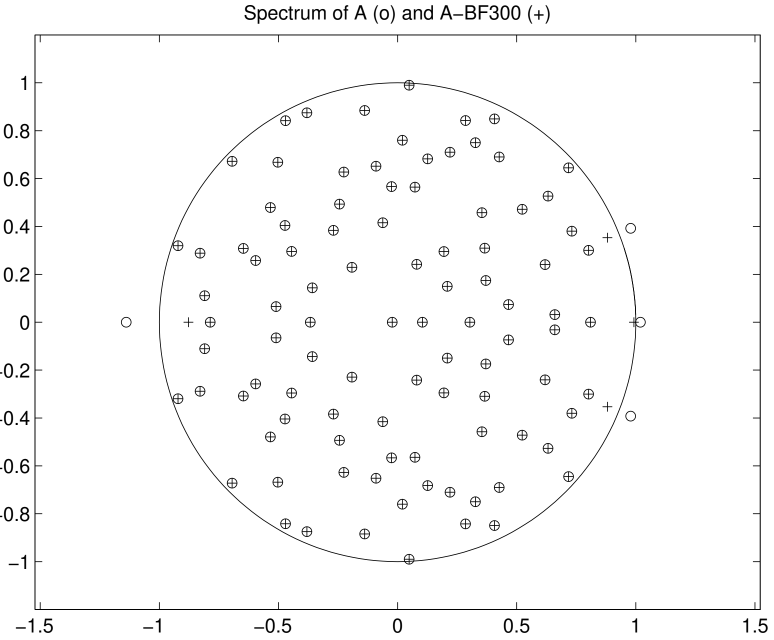

Figure 5.7. SQR: Spectrum of A (o) and $A - BF_{300}$ (+), Example 8

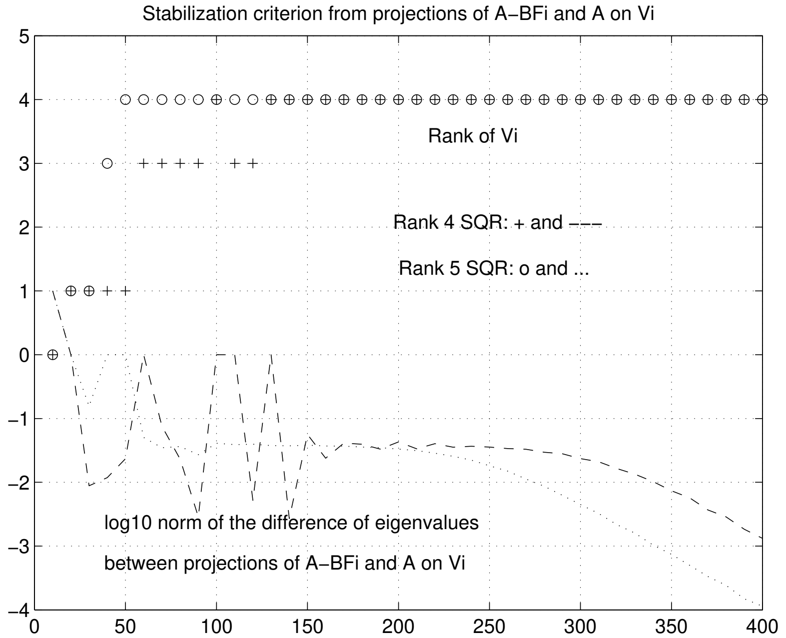

Figure 5.8. SQR: $\log_{10}(\Delta V_i)$, Example 8

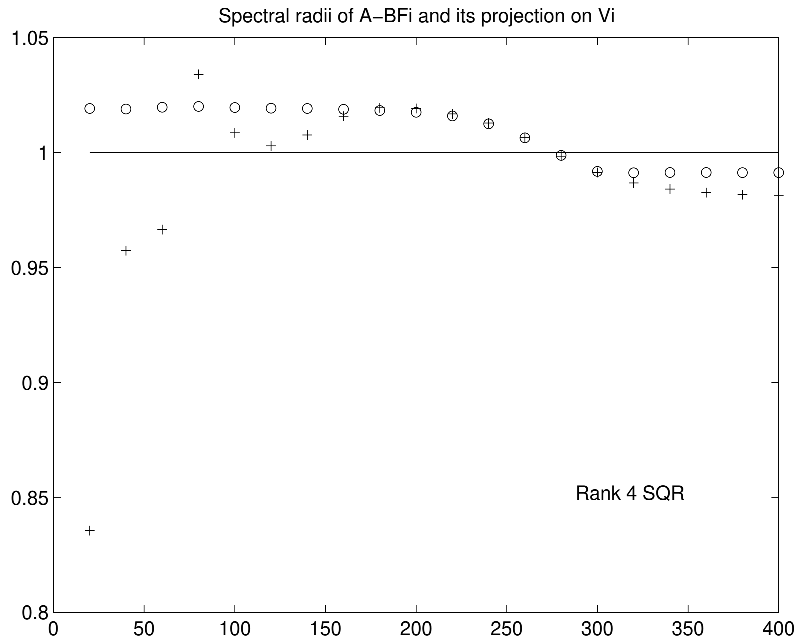

Figure 5.9. SQR: $A - BF_i$ (o) and $V_i'(A - BF_i)V_i$ (+), Example 8

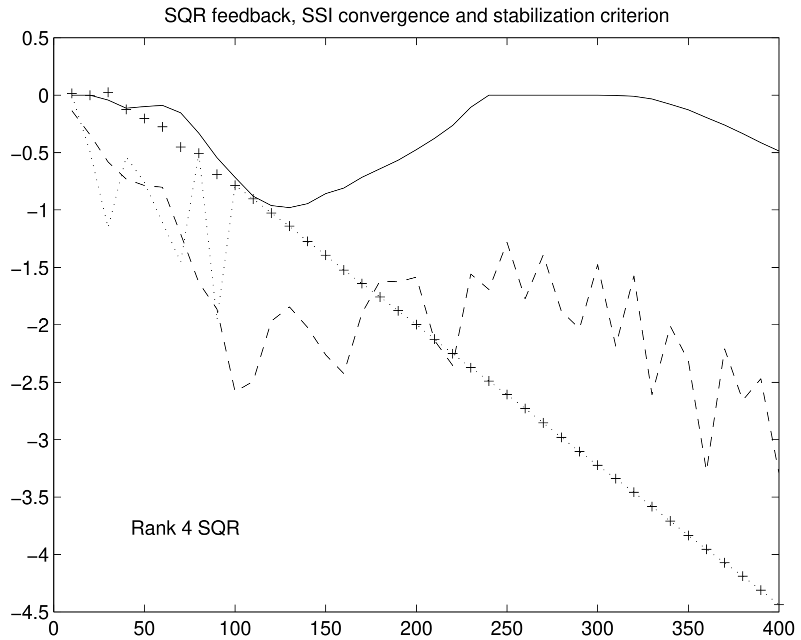

Figure 5.10. SQR: Feedback (- -), SSI (.) and $\prod(A - BF_i)B$ (–), Example 8

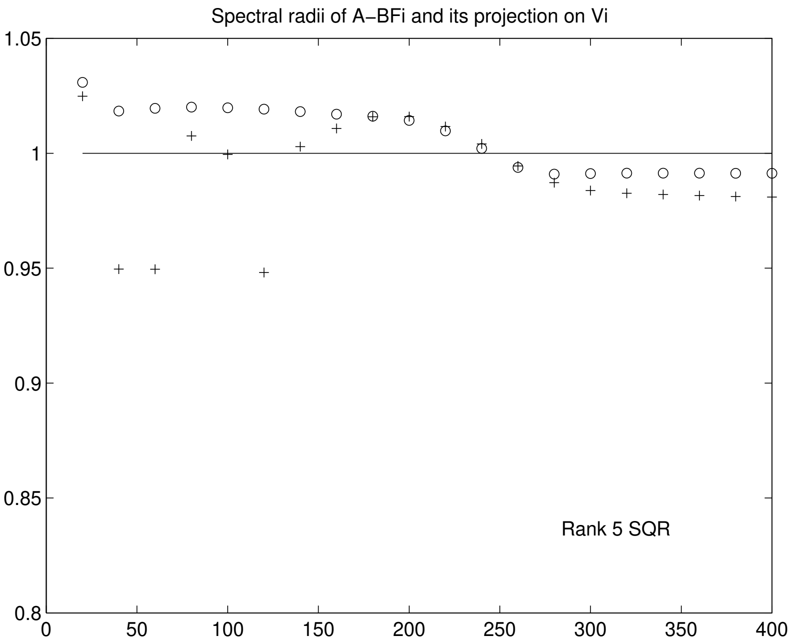

Figure 5.11. SQR: $A - BF_i$ (o) and $V_i'(A - BF_i)V_i$ (+), Example 8

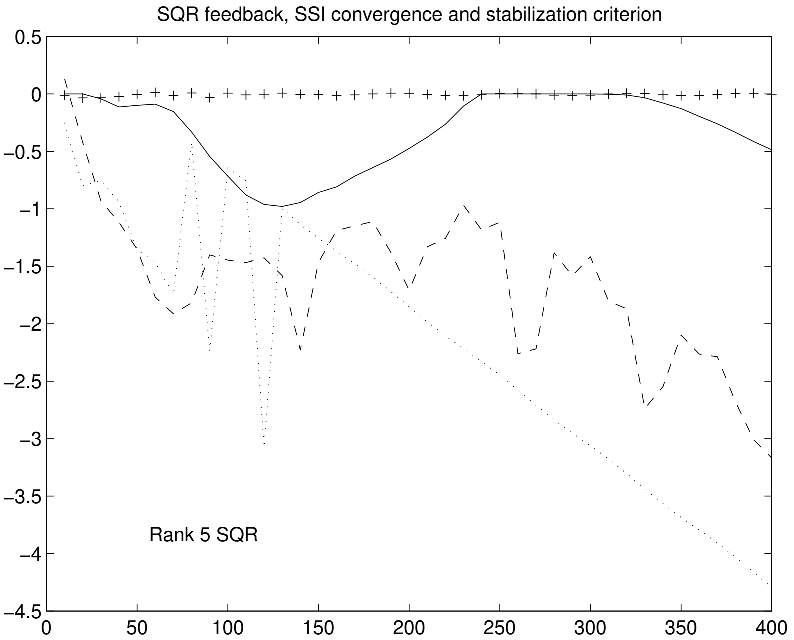

Figure 5.12. SQR: Feedback(- -), V_i (.), SSI (+) and $\prod(A - BF_i)B$ (−) , Example 8

Example 7 illustrates that for a well-conditioned stabilization problem where (1) A has only few unstable eigenvalues and (2) all eigenvalues of A are well separated from unit circle, SQR algorithm is very efficient and stabilization is reached within only a few steps, while the residuals of approximated unstable subspace from SSI have not reached one digit accuracy. We can relax the condition that all eigenvalues of A are well-separated from unit circle to that all unstable eigenvalues of A are well separated from all stable eigenvalues of A. If unstable eigenvalues of A are well-separated from the unit circle, fast stabilization with SQR is expected and feedback convergence with SQR depends on the choice of the rank of $P_0^{1/2}$ as discussed earlier, with the worst case happening when some stable eigenvalues of A are very close to unit circle and we choose an incorrect rank of $P_0^{1/2}$ (larger than the number of unstable eigenvalues of A). In this case, we can monitor the eigenvalue convergence of $V_i'(A - BF_i)V_i$ to catch the stability of $A - BF_i$ or modify the rank of $P_i^{1/2}$ during the iteration. If some unstable eigenvalues of A are very close to unit circle and stable eigenvalues of A are well-separated from unit circle, some scaling on A can help to accelerate both stabilization and feedback convergence. The details are discussed later.

From Example 8, we see the limitations of SQR when applied to a system which is less well-conditioned and A has many stable and unstable eigenvalues which are close to unit circle. From the previous discussion of SQR convergence, both subspace iteration associated with SQR and feedback convergence, it is expected that stabilization and feedback convergence with SQR will be very slow and SSI will converge slowly, also.

5.4 Acceleration of the Basic SQR Algorithm

SSI can be accelerated with Stewart's refinement unless A has too many unstable and near unstable eigenvalues around unit circle. There are two candidate acceleration techniques for SQR: restarting SQR or scaling A.

The idea behind restarted SQR will implement N rounds of K steps SQR iterations. At the end of $i-th$ round, we update A to $A - BF_K^{(i)}$. In $(i+1)-th$ round, we run K steps SQR on updated system (A, B). This means, once an unstable eigenvalue is stabilized, it will be filtered out so the following SQR will focus on the eigenvalues which have not been stabilized. Unfortunately, restarted SQR does not perform that well in our experience. The technique however does perform well for the algorithm of Chapter 6 and is discussed in more detail there.

When using SQR (or, in fact, the DRDE), large eigenvalues tend to be stabilized or moved close to unit circle very fast but, the unstable eigenvalues that are close to unit circle need many more iterations to stabilize. With $r < 1$, when using SQR on $(A/r, B/r)$, A/r will enlarge any unstable eigenvalue λ of A to λ/r which will be stabilized or at least be moved close to unit circle very quickly by $A/r - (B/r)F_i$, and λ will be stabilized by $A - BF_i$.

This scaling technique works very well if there exists a $r < 1$ such that the number of eigenvalues of A with absolute value not less than r is very small. A special case of this is when some unstable eigenvalues of A are very close to unit circle and stable eigenvalues of A are well-separated from unit circle (with the largest stable eigenvalues having modulus near r). For such special case, the stabilization and feedback convergence with general SQR are very slow, but the technique with scaling A, without changing the rank of $P_0^{1/2}$, will accelerate both stabilization and feedback convergence.

In some cases, however, there may not be such a simple separation between the unstable eigenvalues around the unit circle and all of the stable eigenvalues. Some stable eigenvalues may be near the unit circle as well and therefore for the scaled problem will appear in the unstable set. These stable eigenvalues must be moved along with the unstable eigenvalues in order to efficiently stabilize the system. This increase in the number of eigenvalues moved by $A - BF$ with scaling r influences the choice of rank of $P_0^{1/2}$ and suggests we may need to increase the rank of $P_0^{1/2}$ so that the rank is consistent to number of unstable eigenvalues of A/r. If this can be done then efficient stabilization is possible. The stable portion of the original spectrum, however, is modified in contrast to SQR or SSI on the original system.

These difficulties can make the application of scaling to SQR difficult for the general problem. As with restarting, we see in Chapter 6 that scaling techniques applied to the algorithm developed there can be very successful acceleration techniques.

Example 9 *A, B are the same as in Example 8. r is chosen as 0.98, 0.9 and 0.9, the rank of $P_0^{1/2}$ is chosen 6, 10, 18. Figure 5.13 shows the spectral radius of $A-BF_i$ and Figure 5.14 shows the stabilization criterion $\log_{10}\|(\prod_{k=1}^{i}(A-BF_k))B\|_2$. SQR with scaling A takes much less SQR iteration steps to stabilize. The vibration in Figure 5.13 is from the bad rank choice (10) of $P_0^{1/2}$ which is inconsistent with $r = 0.9$. Note the rank of $P_0^{1/2}$ is significantly larger than the number of unstable eigenvalues of A, which is the major disadvantage of SQR and SSI.*

5.5 Comparison of SQR and SSI

For SQR, we focus on feedback convergence and the ΔV_i stabilization criterion to check convergence. For SSI, we focus on subspace residuals. So, SQR and SSI performance can often be difficult to compare in general. Some clear advantages are present in SQR however.

1. The feedback F_i can be computed from $(R_i^\epsilon)^{1/2}$ and $\tilde{K}_i$ cheaply compared to SSI.

2. For stabilization, the feedback convergence check of $\|F_i - F_{i-1}\|$ is therefore very cheap for SQR. Computing subspace residual check costs the same for SQR and SSI.

3. The use of ΔV_i can detect the stability of $A - BF_i$ much earlier than residual convergence check of SSI or feedback convergence check of SQR. It is not known

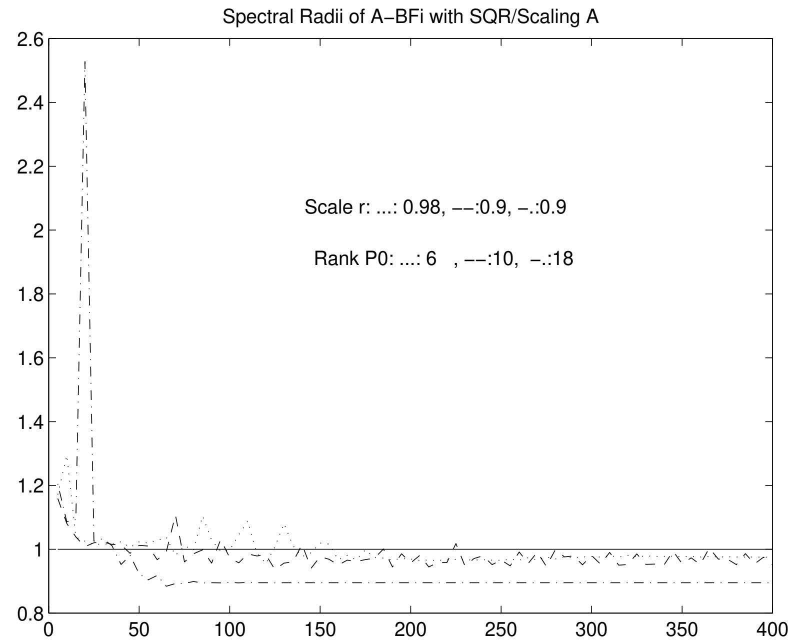

Figure 5.13. SQR with scaling A: $A - BF_i$, Example 9

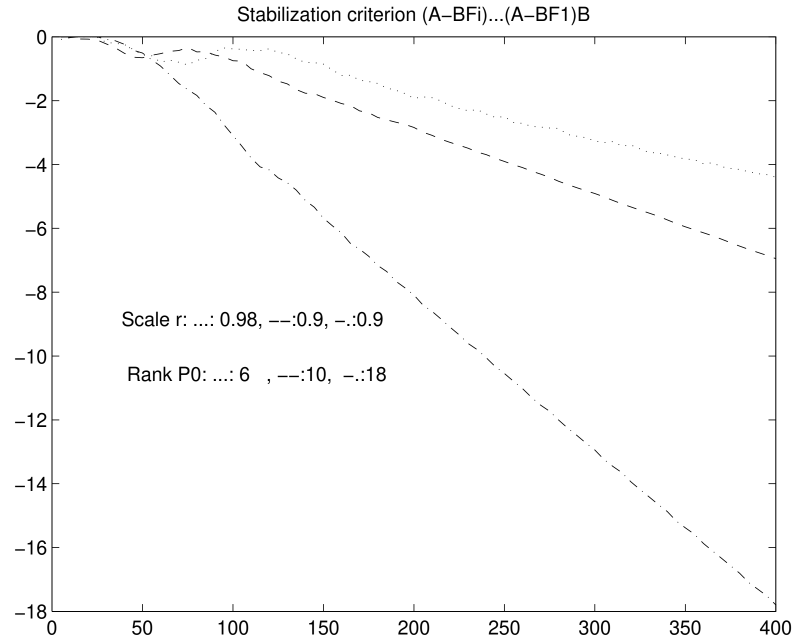

Figure 5.14. SQR with scaling A: $\prod(A - BF_i)B$,Example 9

how accurate the subspace residuals should be in order to stop SSI iterations with a high probability of stabilization.

4. SQR can produce the same subspace sequence as SSI with only the additional cost of low order QR decomposition. As a result, if feedback convergence is too slow, we can always check the associated subspace V_i for adjustment and projection, i.e., SQR has the ability to shift into SSI mode for problems where SQR has inferior convergence.

5. For well-conditioned stabilization problems where the eigenvalues of A are well-separated from the unit circle, the rank choice in SQR is not as critical as it is in SSI. For stabilization purposes, both SQR and SSI need the rank no less than the number of unstable eigenvalues. If an overestimate of the rank is chosen for $P_0^{1/2}$, SQR works fine. SSI, however, may need special additions such as SRR to maintain acceptable convergence.

6. As the number of inputs increases, the stabilization problem for (A, B) improves in conditioning and SQR will need fewer iterations to achieve stabilization. SSI ignores B and the link between A and B and therefore cannot benefit from the increased number of inputs.

5.6 Summary

Based on the discussions in this chapter, we can summarize SQR stabilization algorithm as follows.

Algorithm 7 SQR

1. *Guess the number of unstable eigenvalues of A from prior knowledge about A. Specify an integer l based on this number. Specify an integer N as the maximum steps of SQR iteration (5.5) and (5.6). Specify an integer $K < N$ and an integer $k \ll K$.*

2. *Generate a random vector v. Generate $n \times l$ initial condition $P_0^{1/2}$ with Arnoldi method on A' and v.*

3. *For i from 1 to N,*

   (a) *perform an SQR iteration (5.5) and (5.6). Compute feedback F_i from $(R_i^{\epsilon})^{1/2}$ and $\tilde{K}_i$. If $\|F_i - F_{i-1}\|_2 < \epsilon$, stop. Iteration termination can also be determined by the next two steps.*

   (b) *For any i such that $i \bmod K < k$, first form the orthogonal basis Q_i of $P_i^{1/2}$ with economical QR decomposition. Extract approximate unstable invariant subspace V_i from Q_i with Stewart's SRR refinement. From the successive k V_i, only the V_i with minimum subspace residual is chosen and is used in next step.*

   (c) *For the V_i computed from above step, compute eigenvalues (all should be unstable) of $V_i'AV_i$ and denote their reciprocals (inside the unit circle) as ME_i. Compute the eigenvalues of $V_i'(A - BF_i)V_i$ and denote them as E_{A-BF_i}. Let $\Delta V_i = dist(ME_i, E_{A-BF_i})$ where $dist(X, Y)$ is defined by $\max_{x \in X}(\min_{y \in Y} |x - y|)$. If rank of V_i is l or becomes stable and ΔV_i is 10^{-3} or less, stability of $A - BF_i$ can be signaled and iteration can be terminated.*

4. *If SQR iteration is terminated because i reaches N, first we check if V_i converges in the metric of subspace residual. If V_i converges, we switch to using a projection method. Otherwise, we can try to extract a converged subspace from Q_i with Stewart's SRR refinement, compute feedback F_N from SQR or from projection onto the converged subspace, then update A to $A - BF_N$ and restart SQR.*

When generating the initial $P_0^{1/2}$ with the Arnoldi method, we can preprocess the starting vector v with the power method $(A')^k v$ so that components of stable eigenvectors of A in v and $P_0^{1/2}$ will be weakened. $P_0^{1/2}$ can be generated as random matrix.

For simplicity, ΔV_i can be computed as the norm of difference between ME_i and E_{A-BF_i} where elements of ME_i and E_{A-BF_i} are sorted. The tolerance level of ΔV_i depends on the minimum distance of eigenvalues of A to unit circle. If this distance is not less than 0.02, tolerance can be taken as 10^{-2}. If this distance is not less than 0.002, tolerance can be taken as 10^{-3}. If this distance is too small, SQR might not be a good choice to stabilize these eigenvalues close to unit circle. Instead, we can filter out larger unstable eigenvalues by deflation as described at the last step of above algorithm and leave the tough eigenvalues to a projection method or the algorithm discussed in Chapter 6.

We have seen that the SQR algorithm is an improvement in technology that is superior to SSI-based methods for many problems. As with projection methods, SQR only moves unstable eigenvalues of A (in this case to their reciprocals). The rank of the SQR and SSI algorithms is determined by the number of unstable eigenvalues of A and the distribution of eigenvalues of A. For a well-conditioned system with only few unstable eigenvalues that are well-separated from unit circle, SQR can stabilize the system quickly. If there exists a circle with radius $r < 1$ such that the number of eigenvalues outside the circle is very small, scaling A can accelerate stabilization process significantly at the cost of possibly increasing the rank of the algorithm and thereby requiring a more careful rank choice heuristic. Because SQR can generate the same subspace sequence as SSI, in case the SSI residual convergence is much faster than SQR feedback convergence, we can always implement a low order QR decomposition to extract an approximate unstable invariant subspace from SQR and then continue with a projection method.

When using SQR, with or without scaling, we have noticed that the system is stabilized much earlier than any traditional stopping criterion is reached. The

ΔV criterion has improved this situation but, further research on efficient stopping criteria is needed.

With SQR we have successfully created an algorithm that integrates the two phases of projection methods and improves the convergence rate by reducing the sensitivity to the choice of rank of the algorithm. For a less well-conditioned system which cannot separate unstable eigenvalues from stable eigenvalues, and the number of eigenvalues close to unit circle is significant, or for a system that has many unstable eigenvalues, the rank of both SQR and SSI is too high, or the number of iterations required for stabilization is too large. The large rank of SQR and SSI implies a lack of scalability in the algorithm even though it is known that there are well-conditioned stabilization problems with large numbers of unstable eigenvalues. In Chapter 6 we develop a low-rank stabilization algorithm that deals with these problems more efficiently.

CHAPTER 6

CHANDRASEKHAR ALGORITHM CSQR

The SQR algorithm and its variants in Chapter 5 effectively addressed two of the drawbacks of projection methods such as SSI: convergence sensitivity to the choice of rank and a separate projection/stabilization phase. SQR shares with projection methods a major scalability problem however: the dependence on a basis of at least the order of the unstable invariant subspace. Increasing either the order of the system with a fixed unstable subspace dimension or the unstable subspace dimension with a fixed system order can lead to excessive computational or storage requirements.

In this chapter, we discuss adapting the well-known Chandrasekhar algorithm to create a low-rank stabilization algorithm based on the DRDE with $P_0 = 0$. The resulting stabilization algorithm, denoted CSQR, is a true low-rank algorithm in that it produces the feedback without requiring the maintenance of a basis for the unstable subspace thus reducing significantly computation and storage. We first present the basic idea of the Chandrasekhar square root algorithm from the literature and then adapt it for the stabilization problem using the appropriate theory from Chapter 3. Preprocessing techniques that can save computations and accelerate convergence significantly are also presented. Two modifications of CSQR, restarted CSQR and scaled CSQR, are then presented. They are shown to save a very significant number of computations for problems that are very difficult to stabilize with SSI, SQR and the general form of CSQR. Finally, a new stabilization stopping criterion that detects the stabilization of the closed-loop system as early as possible during the iterations is used in conjunction with restarted CSQR and scaled CSQR.

6.1 The Basic Chandrasekhar Algorithm

The Chandrasekhar algorithm, a square root recurrence, was introduced in [13] by Morf and Kailath for linear least squares estimation in a discrete-time state space system. It is based on the relationships among $\delta P_i = P_{i+1} - P_i, i = 0, 1, \cdots$ of the DRDE.

Theorem 3.3, Corollary 3.2 and Equation (3.5) of Chapter 3 show that the rank of δP_i is non-increasing, and that δP_i is positive semidefinite if δP_0 is positive semidefinite. So, if δP_0 is low-rank, we can keep all δP_i low-rank. The idea of the Chandrasekhar algorithm is to compute δP_i, rather than P_i, in each step of iteration and update other related and required variables such as feedback.

A general efficient algorithm for arbitrary initial P_0 and Q is difficult to develop. For stabilization purposes, however, we have freedom to choose P_0 and Q, so, we only focus on the special case where δP_0 is low-rank and positive semidefinite. Since δP_i is of dimension $n \times n$, it is not practical for large system to compute δP_i directly. Under the assumption that δP_0 is positive semidefinite, we can decompose $\delta P_0 = L_0 L_0'$ and $\delta P_i = L_i L_i'$ where L_i has dimension $n \times l$ and l is the rank of L_0. This implies that L_i should be able to keep the same rank for all $i = 0, 1, \cdots$.

In order to derive CSQR, rewrite the DRDE (3.2) in the form

$$P_{i+1} = A' P_i A - \tilde{K}_i \tilde{K}_i' + Q \tag{6.1}$$

using the definitions of Chapter 5. Subtracting two consecutive iterations yields

$$L_i L_i' + \tilde{K}_i \tilde{K}_i' = A' L_{i-1} L_{i-1}' A + \tilde{K}_{i-1} \tilde{K}_{i-1}'. \tag{6.2}$$

To update L_i, only L_{i-1} and $\tilde{K}_{i-1}$, $\tilde{K}_i$ are involved. They are all low-rank if l and p are small. Equation (3.17) of Chapter 3 implies we need only know δP_{i-1} or its square root L_{i-1} to update F_i. Unfortunately, $\tilde{K}_i$ is on the same side of L_i, that means we can not directly use Equation (6.2).

An effective recurrence can be derived from the following matrix decomposition of

$$\begin{pmatrix} R + B'P_iB & (A'BP_i)' \\ A'BP_i & \tilde{K}_{i-1}\tilde{K}'_{i-1} + A'L_{i-1}L'_{i-1}A \end{pmatrix} \tag{6.3}$$

which can be rewritten (with definitions of $L_i,\ \tilde{K}_i,\ R_i^\epsilon$) as

$$\begin{pmatrix} R^\epsilon_{i-1} + B'L_{i-1}L'_{i-1}B & (\tilde{K}_{i-1}(R^\epsilon_{i-1})^{1/2} + A'L_{i-1}L'_{i-1}B)' \\ \tilde{K}_{i-1}(R^\epsilon_{i-1})^{1/2} + A'L_{i-1}L'_{i-1}B & \tilde{K}_{i-1}\tilde{K}'_{i-1} + A'L_{i-1}L'_{i-1}A \end{pmatrix}. \tag{6.4}$$

Using Equation (6.2), Equation (6.3) can be rewritten as

$$\begin{pmatrix} R^\epsilon_i & (\tilde{K}_i(R^\epsilon_i)^{1/2})' \\ \tilde{K}_i(R^\epsilon_i)^{1/2} & \tilde{K}_i\tilde{K}'_i + L_iL'_i \end{pmatrix}. \tag{6.5}$$

Decomposing Equations (6.4) and (6.5) into their square root forms, yields an equation that is similar to the SQR algorithm we have seen in Chapter 3 called the Integrated form of the Chandrasekhar algorithm [13]:

$$\begin{pmatrix} (R^\epsilon_{i-1})^{1/2} & B'L_{i-1} \\ \tilde{K}_{i-1} & A'L_{i-1} \end{pmatrix} U_i = \begin{pmatrix} (R^\epsilon_i)^{1/2} & 0 \\ \tilde{K}_i & L_i \end{pmatrix} \tag{6.6}$$

where U_i is an orthogonal matrix of dimension $(p+l) \times (p+l)$.

It is not necessary to implement a full QR decomposition in Equation (6.6). We need only perform a block QR decomposition on the first row of the left side of Equation (6.6) and update the second row accordingly. This leads to the two-stage CSQR:

$$\begin{pmatrix} (R^\epsilon_{i-1})^{1/2} & B'L_{i-1} \end{pmatrix} U_i = \begin{pmatrix} (R^\epsilon_i)^{1/2} & 0 \end{pmatrix} \quad \text{QR stage} \tag{6.7}$$

$$\begin{pmatrix} \tilde{K}_{i-1} & A'L_{i-1} \end{pmatrix} U_i = \begin{pmatrix} \tilde{K}_i & L_i \end{pmatrix} \quad \text{Update stage} \tag{6.8}$$

To compute U_i, the QR decomposition in QR stage is applied to a matrix of dimension $(p+l) \times p$. In second stage, U_i is used to update L_i and $\tilde{K}_i$. F_i can be updated in two ways. One is via $F'_i = \tilde{K}_i(R^\epsilon_i)^{1/2}$, where we waste one step of L_i.

The second uses the formula $F_{i+1} = F_i + F_A \delta P_i$ where $F_A = R^{-1}B'(A')^{-1}$ and the disadvantage is the computation of F_A.

A one step error analysis for optimal filtering version Chandrasekhar algorithm is done in [27]. The authors indicate that the accuracy from Chandrasekhar algorithm is inferior to other type square root algorithms to compute optimal filter because the error in P_i is cumulative from L_i. Conventional wisdom also says that the Schur method with scaling is thought to be superior to all of the square root methods. In our experience, however, we have had no significant difficulties with CSQR. In fact, when used to solve the example DREs in [28] in MATLAB, CSQR produced much higher precision than the Schur method with scaling. Further work is therefore needed on the numerical stability and accuracy of CSQR for stabilization compared to the other methods.

6.2 Efficiency of CSQR

The most appealing feature of the CSQR of Equations (6.7) and (6.8) is the low-rank of L_i. First, Equations (6.7) and (6.8) can keep the dimension of L_i the same as the dimension of L_0. Other types of square root algorithms, for example, Penzl's algorithm in [36] to solve Lyapunov equations, Verlaan's reduced rank square root (RRSQRT) in [37] and the truncated square root algorithm (TSQR) in Chapter 7, truncate a much larger matrix in each iteration. This usually requires much more computation than CSQR and may lose some important information due to truncation even for the special case $Q = 0$.

A second positive factor of CSQR is that to compute a stabilizer F, only a low-rank L_0, or a low rank δP_0 is needed. To obtain such a δP_0, P_0 is chosen (implicitly or explicitly) as a solution of a DRE with the same parameters (A, B). For example, assume we have P_0 that satisfies

$$P_0 = A'(P_0 - P_0 B(R + B'P_0 B)^{-1}B'P_0)A + Q_0 \tag{6.9}$$

for some R and Q_0 (maybe not positive semidefinite). Take the same R and let Q_1 be positive semidefinite such that $Q = Q_0 + Q_1$ is positive semidefinite. For the DRDE with parameters (A, B, R, Q) and initial condition P_0, we have $\delta P_0 = Q_1$ which is low-rank if Q_1 is low-rank. For any choice of P_0, Q and R such that δP_0 is positive semidefinite, there is a Q_0 which satisfies Equation (6.9). A special case is $P_0 = 0$, where we can choose any low-rank positive semidefinite Q such that $\delta P_0 = Q$ is low-rank. Unless specified, P_0 is chosen as 0 when we discuss the CSQR and we choose $L_0 = C$ where $Q = CC'$.

The low-rank CSQR has a significant advantage in computation costs over SSI and SQR for the purpose of stabilization. From Chapters 4 and 5, the rank of SSI and SQR should be at least the number of unstable eigenvalues of A. For CSQR, unless A has some unstable eigenvalues with multiplicity larger than 1, we can choose C with rank 1, even if the system has many unstable eigenvalues. Theorem 3.1 gives the condition on C for the DRDE to reach a stabilizing solution – the detectability of (C, A), or equivalently, the stabilizability of (A', C). In other words, the full rank of $(C, A' - \lambda I)$ for any unstable eigenvalue λ of A is required, which is independent of the number of unstable eigenvalues.

Unless λ is an eigenvalue with multiple eigenvectors where we need the rank of C the same as the multiplicity of λ, we can choose C as a vector. Without any prior knowledge about the multiplicity of unstable eigenvalues of A, at worst, we can choose the rank of C the same as p, the number of inputs. On the other hand, the implication of stabilizability of (A, B), from Theorems 3.2 and 3.3, is whatever C ($Q = CC'$) is chosen, the corresponding DRDE starting from $\delta P_0 \geq 0$ will converge and will be bounded. Therefore CSQR will converge even if we do not get a stabilizing solution, and deflation can be used for stabilization if necessary.

Assume L_i has l columns. Examination of the computations required in Equations (6.7) and (6.8) reveal a significant saving in operations compared to the SQR and SSI methods. The computation $(B', A')L_{i-1}$ requires $O(\mu l)$ operations where μ is the number of nonzero elements in the sparse matrix (B', A'). The QR factorization of $\begin{pmatrix} (R^{\epsilon}_{i-1})^{1/2} & B'L_{i-1} \end{pmatrix}$ requires $O((p+l)p^2)$ operations. Updating L_i and $\tilde{K}_i$ requires $O(n(p+l)^2)$ operations. The total computation cost per iteration is therefore

$$O(\mu l) + O((p+l)p^2) + O(n(p+l)^2)$$

operations. The storage requirements (in addition to that for A and B) include space for $(R^{\epsilon}_i)^{1/2}, \tilde{K}_i$ and L_i (F_i can be computed from $(R^{\epsilon}_i)^{1/2}$ and $\tilde{K}_i$) and total $p^2/2 + n(p+l)$ locations.

The advantage of Chandrasekhar algorithm for large sparse systems is clear, especially for single input system and $l = 1$.

Therefore, the general version of the CSQR algorithm for stabilization is:

Algorithm 8 General CSQR

1. **Choose C:** *Specify an integer l (between 1 and p, usually 1). Randomly generate C with dimension $n \times l$.*

2. **QR and Update:** *Specify a number N and a tolerance ϵ. Let $L_0 = C$, $(R^{\epsilon}_0)^{1/2} = I$, $\tilde{K}_0 = 0$ and $F_0 = 0$ accordingly. Run the two stages CSQR (Equations (6.7) and (6.8)). Compute the 2-norm of $\Delta F_i = F_i - F_{i-1}$. Iterations stop if: (1) $\|\Delta F_i\|_2$ reaches tolerance ϵ, or (2) iteration number reaches N.*

3. **Restart/Deflation:** *If QR and Update stops because of the second reason, or if QR and Update stops because of the first reason but $A - BF_i$ is not stable, update A to $A - BF_i$. Choose a new C and back to QR and Update stage. Otherwise the iterations complete successfully.*

6.3 Results for the General CSQR Algorithm

Example 10 illustrates the low-rank (rank 1) general CSQR applied to a well-conditioned system that has a few unstable eigenvalues and all eigenvalues are well-separated from unit circle. In this chapter, a system (A, B) is well-conditioned means that there exists a stabilizing feedback F such that the eigen-condition of closed-loop $A - BF$ is good.

Example 10 *The parameters (A, B) are the same as in Example 7. C is a normalized random vector. Figure 6.1 and Figure 6.2 show the closed-loop spectral radius convergence and feedback convergence (measured by $\log_{10}(\|F_i - F_{i-1}\|_2)$) respectively.*

A system (A, B) being well-conditioned does not necessarily mean we have an easy and fast way to find a stabilizer. As we know from Chapter 3, the convergence of the DRDE depends on the spectral radius of the corresponding closed-loop matrix $A - BF_\infty$ when projected onto the subspace spanned by P_∞. The CSQR is only a square-root implementation of DRDE, so, its convergence also depends on the same factor. Example 11 shows a difficult problem where the system (A, B) is well-conditioned, but has many of unstable and stable eigenvalues close to the unit circle.

Example 11 *The system (A, B) is the same as in Example 8. We have seen the limitations of SSI and SQR in Chapter 5. Here C is generated via Arnoldi with A' and a normalized random starting vector v.*

Figure 6.3 shows that rank 1 CSQR requires about 400 steps of iterations for the closed-loop spectral radius to start to converge. For rank 2 and 4 CSQR, we need about 300 steps where the computations are equivalent to 600 steps and 1200 steps rank 1 CSQR respectively.

Figure 6.4 shows the feedback convergence rates are similar but, the higher rank CSQR takes fewer iterations and reaches the (last) peak of the plot earlier. For rank 1 CSQR, the feedback needs more than 600 steps of iterations to reach accuracy 10^{-6}, which is, of course, too costly.

For a system (A, B) as in Example 11, we have seen all of the algorithms discussed thus far in the thesis have significant trouble stabilizing efficiently. To accelerate the stabilization process, both SSI and SQR need to enlarge the rank (of initial subspace or $P_0^{1/2}$) significantly. In the remainder of this chapter, we explore methods to improve the general CSQR to get faster convergence or faster stabilization while maintaining the rank, l, of the algorithm.

6.4 Choosing the Matrix C

When using the DRDE to solve a stabilization problem, we have two free parameters: R and Q (or C). For single-input systems, changing R only changes the scale of F. For multiple-input systems, changing R is equivalent to transforming B. So, usually for stabilization, we take R as an identity matrix to simplify the computations. The choice of C, however, is the major concern when tuning CSQR, which will determine the speed of convergence and stabilization. For a given C or Q, the corresponding sympletic form generalized eigenvalue problem $Mz = \lambda Lz$ where

$$M = \begin{pmatrix} A & 0 \\ -Q & I \end{pmatrix} \quad and \quad L = \begin{pmatrix} I & BR^{-1}B' \\ 0 & A' \end{pmatrix}$$

relates the spectrum of the closed-loop matrix to A and B. The stable generalized eigenvalues are the eigenvalues of the closed-loop matrix. This does not, however, yield a practical strategy to choose C given (A, B) to achieve a certain spectral radius and/or eigen-condition of the closed-loop matrix. Unfortunately, although there has been a tremendous amount of research on the DRE, we are not aware of any useful result on how C (or Q) affects the convergence rate of the DRDE and the spectrum of the corresponding closed-loop matrix.

We have already mentioned that we only need the rank of C large enough to guarantee the stabilizability of (A', C) in order to obtain a stabilizer F. In fact, for convergence, we do not even need that restriction but, stabilization would have to be done via a series of deflation steps. Example 11 shows that higher rank of C usually tends to achieve faster stabilization by reducing the number of iterations and a better spectral radius of closed-loop matrix results. It is also clear however from the same example that the total computation cost for the higher rank C to reach a point where feedback convergence reaches a specified level is also larger. So, in general, increasing the rank of C cannot be expected to always improve things significantly for difficult problems. There are, of course, some special situations where this is not the case. In particular, if the unstable eigenvalues are well-separated then when the preprocessing discussed below is done it is often worthwhile to increase the dimension of C slightly.

In the general CSQR algorithm, C is chosen randomly. No information from the system (A, B) is used to guide the choice. Intuitively, if we can concentrate only on the unstable eigenvalues and at most the near unstable eigenvalues, we should reach a stabilizer much faster. Consider, for example, the special case where $A = \begin{pmatrix} A_s & A_{22} \\ 0 & A_u \end{pmatrix}$ such that all unstable eigenvalues are located in A_u and partition C and L_i as follows $C = \begin{pmatrix} C_s \\ C_u \end{pmatrix}$ and $L_i = \begin{pmatrix} L_{si} \\ L_{ui} \end{pmatrix}$.

Starting from $L_{s0} = C_s = 0$, we can get $L_{si} = 0$ for all $i > 0$. If we decompose $B = \begin{pmatrix} B_s \\ B_u \end{pmatrix}$, in the QR stage, we have $B'L_{i-1} = B'_u L_{u(i-1)}$. It follows that the orthogonal matrix U_i and all $\tilde{K}_i$ and L_i depend only on the unstable components.

We can, therefore, treat the two-stage CSQR as a reduced version CSQR with parameters (A_u, B_u, C_u) from the projection onto the unstable invariant subspace of A'. Furthermore, if $C_s = 0$, the result P_∞ will only contain unstable components of A', and the feedback convergence rate of CSQR will be determined by the spectral radius of the reduced CSQR obtained by projecting CSQR on the subspace spanned by P_∞. For a system (A, B) as in Example 10 where unstable eigenvalues of A are well separated from unit circle, focusing C on unstable components of A means much faster convergence rate, as seen in Example 13 and Figure 6.8.

A simple preprocessing strategy can steer C in the direction of the unstable space. Choose a small integer K, e.g., 10, and a random $n \times l$ matrix C_0. Let C be the orthogonal base of $(A')^K C_0$. This is easily accomplished via the power method or

(SSI if $l > 1$). The cost of K steps of preprocessing by the power method is roughly equivalent to K CSQR iterations. Amazingly, a small investment of $K = 5$ or 10 tends to result in a significant reduction of the number of iterations required for stabilization.

Example 12 *Using the same (A, B) pair as in Example 11, we choose C_0 as the normalized random vector we used as C in Example 11. This vector is then preprocessed to yield C as a normalized vector of $(A')^K C_0$.*

Figure 6.5 shows that with $K = 10$ preprocessing steps, the system is stabilized within 200 steps as opposed to within 400 steps without preprocessing C. Moreover, the spectrum of closed-loop from CSQR with preprocessing C is not as close to unit circle as without preprocessing C. Figure 6.6 shows that the feedback with preprocessing C converges much faster ($\|F_i - F_{i-1}\|_2$ reaches 10^{-7} within 550 steps with preprocessing C, 700 steps without preprocessing C). Further preprocessing C with more steps of power method does not help. As we see from both figures, the result from $K = 20$ is almost the same as the result from $K = 10$. Note that this is the same example for which increasing the rank of CSQR reduced the number of iterations. However, here we have saved considerable work over the rank 4 CSQR.

Not all examples are improved by preprocessing. In the next example we see the relatively easy problem of Example 10 does not stabilize in fewer iterations.

Example 13 *Using (A, B) from Example 10 and preprocessing as in Example 12 yields the results in Figures 6.7 and 6.8. Figure 6.7 shows that CSQR requires 12 more steps to obtain a stabilizing solution. There has however been improvement in the convergence of the feedback. Figure 6.8 shows the feedback from the CSQR after preprocessing C converges much faster and $\|F_i - F_{i-1}\|_2$ reaches 10^{-15} in 40 steps with preprocessing C as opposed to 123 steps without preprocessing C.*

Note that the "good" and "bad" problems have switched roles here compared to the general CSQR algorithm. This is easily understood. If we apply $(A')^k$ on an $n \times 1$ C, when $k \to \infty$, at most we have 1 dominant eigenvector left. In Example 13, unstable eigenvalues are well-separated. After several steps, the components of the eigenvectors in C except the largest one have vanished, so it takes longer time to stabilize these non-dominant but unstable eigenvalues. This problem is one that could benefit from increasing the rank of C to 2 or 3 without increasing the overall complexity unacceptably.

The improved convergence of the feedback seen in Figure 6.8 with preprocessing C is also easily understood. In Example 13, with a random choice of C, the closed-loop spectral radius dominates the feedback convergence. After several steps preprocessing on C with the power method, the components of stable eigenvectors in C vanish, and the spectral radius of reduced problem CSQR obtained by projecting CSQR onto the unstable invariant subspace of A that dominates the feedback convergence of CSQR.

The final aspect of the choice of C is its magnitude. For the scalar DRE, large Q means small $A - BF$. If $Q \to \infty$, $A - BF \to 0$. Since the convergence of DRDE depends on the spectral radius of closed-loop $A - BF$, intuitively larger C means faster convergence. More precisely, if $C \to \infty$, (or $Q \to \infty$), the eigenvalues with eigenvector v such that $Qv \neq 0$ should be mapped to 0. (Note that this does affect the relationship of the closed-loop spectrum to the original spectrum.) There is a limit on the number of eigenvalues that can be mapped to 0. Example 14 show that the number of eigenvalues that vanish in the closed-loop matrix is bounded by the number of inputs. In this example, C is taken as a full rank square matrix, so, if we choose C as a low rank matrix with large norm, we could expect less eigenvalues of closed-loop to vanish. The implication is that the usefulness to control convergence rate of CSQR via increasing the norm of C is limited and in fact may be counterproductive. For example, if we enlarge the C used in Example 12 by 10 times, the stabilization process takes longer time to reach a stabilizer.

Example 14 Use RANDN of MATLAB to generate a 10×10 matrix A and a $10 \times p$ matrix B, where p is taken between 1 and 10. C is chosen as $10000I$. When applying CSQR to find the stabilizing feedback F, we find that exactly p eigenvalues of $A - BF$ are close to zero. Note this is not a limitation of the stabilization algorithm. With the deflation method of Chapter 2, even if $p = 1$, we can move all eigenvalues of A to zero one by one. (These eigenvalues must be defective however due to the theoretical limitations of pole placement.) So, the limitation on the number of eigenvalues that can be moved close to zero is from the DRE, not from the stabilization algorithm.

On the other hand, if the norm of C is very small, we can expect the solution P should close to the solution with $Q = 0$ which is the solution with the minimal norm. When using the DRDE starting from $P_0 = 0$ as we do with CSQR, and considering the norm consistent to spectral radius, the upper bound of $\|P_i\|$ is $(1 + \cdots + \|A\|^{2i})\|Q\|$. Now consider a problem where $\|P_\infty\|$ is large and the largest eigenvalue of A is close to 1. This is one of the known difficult problems where the system is only slightly to moderately unstable. Since we start from $P_0 = 0$ we must build P_i up to achieve the large final norm and given the bound above this must take many iterations. So, we cannot take C with too small magnitude and expect any improvement on difficult problems. We must, therefore, look to other acceleration strategies to improve CSQR.

6.5 Restarted CSQR

We observed that preprocessing C via the power method or SSI sometimes increases the computational cost for CSQR to reach a stabilizer F. Usually, this happens when there is large separation of unstable eigenvalues and the number of unstable values is larger than the rank of C. Since the power method tends to strengthen components of dominant eigenvalues, smaller unstable eigenvalues have almost non-significant components left in C, so, the smaller eigenvalues need more iterations to stabilize. Enlarging the size of C can keep components of more unstable eigenvalues. For example, if the size of C is just the number of unstable eigenvalues, preprocessing C with subspace iteration is close to projecting CSQR on unstable invariant subspace and the resulting algorithm is an interesting combination of the methods described thus far in this thesis. However, as we have learned from Example 14 and Example 10, the benefit of enlarging the rank of C might not worth the price of increasing total computations.

On the other hand, the DRDE takes a long time to stabilize eigenvalues that are close to unit circle (e.g., Example 10), even though other large unstable eigenvalues have been stabilized. It is therefore worth examining the removal of the larger eigenvalues once they are stabilized so that the stabilization process can concentrate on eigenvalues close to unit circle. If there are very few very large unstable eigenvalues (say 1 or 2), we can use the deflation algorithm from Chapter 2 to stabilize these large eigenvalues explicitly. If many large unstable eigenvalues exist, we can run CSQR iterations for a some number of steps to stabilize the large unstable eigenvalues implicitly and then filter out these large unstable eigenvalues by updating A to $A - BF$ and apply CSQR on $(A - BF, B)$. This notion is the basis for the restarted CSQR algorithm.

The restarted CSQR is simply:

1. Run several steps of CSQR.
2. Update A to $A - BF$ (this is not done explicitly of course so as not to destroy the sparsity of A).
3. Choose a new C and go to step 1.

There are three basic problems that must be addressed for this algorithm to work:

1. when to restart;
2. how to choose a new C;

3. when to stop the restarted CSQR iteration.

It is difficult to determine when to restart. One extreme is the general CSQR, where restarting occurs only after a large number of iterations and no termination criteria are met. The other extreme is to choose a new C and restart on every CSQR iteration. The first approach cannot take any advantage of restart algorithm. If the stabilization problem is well-conditioned and easy, no restart should be needed. If the problem is tough, the general CSQR might restart too late and waste many computations. The second approach tends to stabilize too much. The spectral radius of closed-loop $A - BF$ tends to be too small and the eigen-condition of closed-loop tends to be poor. Further research is definitely required to determine the relationship between the optimal number of steps between restarts to achieve fast stabilization and good eigen-conditioning of the final $A - BF$ from factors such as the number of unstable eigenvalues and near unstable eigenvalues, the distance of eigenvalues of A from unit circle, etc.

Figures 6.9 and 6.10 illustrate the results for Example 15. When the number of steps between restarts is smaller, the spectral radius of $A - BF_i$ is smaller and eigen-condition number is larger. Nevertheless the stabilization process is faster when monitored by a stabilization criterion discussed later in this chapter. The 50 steps between restarts used in this example seems a good choice in practice. It stabilizes the system fast enough with reasonable eigen-conditioning of the final closed-loop matrix.

There are four ways to choose new C at the beginning of each round of CSQR iterations. We can keep the same C for all rounds, or randomly choose a new C with the method we have discussed for general CSQR. Both of these do not use any information from previous round of CSQR iterations.

Note, however, that δP_i, in fact, is the residual of DRE with P_i as an estimated solution:

$$P_{i+1} = A'P_iA - A'P_iB(R + B'P_iB)^{-1}B'P_iA + Q$$

which is equivalent to

$$P_i = A'P_iA - A'P_iB(R + B'P_iB)^{-1}B'P_iA + (Q - \delta P_i).$$

The implication is that the reason P_i does not solve the DRE or reaches a stabilizer is that in the current iterations δP_i is not taken into consideration. So, it is better to use L_i which comes from $\delta P_i = L_iL_i'$ to construct the new C.

We have considered two such methods to choose C. The first uses L_i from the previous round; the second takes an orthogonal basis of the $IM(L_i)$. The first choice is better if $\|L_i\|$ is very large. For system like that in Example 11, the basis of L_i is the best choice. All other choices do not work well, especially the ones which do not use any information from previous round.

6.6 A New Stabilization Criterion

It is the stabilization process, not feedback convergence, that the restarted CSQR accelerates. Usually, the feedback convergence with the restarted CSQR is slow, since between each restart we run a new round of general CSQR iterations. Unless the number of iterations in each round is very large, it is not practical to use feedback convergence as a stopping criterion.

The other stopping criteria used for SSI and SQR depend on the availability of a basis of a space for which a residual can be computed or that can be used for the ΔV test. So a new criterion to catch stabilization with the restarted CSQR is needed. We propose to use $\tilde{W}_i = (\prod_{k=1}^{k=i}(A - BF_k))W$. to catch stabilization. The idea is, with a norm that is consistent with the spectral radius once $A - BF_i$ is stable, we should have $\|\tilde{W}_i\| \leq \|\tilde{W}_{i-1}\|$. More than that, if there exist K and $\delta > 0$ such that spectral radius $\rho(A - BF_i) \leq 1 - \delta$ for $i > K$, we should have $\|(\prod_{k=1}^{k=i}(A - BF_i))W\| \to 0$ when $i \to \infty$ with a convergence rate that is at least

$1-\delta$. To implement this criterion, we choose a (full) rank $n \times q$ W: starting from $\tilde{W}_0 = V$, in iteration i, define $\tilde{W}_i = (A - BF_i)\tilde{W}_{i-1}$. Let M_i denote the largest norm (2-norm) of $\tilde{W}_j,\ j = 0, \cdots, i$, and compute $s_i = \log_{10}(\|\tilde{W}_i\|_2 / M_i)$. Unless $\rho(A-BF_i)$ is very close to 1, s_i should show a decreasing trend and we do not need a very small s_i to catch stabilization. When the number of iterations in each round is small, $\rho(A - BF_i)$ tends to move quickly away from 1 and we can expect fast convergence of s_i. The best rank of W should be the number of unstable eigenvalues of A. For most cases, numerical experiments show the difference of s_i between random W and $W = B$ is very small. Example 16 below demonstrates that s_i with B can detect the stabilization of stabilizable eigenvalues and ignore the unstabilizable eigenvalues when (A, B) is not stabilizable. So, we will use s_i starting from B as an inexpensive termination criterion.

The restarted CSQR therefore has the form:

Algorithm 9 The restart CSQR

1. **Choose C:** *Same as step 1 in Algorithm 8. We should preprocess C with the power method or SSI.*

2. **Restart:** *Specify integers N (outer loop) and K (inner loop). Let $A^{(0)} = A,\ L_0^{(0)} = C,\ M_0 = \|B\|_2$ and $VB_0 = B$. For j from 0 to N, run K steps of CSQR (6.6): Let $(R_0^\epsilon)^{1/2}$ and $\tilde{K}_0$ be 0. For i from 1 to K:*

   (a)

$$\begin{pmatrix} (R_{i-1}^\epsilon)^{1/2} & B' L_{i-1}^{(j)} \\ \tilde{K}_{i-1} & (A^{(j)})' L_{i-1}^{(j)} \end{pmatrix} U_i = \begin{pmatrix} (R_i^\epsilon)^{1/2} & 0 \\ \tilde{K}_i & L_i^{(j)} \end{pmatrix}$$

   (b) Compute $F_i^{(j)} = (\tilde{K}_i (R_i^\epsilon)^{-1/2})'$

   and $VB_{jN+i} = (A^{(j)} - BF_i^{(j)}) VB_{jN+i-1}$.

   Let $M_{jN+i} = \max(M_{jN+i-1}, \|VB_{jN+i}\|_2)$ and

$s_{jN+i} = \log_{10}(\|VB_{jN+i}\|_2/M_{jN+i})$. If s reaches a specified stopping criterion, stop.

At the end of each inner loop, let $A^{(j+1)} = A^{(j)} - BF_K^{(j)}$. Choose a new C ($L_0^{(j+1)}$) with one of the four ways we described.

3. *If the iteration succeeds, assemble F from all $F_K^{(j)}$. If not, we can choose a smaller K and repeat the process.*

Example 15 *In this example, we use the same system (A, B) as in Example 11. The initial C is a normalized random vector. In each new round of CSQR iterations, we choose the new C (or L_0) as the orthogonal basis of L_N from the previous round of CSQR iterations. Figure 6.9 and Figure 6.10 show the efficiency of the restarted CSQR with different number of iterations in each round. Here the total number of iterations (inner $\times$ outer) is constant.The solid line, '*', '-.', '.', broken line and 'o' represent the cases where the number of iterations in inner loop is 800, 400, 200, 100, 50, 25, respectively.*

In Figure 6.9, we see that when the number of iterations in inner loop becomes smaller, the eigenvalues of A are stabilized deeper into unit circle and fewer total iterations are used to reach a stabilizer. The cost is the eigen-condition of closed-loop matrix degrades if we do not catch stability as early as possible.

In Figure 6.10, we use s_i to measure the stabilization. Both figures show the smaller the steps in inner loop, the faster the stabilization process. If we compare the computations of no restarting (iterations of each inner loop: 800 here) to the computations with restart after each 50 steps of CSQR iterations, we can see the system is stabilized within about 400 steps without restart and is stabilized within about 200 steps with restart and 200 steps are saved, on the other hand, to get s_i to reach 4 digits (10^{s_i}), we need about 800 steps without restart and 300 steps with *restart. Choosing a smaller number (say 25 here) of steps in inner loop will further save the computations, but the eigen-condition of corresponding closed-loop worsens.*

6.7 Scaled CSQR

The reasons that cause the system (A, B) to be difficult to stabilize are: (1) the stabilization problem is poor-conditioned (usually measured by eigen condition of closed-loop) and (2) some eigenvalues are too close to the unit circle. The first problem is hard to solve and often indicates an inherent ill-conditioning in the stabilization problem. For the second problem, we have used techniques such as preprocessing C and restarting to accelerate the stabilization process. What will happen if we apply a transformation on A? Clearly the main restrictions on such a transformation is that it does not reduce the sparsity of A unnecessarily and that it does not degrade the process of finding a stabilizer. We will see a simple transformation such as scaling A can accelerate stabilization process by as much as an order of magnitude in iterations.

The scaling technique on A is close to the algorithm which assigns eigenvalues to a specified disk discussed in [38] by Furuta and Kim, where the authors use the DRE to stabilize continuous-time systems by shifting and scaling A. When using this method, we scale and shift the system matrix A to $\tilde{A} = (A - \alpha)/r$ and then stabilize $\tilde{A}$. If $\tilde{A} - (B/r)F$ is stable, all the eigenvalues of $A - BF$ are located in disk centered at α with radius r. We will ignore the shift here, so the result disk is centered at 0 with radius r.

For the purpose of stabilization, we need $r < 1$, which means usually that $\tilde{A} = A/r$ will have more unstable eigenvalues than A. The stabilizability of $(\tilde{A}, B/r)$ is the minimum requirement to get a stabilizer F for $(\tilde{A}, B/r)$ and to get the convergence of δP_i in the CSQR iteration when applied to $(\tilde{A}, B/r)$. Scaling, however, has some subtle consequences for stabilizability. Any unstable eigenvalue of A remains stabilizable in A/r by Proposition 3.1. A stable eigenvalue of A may not remain stabilizable for A/r. Since stable eigenvalues may become unstable eigenvalues of A/r we must consider what happens to unstabilizable unstable eigenvalues when CSQR is applied. We consider first a simple case.

Assume $A = diag(\Lambda_1, \Lambda_2)$, $B = [B_1', B_2']'$, and $B_2 = 0$. Partition $L_i = [L_{i1}', L`_{i2}]'$ accordingly. The unstable eigenvalues in Λ_1 are stabilizable while those in Λ_2 are not. For any unstable eigenvalue λ_1 in Λ_1 and unstable eigenvalue λ_2 in Λ_2, we assume $|\lambda_1| > |\lambda_2|$. Because $B'L_i = B_1'L_{i1}$ for any i, L_{i2} does not contribute to $B'L_i$ and U_i in the CSQR iteration given by Equations 6.7 and 6.8. The computation of L_i is equivalent to running CSQR on the DRDE based on parameters $(\Lambda_1, B_1, R, L_{01})$. It follows that L_{i1} and P_{i1} (the upper part of P_i) should converge and L_{i2} is dominated by L_{i1} before L_{i1} converges. After L_{i1} converges, $B'L_i \approx 0$ and $L_{(i+1)2} = \Lambda_2 L_{i2}$ will blow up eventually. But F_i will not change. Therefore, F_i will converge.

In other words, if the stabilizable unstable eigenvalues dominate the unstabilizable unstable eigenvalues we expect a feedback that stabilizes the former and ignores the latter while P_i diverges. If an unstabilizable unstable eigenvalue dominates one or all of the stabilizable unstable eigenvalues, however, we expect both the feedback and P_i to diverge. This can be seen from reasoning similar to that above. Example 16 demonstrates these trends of CSQR on a system with unstabilizable eigenvalues.

Example 16 *Let's consider a simple system. Let $A = diag(2, 3)$ and $C = (11)'$. We choose $B = (0, 1)'$ in Case 1 and $B = (1, 0)'$ in case 2. Run CSQR iteration for 50 steps. For Case 1, eigenvalue 3 is stabilized and the feedback converges. For Case 2, eigenvalue 2 is stabilized, but the feedback diverges. Interestingly, Case 2 is one where we expect feedback and P_i to diverge. In fact they do but, the stabilization criterion using the s_i (with B) we have used for the restarted CSQR can detect the stabilization of the stabilizable eigenvalue and ignore the unstabilizable eigenvalue before the blowup of the feedback is too serious. This is not expected for all such* *problems but for problems of both types the stabilization criterion using the s_i can effectively detect stabilization of the stabilizable eigenvalues and ignore the others.*

We have assumed the stabilizability of (A, B). So, the only possible unstabilizable unstable eigenvalues of $\tilde{A}$ come from stable eigenvalues of A after scaling. They are, therefore, dominated by the stabilizable unstable eigenvalues of A after scaling. The implication is, when applying the CSQR on $(\tilde{A}, B/r)$, in theory, the feedback will converge even if L_i diverges. On the other hand, when applying CSQR to $(\tilde{A}, B)$, although the eigenvalues of A will move into a specified disk with radius r, it does not mean the algorithm will converge with rate r and will be faster than unscaled case, since the convergence depends on the spectrum of the closed-loop $\tilde{A} - (B/r)F$ associated with the transformed system $(\tilde{A}, B/r)$. Therefore, we need to consider more carefully the situation typically encountered with CSQR to justify this approach.

The reason that we use the scaling techniques is the observation that with DRDE-based algorithms such as SQR and CSQR the unstable eigenvalues tend to be moved very close to unit circle very quickly if they are not stabilized. This is especially true for large unstable eigenvalues. With scaling $r < 1$, if the spectral radius of $\tilde{A} - (B/r)F_i$ is at most slightly larger than 1, the spectral radius of $A - BF_i$ will be close to r.

While a general convergence criterion such as feedback convergence might not be useful, we can use the stabilization criterion introduced with the restarted CSQR. We do have the possibility that some eigenvalues of $\tilde{A}$ could be unstabilizable because of scaling and therefore we cannot move these eigenvalues at all. As we have learned from Example 16, these unstabilizable eigenvalues of $\tilde{A}$ will not affect the convergence of s_i (from $(\prod(A - BF_i))B$) which will converge with a rate close to r.

On the other hand, we do not need the convergence of s_i. The level of s_i to detect stabilization of (A, B) depends on how far away r is from 1. Unlike the restarted CSQR where the spectral radius of the corresponding closed-loop cannot be controlled, here we can control r, therefore, the level of s_i can be specified. The safe way is, once s_i reaches the specified level, monitor additional K steps (a small integer, for example 10) to see if the decreasing trend is maintained. Empirical evidence shows that the further r is away from 1, the faster s_i converges and the better s_i detects the stabilization. But, once r becomes small, the eigen-condition of the corresponding closed-loop could be considerably poorer. If r is too close to 1, both the stabilization process takes many iterations and the s_i are a poor basis for a termination criterion. Generally speaking, r is better to choose around 0.9, and -1 or -2 are appropriate levels for s_i.

The CSQR with scaling A is summarized as following:

Algorithm 10 CSQR with scaling A

1. *Choose C as in the general CSQR. Use the power method or SSI to preprocess C. Choose r around 0.9. Choose s=-1. Let $M_0 = \|B\|_2$, $VB_0 = B$ and $R = I$. Specify integers N and K.*

2. *Run N steps of the two stage CSQR iterations (6.7) with $(A/r, B/r, R, C)$. In each step compute F_i , $VB_i = (A - BF_i)VB_{i-1}$, $M_i = \max(M_{i-1}, \|VB_i\|_2)$ and $s_i = \log_{10}(\|VB_i\|_2/M_i)$. If $s_i < s$, monitor further K steps to see if s_i continues to decrease. If so, stop. Otherwise, continue.*

Example 17 *We use the values of $A, B,$ and C from Example 11. Figures 6.11 and 6.12 show the efficiency of stabilization via CSQR with scaling on A. The solid line, broken line, "-.", and small dots represent the cases with scale r as 1, 0.95, 0.9 and 0.85 respectively. Figure 6.11 shows the system (A, B) is stabilized in about 400, 120, 80, 60 steps respectively. The stabilization criterion is a little tricky. With large r, we need lower level of s_i. As observed, for $r = 0.85$ or 0.9, a level of -1 is more* *than enough. For $r = 0.95$, a level of -2 is enough but, for r close to 1, we need a deeper level of -4.*

6.8 Stabilizing Many Unstable Poles

In the previous chapters, we have seen that SSI and SQR algorithm cannot be scaled to handle a system with a large number of unstable eigenvalues, since the rank of SSI and SQR is at least the number of unstable eigenvalues. The CSQR algorithm does have much better scalability on the number of unstable eigenvalues, provided that the conditioning of the system (A, B) is not to ill-conditioned. The following example shows that rank 1 Chandrasekhar algorithm can stabilize a system with 48 unstable eigenvalues.

Example 18 *In this example, A is a 100×100 matrix generated by RANDN of MATLAB, which is scaled by 7 so that A has the spectrum shown in Figure 6.13. B is a 100×20 matrix generated by RAND. A multiple-input system is required to keep the conditioning acceptable. Stabilizing large numbers of eigenvalues with a single-input is very difficult. C is a random vector generated by RAND. We use rank 1 CSQR and CSQR with scaling of A by 0.9 to stabilize the system.*

400 steps of CSQR are run. The spectral radii of the $A - BF_i$ are plotted in Figure 6.14 and the stabilization/convergence criteria are plotted in Figure 6.15. We can see that both implementations stabilize A in 100 steps. With scaling, the stabilization can be detected within 100 steps. Without scaling, we need 150 steps to detect stabilization. More steps are needed if we use feedback convergence as the termination criterion.

Table 6.1 shows that the resulting feedback for both scaled and unscaled CSQR is moderate in norm and reasonably robust even though the problem is moderately ill-conditioned.

Table 6.1. Additional data for Example 18

	$\|F\|_2$	$\kappa_2(X)$	$(1-\rho)/\kappa_2(X)$	$r_{\mathbb{C}}(A,B,F)$
Without scaling	1.2157	3.4565×10^3	8.3610×10^{-6}	7.8832×10^{-4}
With scaling	1.5185	1.2311×10^4	1.1022×10^{-5}	0.0042

6.9 Nonzero Initial Guess

So far , we have chosen $P_0 = 0$ when using CSQR. We can also choose P_0 ($P_0^{1/2}$ exactly) nonzero and compute $(R_0^\epsilon)^{1/2}$ and $\tilde{K}_0$ accordingly. This means we use an implicit Q such that δP_0 is positive semidefinite and low-rank. Unfortunately, such a Q might not exist if we only choose $P_0^{1/2}$ randomly, and the numerical experiments do not show any advantages over zero initial guess.

On the other hand, we can take P_0 from a solution of another DRE with the same (A, B, R) but different Q. This idea can be used to restart CSQR in a significantly different manner than those discussed for the restarted CSQR above: on each restart, keep $(R^\epsilon)^{1/2}$ and $\tilde{K}$ from the previous round, choose a new C or L_0 and leave A unchanged. Unfortunately, this type of restarted CSQR does not seem to perform any better than the general CSQR algorithm. This seems to be mainly due to the fact that the eigenvalues of A that have been stabilized are not filtered out.

6.10 Summary of CSQR

Comparing Example 7 to Example 10 and related figures, we can see the general CSQR works at least as well as SSI and SQR for well-conditioned systems that have very few unstable eigenvalues. For less well-conditioned systems that are difficult to stabilize, Example 8, Example 11 and related figures show that the efficiency of the general CSQR is not worse than that of SQR and SSI. While any acceleration method for SQR and SSI needs to increase computation cost significantly, the modifications of CSQR, such as preprocessing C, restarted CSQR and scaled CSQR, can significantly reduce the number of CSQR iterations and beat both SQR and SSI by an order of magnitude.

If the number of unstable eigenvalues increases, SQR and SSI are much more costly since the rank of SQR and SSI must increase and therefore CSQR methods will be more efficient. Moreover, as the number of inputs and the number of unstable eigenvalues increase, stabilization with CSQR remains efficient.

If we consider matrix-vector multiplication $A'v$ as one basic unit of computation, the computation cost to stabilize the system in Example 8 and Example 11, with SQR (rank 4), CSQR, C-CSQR (CSQR with preprocessing C), R-CSQR (restarted CSQR, 50 steps in each round), and S-CSQR (CSQR with scaling A, $r = 0.9$) are summarized in Table 6.2 (All CSQR are rank 1 and C-CSQR includes 10 multiplications for preprocessing C.)

Table 6.2. Complexity for system of Example 8 and Example 11.

Alg	SQR	CSQR	C-CSQR	R-CSQR	S-CSQR
Rank	4	1	1	1	1
Costs	1200	400	310	250	80

The computation costs to reach 5 digits convergence with SQR (rank 4), SSI (rank 4) and the CSQR methods are summarized in Table 6.3. SQR, CSQR, and C-CSQR use feedback convergence as a termination test. SSI uses subspace residual. R-CSQR and S-CSQR use $(\prod_{k=1}^{i}(A - BF_k))B)$.

Table 6.3. Complexity for system of Example 8 and Example 11 using various termination tests.

Alg	SQR	SSI	CSQR	C-CSQR	R-CSQR	S-CSQR
Rank	4	4	1	1	1	1
Costs	>1600	>1600	600	410	350	150

While Tables 6.2 and 6.3 show that the CSQR methods are much more efficient than SQR and SSI, they also demonstrate the most significant problem with the DRDE-based methods – stabilization with both SQR and CSQR methods occurs much earlier than any termination criterion detects. Further research on more effective stopping criteria is needed.

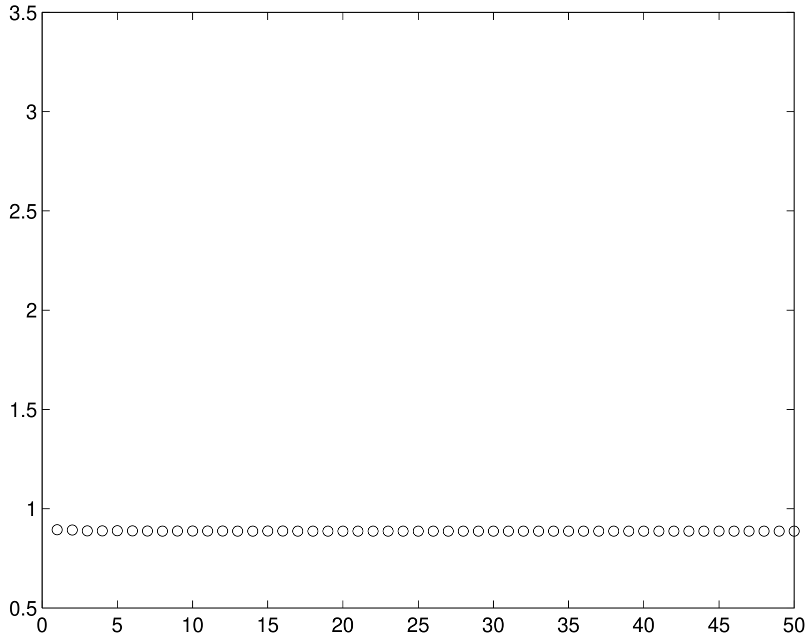

Figure 6.1. CSQR: Spectral radius of $A - BF_i$, Example 10

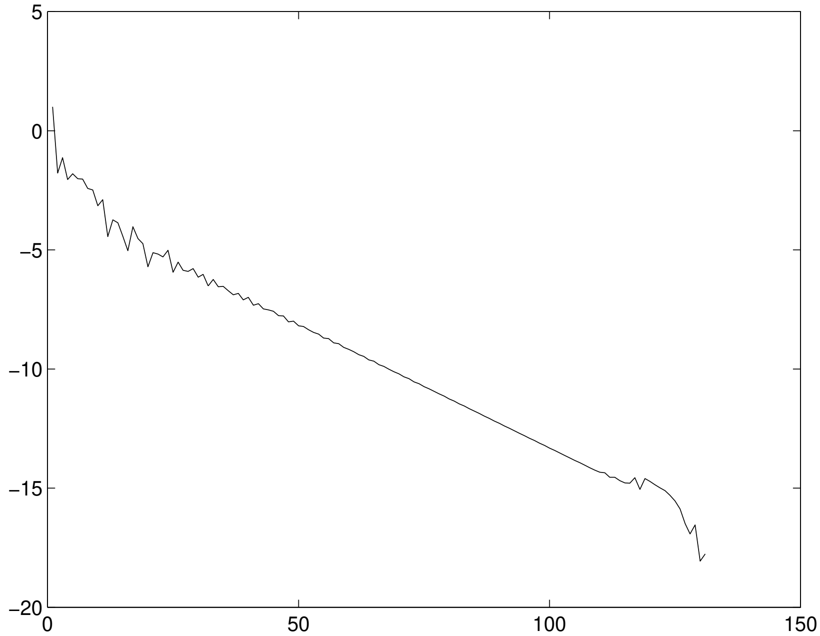

Figure 6.2. CSQR: Feedback convergence, Example 10

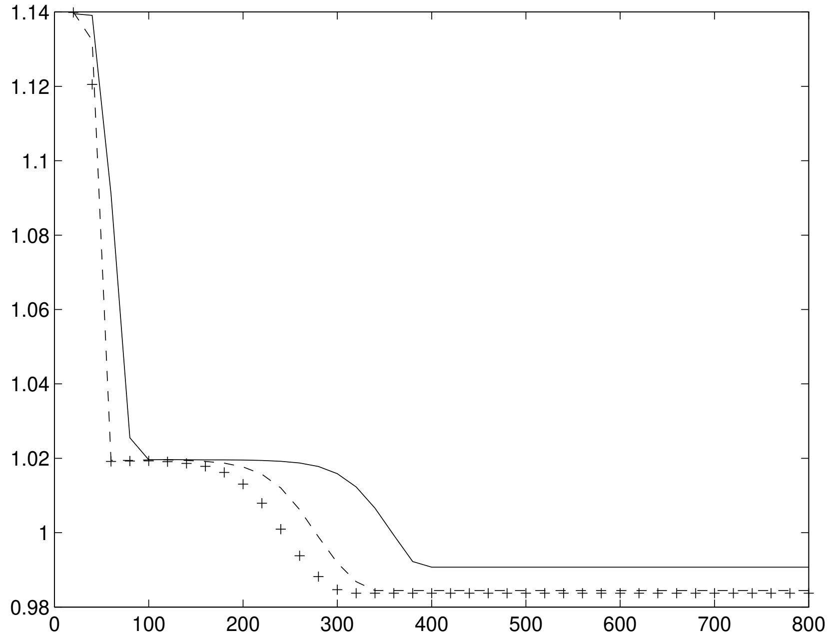

Figure 6.3. CSQR: Spectral radius $A - BF_i$, Example 11 rank 1 (–), rank 2 (- -), rank 4(+)

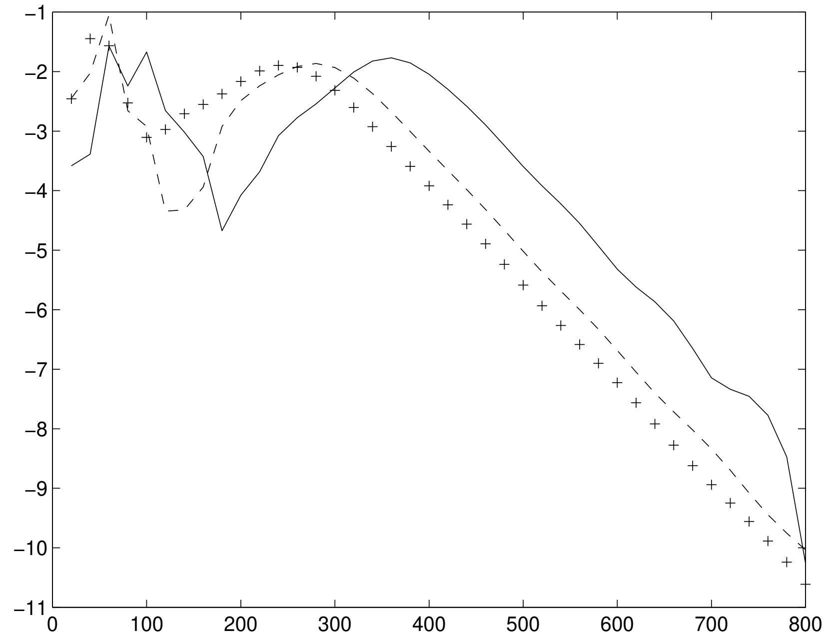

Figure 6.4. CSQR: Feedback convergence, Example 11, rank 1 (–), rank 2 (- -), rank 4(+)

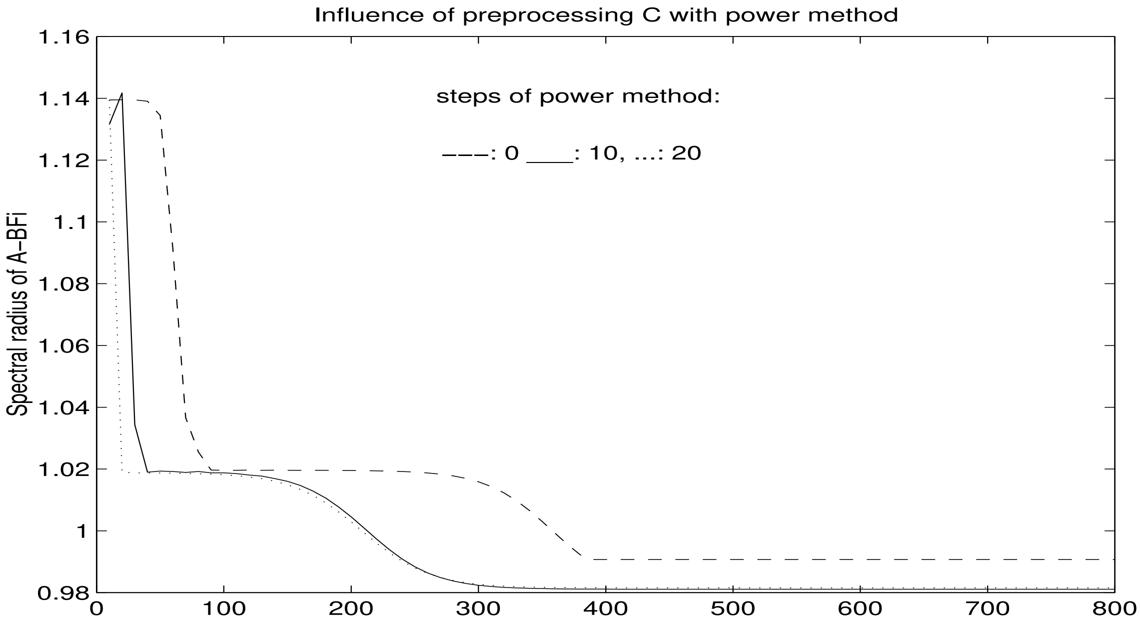

Figure 6.5. CSQR: preprocessing C, Example 12

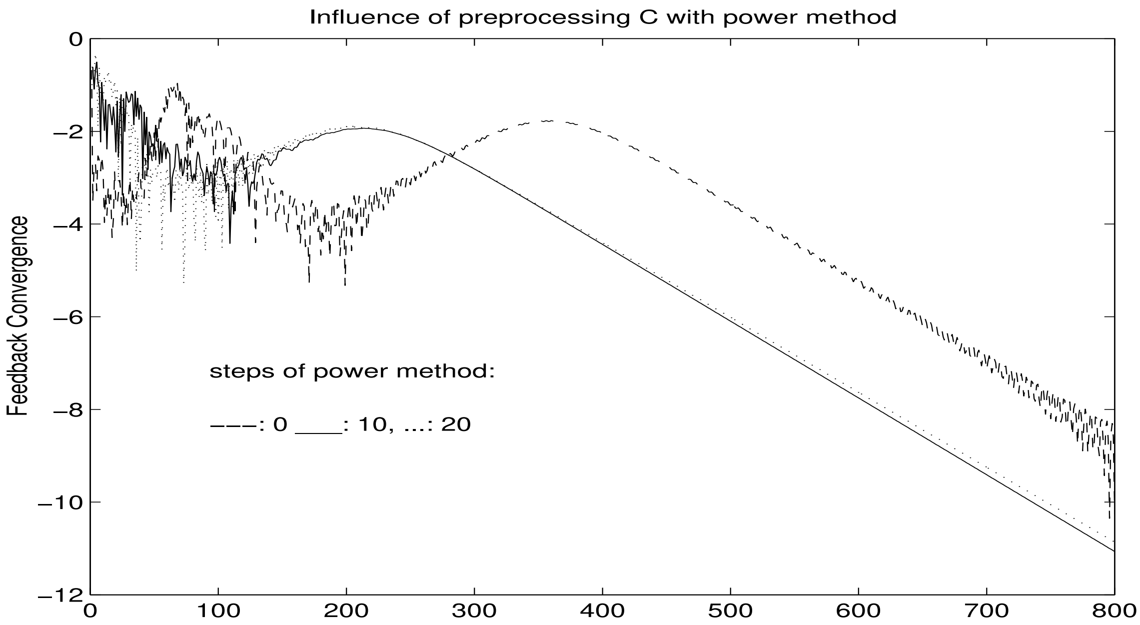

Figure 6.6. CSQR: preprocessing C, Example 12

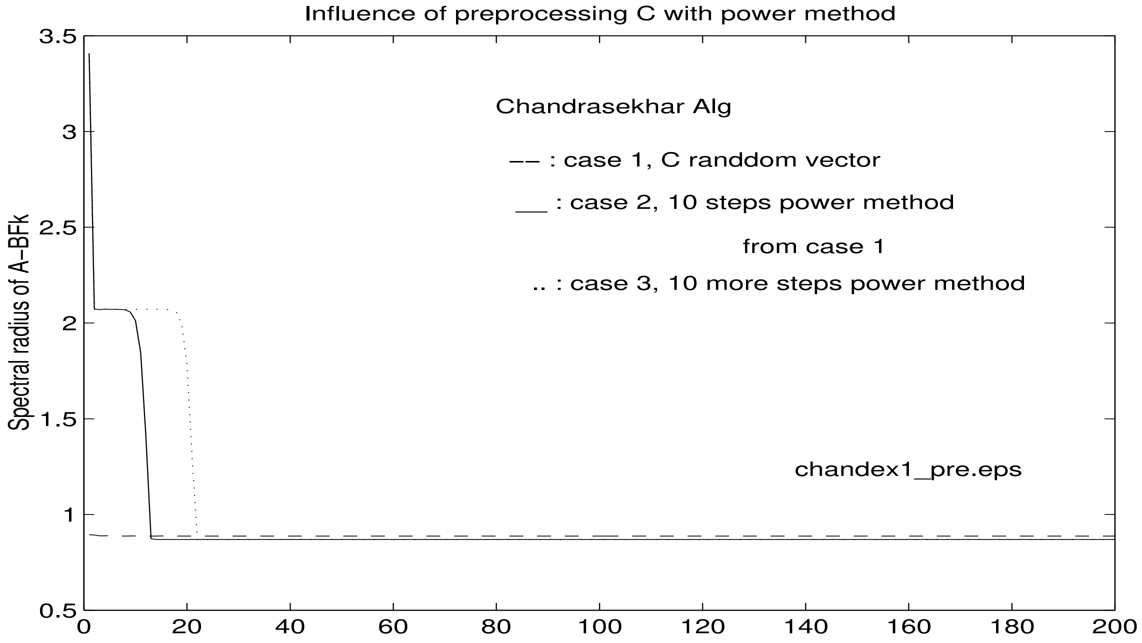

Figure 6.7. CSQR: preprocessing C, Example 13

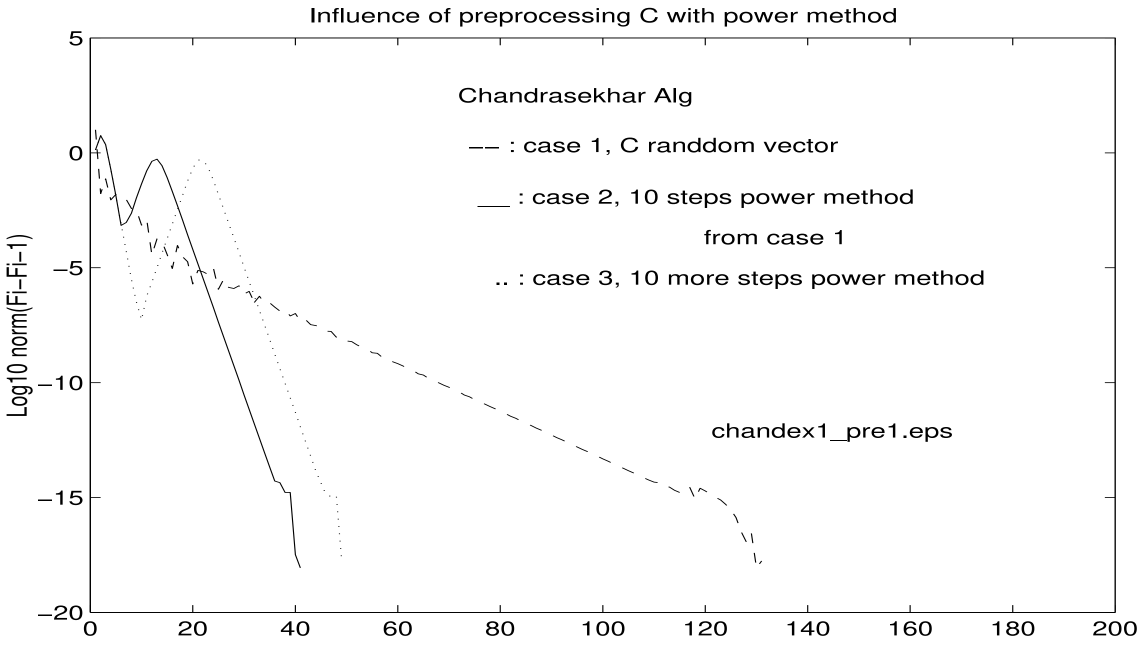

Figure 6.8. CSQR: preprocessing C, Example 13

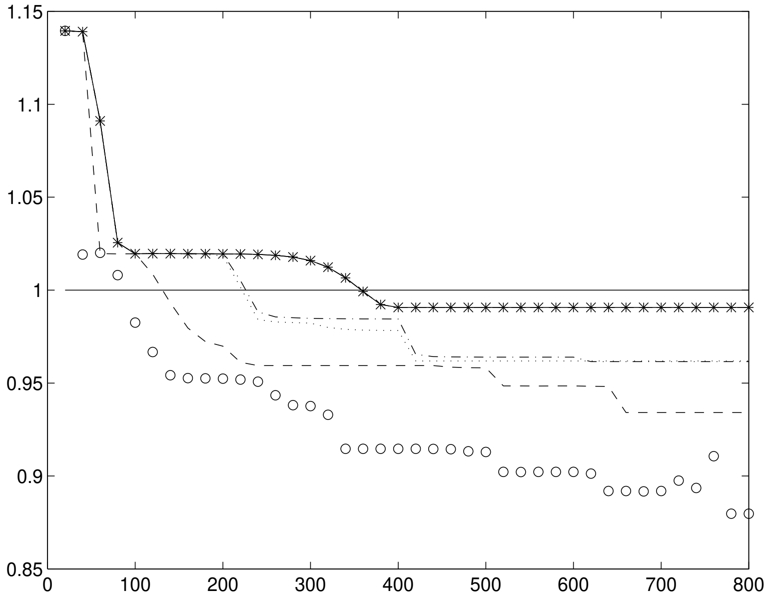

Figure 6.9. Spectral radius of $A - BF_i$ with the restart CSQR, Example 12

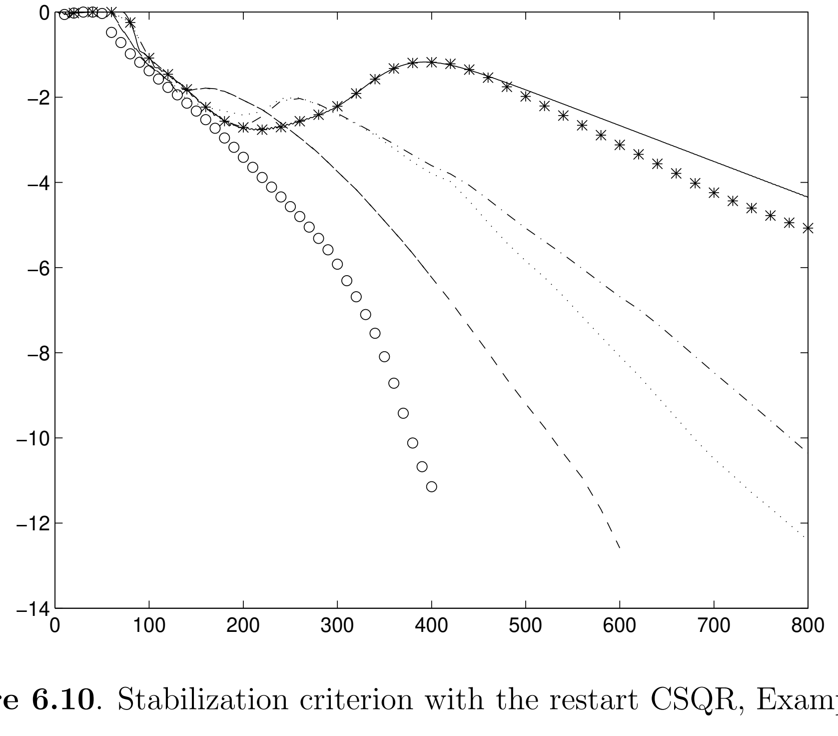

Figure 6.10. Stabilization criterion with the restart CSQR, Example 12

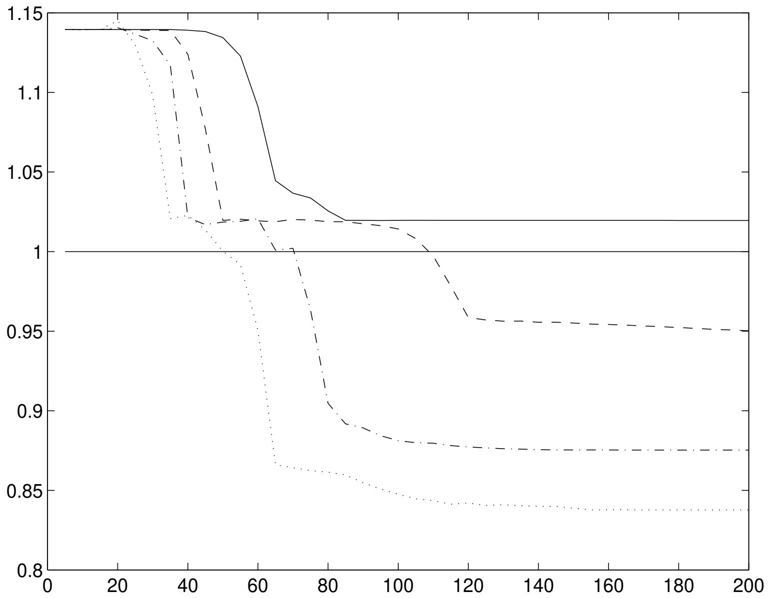

Figure 6.11. Spectral radius of $A-BF_i$ via the CSQR with scaling A, Example 12

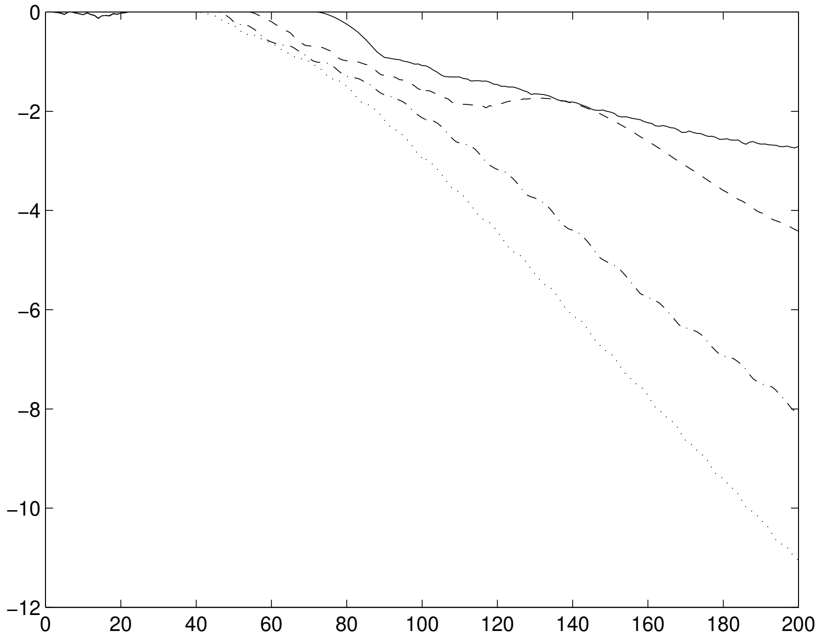

Figure 6.12. Stabilization criterion of the CSQR with scaling A, Example 12

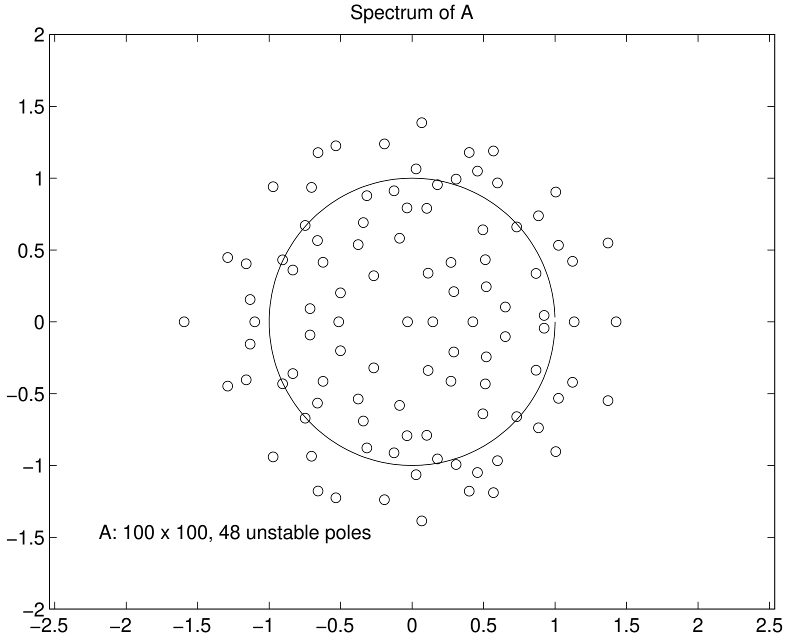

Figure 6.13. Spectrum of A, Example 18

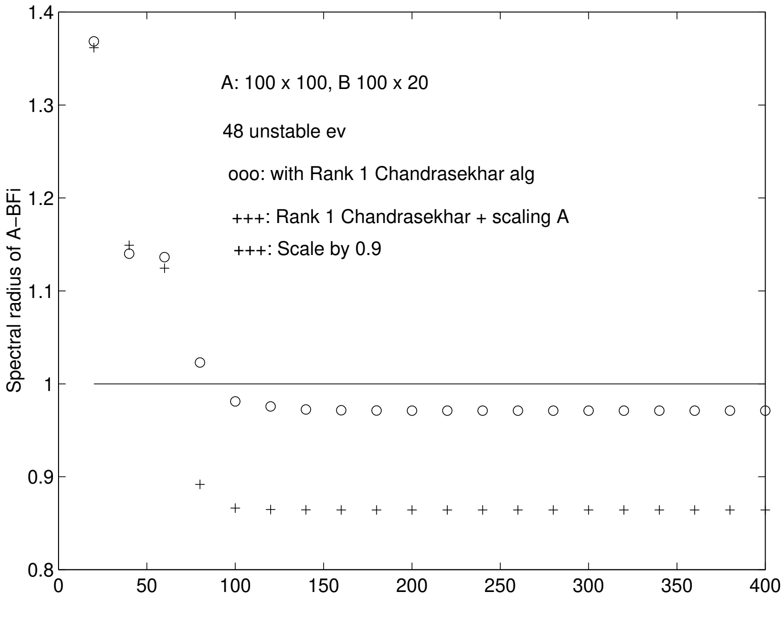

Figure 6.14. Spectral radius convergence of $A - BF_i$ with CSQR, Example 18

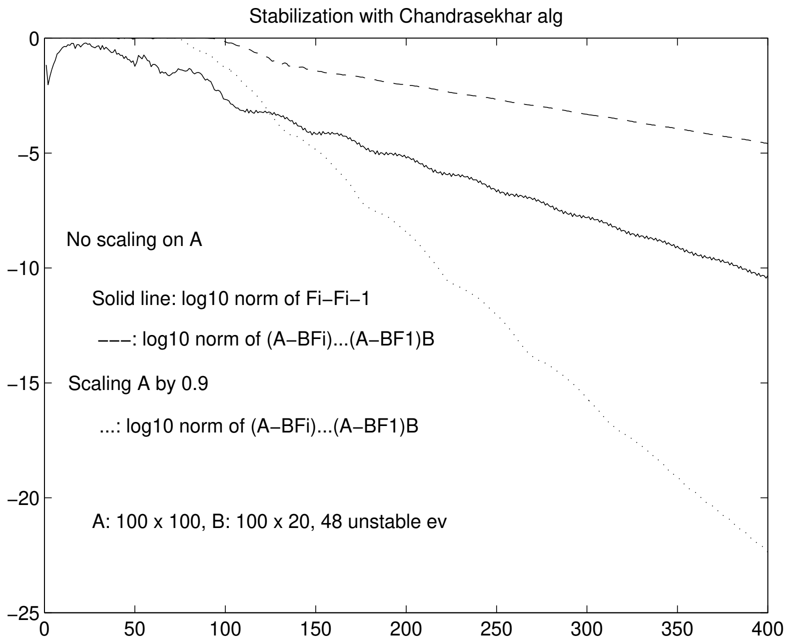

Figure 6.15. Convergence of Chandrasekhar, Example 18

CHAPTER 7

APPROXIMATE NEWTON METHOD WITH SQUARE ROOT DECOMPOSITION

The low-rank algorithms SQR and CSQR stabilize large scale (sparse) discrete time system are two different implementations of the DRDE with different parameters and initial conditions. The DRDE is a *fixed point* iteration method to solve the DRE. Chapters 5 and 6 have shown that SQR and CSQR, like all fixed point iterations, converge linearly with the rate $\rho^2(A - BF_\infty)$. If the spectral radius of $A - BF_\infty$ is very close to 1, which is the usual case if A has eigenvalues close to unit circle, a very large number of iterations may be needed for convergence and stabilization. The projection method SSI also converges linearly with the rate, for rank m SSI, $|\lambda_{m+1}/\lambda_m|$. If we happen to choose a m such that $|\lambda_{m+1}/\lambda_m|$ close to 1, SSI will have trouble converging.

Newton's method, in theory, has quadratic convergence when used to solve non-linear equations but, it is not easy to construct Newton's iteration for nonlinear matrix equations like the DRE. In this chapter, we first summarize some basic results for Newton's method and construct Newton's iteration for DRE. A new square root algorithm, different from SQR and CSQR, is introduced as an implementation of an approximate Newton's iteration, of which the DRDE is a special case. Finally, the efficacy of the truncation technique to approximate the solution of DRE is discussed.

7.1 Basic Theory

To construct Newton's method for the DRE, we must first construct an appropriate function $\mathcal{R}$ of an $n \times n$ matrix X and the derivative of $\mathcal{R}$. For a function like $\mathcal{R}$ we must use the Frechet derivative which is defined for nonlinear mapping. See Chapter 3 and [29] for the definition of the Frechet derivative. An excellent discussions on fixed point and Newton method to solve nonlinear operator equations can also be found in [29].

Let $\mathcal{R}(X)$ be the residual of the DRE with X as an approximate solution:

$$-X + A'XA - A'XB(R + B'XB)^{-1}B'XA + Q$$

The Frechet derivative $\mathcal{R}'_X$ is given by

$$\mathcal{R}'_X(S) = -S + \tilde{A}'S\tilde{A} \tag{7.1}$$

where $\tilde{A} = A - B(R + B'XB)^{-1}B'XA$. The Newton method for the DRE based on $\mathcal{R}_X$ is

$$X_i = X_{i-1} - (\mathcal{R}'_{X_{i-1}})^{-1}\mathcal{R}(X_{i-1}), i = 1, 2, \cdots, \tag{7.2}$$

given that the mapping $\mathcal{R}'_{X_i}$ is invertible.

Applying the definitions of $\mathcal{R}$ and $\mathcal{R}'$ to Equation (7.2), yields

$$\begin{aligned} \mathcal{R}'_{X_{i-1}}(X_i - X_{i-1}) &= -\mathcal{R}(X_{i-1}) & (7.3)\\ -(X_i - X_{i-1}) + \tilde{A}'(X_i - X_{i-1})\tilde{A} &= X_{i-1} - A'X_{i-1}A - Q & (7.4)\\ &\quad + A'X_{i-1}B(R + B'X_{i-1}B)^{-1}B'X_{i-1}A & (7.5) \end{aligned}$$

where $\tilde{A} = A - B(R + B'X_{i-1}B)^{-1}B'X_{i-1}A$.

A matrix form of Newton's iteration for the DRE is ([39])

$$X_i - A'_{i-1}X_iA_{i-1} = Q + F'_{i-1}RF_{i-1}, i = 1, \cdots, \tag{7.6}$$

where

$$F_i = (R + B'X_iB)^{-1}B'X_iA,\ A_i = A - BF_i \tag{7.7}$$

The discussion of the solutions and convergence of Newton's method for the DRE can be found in [39] and its references. The following result which combines Theorem 1.1, Theorem 1.3 and Theorem 1.4 of [39] summarizes sufficient conditions for the existence of a stabilizing solution, convergence and the convergence rate of Newton's method applied to the DRE.

Theorem 7.1 *Let (A, B) be stabilizable. Assume that there is a symmetric solution X of the inequality $\mathcal{R}(X) \geq 0$ for which $R + B'XB > 0$. For any F_0 such that $A_0 = A - BF_0$ is stable and X_0 which satisfies*

$$X_0 - A'_0X_0A_0 = Q + F'_0RF_0$$

the iteration (7.6) determines a decreasing sequence of symmetric matrices X_i such that $A-BF_i$ is stable and X_i converges to X_+, a maximal symmetric solution of DRE. (Maximal means that for any symmetric solution X of DRE, X_+-X is nonnegative.) Moreover, $A - B(R+B'X_+B)^{-1}B'X_+A$ is stable and there is a constant $c > 0$ such that $\|X_{i+1} - X_+\| \leq c\|X_i - X_+\|^2$ for $i = 0, 1, \cdots$ and any matrix norm $\|\cdot\|$.

Although Newton's iteration converges quadratically, the implementation is not easy. First, Newton's method is usually sensitive to the initial guess. Theorem (7.1) indicates that a stabilizing initial guess is required, or equivalently, a stabilizing feedback is required. Given that a stabilizing feedback is what we are trying to determine in the first place this is unacceptable. Even with a good initial guess, the implementation of equation (7.6) is difficult. It requires the solution of a discrete

Lyapunov equation during each iteration. Any unstable A_i will result in that the corresponding Lyapunov equation does not have bounded solution. Furthermore, the computation to solve the Lyapunov equation is not cheap (almost the same as solving DRDE). In practice, we need to remove the strict constraints on the initial guess and avoid solving Lyapunov equations. In the rest of the chapter, we investigate the approximate implementation of Newton's method given below. The method has a convergence rate between the full implementation of Newton method and the DRDE approaches of the earlier chapters.

Algorithm 11 Approximate Newton Iteration for DRE: *Specify a small integer K (usually around 5). Starting from $X_0^K = X_0$, a positive semidefinite random matrix, for $i = 1, 2, \cdots,$ (outer loop) and $j = 1, 2, \cdots, K$ (inner loop),*

$$X_i^j = A'_{i-1} X_i^{j-1} A_{i-1} + Q + F'_{i-1} R F_{i-1} \tag{7.8}$$

where $X_i^0 = X_{i-1}^K,\ F_{i-1} = (R + B' X_{i-1}^K B)^{-1} B' X_{i-1}^K A,\ A_{i-1} = A - BF_{i-1}$.

Algorithm 11 does not require a stabilizing initial guess since we only use finite steps of iteration to approximate each Lyapunov equation. We need only randomly choose a positive semidefinite X_0. Note we use X_{i-1}^K instead of 0 as the initial guess of the next round's approximation of Lyapunov equation. When applied to stabilize the system (A, B), the full rank implementation of Algorithm 11 stabilizes and converges much faster than the DRDE-based CSQR algorithm, because we can freely choose the initial guess while Chandrasekhar algorithm restricts initial guess to zero. If K is large enough, the convergence is quadratic. (measured by number of outer loops used).

7.2 Efficiency Considerations

If we take $K = 1$, Equation (7.8) is the DRDE with parameters A, B, R, Q. The DRDE here has form

$$P_i = A_i' P_i A_i + F_i' R F_i + Q \tag{7.9}$$

which is equivalent to the form

$$P_i = A'(P_i - P_i B(R + B' P_i B)^{-1} B' P_i) A + Q \tag{7.10}$$

we used in previous chapters. While Newton's iteration usually converges quadratically, the convergence is relative to number of Lyapunov equations or number of outer loops in Algorithm 11. If measured by total number of iterations (outer $\times$ inner), the convergence of Algorithm 11 may or may not be significantly different from the DRDE. In Example 19, we find Newton's iteration with Algorithm 11 does accelerate stabilization process for the difficult example we have used in previous chapters.

Direct implementation of Algorithm 11 can be as unstable as the direct implementation of the DRDE. Note all the terms in both sides of Equation (7.8) should be positive semidefinite. So, we can borrow some ideas from the DRDE algorithms to keep X_i^j positive semidefinite. To do so, we decompose the matrices as follows

$$\begin{aligned} X_i^j &= (X_i^j)^{1/2} (X_i^j)^{T/2} \\ Q &= Q^{1/2} Q^{T/2} \\ R &= R^{1/2} R^{T/2}. \end{aligned}$$

After substitution Equation (7.8) becomes

$$\left[A_{i-1}' (X_i^{j-1})^{1/2} \;\; Q^{1/2} \;\; F_{i-1}' R^{1/2}\right] U = \left[(X_i^j)^{1/2} \;\; 0\right] \tag{7.11}$$

where U is orthogonal.

With a nonzero Q, the rank of the solution P (X_+ here) of the DRE depends on the number of right eigenvectors of A that are not orthogonal to Q and the choice of X_0. For small dense problems where n is small, we can implement a full rank version of Equation (7.11) and leave X_i^j with dimension $n \times n$. For large scale systems, especially large sparse systems, a full dimension implementation of Equation (7.11) with an $n \times n$ X_i^j is computationally impractical. Unless $Q = 0$, where the rank of the solution P or X_+ is at most the number of unstable eigenvector, we have to use some form of truncation to approximate $(X_i^j)^{1/2}$ such that number of columns of approximate $(X_i^j)^{1/2}$ is acceptably small. Usually, we only keep the dominant part of X_i^j as measured by its singular values.

First we specify a maximum number of columns in $(X_i^j)^{1/2}$, l. If $Q^{1/2}$ has rank m, the dimension of $(\tilde{X}_i^j)^{1/2} = \left[A'_{i-1}(X_i^{j-1})^{1/2} \;\; Q^{1/2} \;\; F'_{i-1}R^{1/2}\right]$ is $n\times(l+m+p)$. If l, m, p are very small, the QR decomposition on a matrix of dimension $(l + m + p) \times n$ can not produce the right side of Equation (7.11). Instead, we need to use the economic form of the SVD on $(\tilde{X}_i^j)^{1/2}$ such that $(\tilde{X}_i^j)^{1/2} = U_i^j S_i^j V_i^j$, and take the first l columns of U_i^j and first $l \times l$ submatrix of S_i^j to form X_i^j, i.e., $X_i^j = U_i^j(:, 1 : l)S_i^j(1 : l, 1 : l)$. In [37], Verlaan uses the same idea to decompose the DRDE into a square root form. Instead of using the form of the DRDE in Equation 7.9 adjusted to include feedback

$$P_i = (A - BF_{i-1})'P_{i-1}(A - BF_{i-1}) + Q + F'_{i-1}RF_{i-1}$$

which is easy to decompose, [37] uses the usual form of DRDE in Equation 7.10 with a method of approximating the square root $(\tilde{P}_i)^{1/2}$ of $P_{i-1} - P_{i-1}B(R + B'P_{i-1}B)^{-1}B'P_{i-1}$ based on ideas from Kalman filtering. The truncation is implemented with a full SVD on $\left[A'(\tilde{P}_i)^{1/2} \;\; Q^{1/2}\right]$. The square root algorithm and truncation algorithm in this thesis is easier to understand and to implement.

The key to use square root and truncation algorithm for stabilization is the choice of l. l can not be too small, at least it should be larger than the number of unstable eigenvalues. Even so, since a eigenvalue is unstable does not necessarily mean that it will have larger components in X_i^j than stable ones, we have to be very careful to choose l. Another important issue is the convergence of the truncation algorithm, which needs further research even for approximation of DRDE. For stabilization purpose, if l is not smaller than the number of unstable eigenvalues and the spectral radius of corresponding closed loop is not too close to 1, we can use $(A - BF_i) \cdots (A - BF_1)B$ to measure the stabilization. One exception is when $Q = 0$ where the rank of a stabilizing solution of DRE is determined by the number of unstable eigenvalues, so we can use the stabilization/convergence measures we used in Chapter 5. The truncated square root Newton's iteration (TSQR) given below.

Algorithm 12 Truncated Square Root Newton's Iteration: *Let A be $n \times n$ and B be $n \times p$.*

1. *Specify integers $l > 0$ and m (m is zero if we choose $Q = 0$). Take $(X_0)^{1/2}$ and $Q^{1/2}$ as random matrices with dimension $n \times l$ and $n \times m$. Take $R = I$ and $F_0 = (R + B'X_0B)^{-1}B'X_0A$.*

2. *Specify a small integer $K > 0$ and an integer N. Let $(X_0^K)^{1/2} = (X_0)^{1/2}$. For i from 1 to N,*

   (a) *For j from 1 to K, do*

      i. $(\tilde{X}_i^j)^{1/2} = \left[(A - BF_{i-1})'X_i^{j-1} \quad Q^{1/2} \quad F_{i-1}'R^{1/2}\right]$.

      ii. *Apply economic SVD on $(\tilde{X}_i^j)^{1/2}$ to get $(\tilde{X}_i^j)^{1/2} = U_i^j S_i^j V_i^j$.*

      iii. *Update $(X_i^j)^{1/2} = U_i^j(:, 1:l)S_i^j(1:l, 1:l)$.*

   (b) *Compute $F_i = (R + B'X_i^jB)^{-1}B'X_i^jA$ and let $(X_{i+1}^0)^{1/2} = (X_i^K)^{1/2}$.*

   (c) *Compute some stabilization criterion or convergence criterion. Stop if any criterion is reached.*

Different stopping criteria can be used in Algorithm 12. If TSQR is expected to converge, we can use the feedback difference $F_i - F_{i-1}$ to measure convergence. If $Q = 0$, we can even borrow the stabilization criteria from Chapter 5. Otherwise, we can use $(A - BF_i) \cdots (A - BF_1)B$ as the stabilization criterion (see Chapters 5 and 6).

First, consider an example where $Q = 0$ (so truncation will not cause trouble with convergence if the rank of $(X_i^j)^{1/2}$ is at least the number of unstable eigenvalues). A nonzero Q is discussed later. Note the cost of one step of the inner iteration is closed to the cost of one step of the SQR iteration with rank l.

Example 19 *The matrix A and B is taken from Example 8 and Example 11. $(X_0)^{1/2}$ is generated as a 100×4 random matrix by RAND of MATLAB. Figure 7.1 plots the spectral radius of the corresponding closed-loop matrix vs total number of steps of TSQR (inner loop $\times$ outer loop). Figure 7.2 shows the feedback convergence (here we use $F_i - F_\infty$ instead feedback increment). Note $K = 1$ is the DRDE. Measured by the total number of iterations, in the case where $K = 5$ or $K = 10$, both the spectral radius of corresponding closed-loop and feedback converges much faster than the DRDE. For the corresponding closed-loop spectral radius to reach convergence, the DRDE uses at least 300 steps while TSQR with $K = 5$ or $K = 10$ uses less than 100. Furthermore, from Figure 7.2, we can see that TSQR with $K = 5$ or $K = 10$ starts to converge at the beginning of the iteration while the DRDE spends some iterations adjusting. The reason is, some unstable eigenvalues of A are very close to 1, and TSQR with larger K gives these eigenvalues more opportunities to increase their components in $(X_i^j)^{1/2}$ and F_i. When applied to the system in Example 10, since all the eigenvalues are well-separated from unit circle, TSQR with larger K uses more steps that the DRDE to converge to the same tolerance.*

With nonzero Q, TSQR is very sensitive to the choice of l. In Example 19, if we only add a nonzero $Q^{1/2}$ and keep all other parameters fixed, the case with $K = 1$ (DRDE) needs a very large l to reach a stabilizing feedback. Even when we choose $K = 5$ or $K = 10$, we still need to choose K larger than the number of unstable eigenvalues and need large number of iterations to reach a stabilizing feedback. The reason is, if Q is not orthogonal to a right stable eigenvector, we can not assume the component of this eigenvector in P or X_+ is less significant than unstable eigenvectors. TSQR sometimes performs poorly for stabilization since the components of some unstable eigenvectors in (X_i^j) and F_i are truncated. However, iff $Q^{1/2}$ only contains unstable components of A, that is, $v'Q^{1/2} = 0$ for any right stable eigenvector of A, the solution of P or X_+ of DRE should not contain any components of stable eigenvectors of A. So, truncation should be able to keep the unstable components instead of the stable components.

As a modification and refinement of TSQR could be modified to improve this situation by inserting SSI on $Q^{1/2}$ along with each $(X_i^j)^{1/2}$ (inner) iteration such that $(X_i^j)^{1/2}$ will only focus on the unstable components of A. We can also only insert the SSI on Q into outer loop along with the update of F_i. For $K = 1$, this method is equivalent to solving a DRDE $P_i = A'(P_{i-1} - P_{i-1}B(R + B'P_{i-1}B)^{-1}B'P_{i-1})A + Q_{i-1}$ where $Q_{i-1}^{1/2}$ is updated via SSI on $Q_0^{1/2}$ with A'. Since SSI will converge to eigenvectors with largest eigenvalues, for best performance, we should choose the rank of $Q_0^{1/2}$ the same as the number of unstable eigenvalues of A so that all unstable eigenvectors of A will have significant components in Q_i. On the other hand, it is better to choose l larger than the number of unstable eigenvalues for higher flexibility in truncation. Example 20 shows that the performance of both stabilization and feedback convergence are enhanced significantly with this type of modification on Q if the rank of $(X_i^j)^{1/2}$ is not less than the number of unstable eigenvalues of A.

To use TSQR to approximate the DRE or DRDE where all the parameters (A, B, R, Q) are predetermined (for example in optimal filtering), we have to be very careful when choose l. In general, we need to find a small l such that $|\lambda_{l+1}/\lambda_l|$ not close to 1.

The convergence of TSQR depends on the spectral radius of corresponding closed-loop matrix and can be limited by the separation of eigenvalues of A from unit circle. The restart technique with deflation we used in Chapter 6 should be effective here. TSQR cannot use scaling A to accelerate stabilization process. This is due to the fact that for any $r < 1$, the scaled system $(A/r, B/r)$ usually has more unstable eigenvalues the original system (A, B). Since a larger eigenvalue does not necessarily have more significant components in $(X_i^j)^{1/2}$, it is possible that the component of this eigenvalue is truncated out unless l is large enough to be the number of unstable eigenvalues of A/r. Since the (inner) iteration uses $A - BF_i$ instead A as in SQR and CSQR, for problems which have poor eigen-conditioning for the corresponding closed-loop matrix, the performance of TSQR could be poor.

Example 20 *The system* (A, B) *and* $(X_0)^{1/2}$ *is the same as in Example 19.* $Q_0^{1/2}$ *is chosen as* 100×4 *and* l *is chosen as 6.* $Q^{1/2}$ *is updated by SSI as discussed. Compared to Example 19, Figure 7.3 shows that with SSI on* Q*, the total number of iterations to reach a stabilizing feedback is cut almost by half for* $K > 1$*, while feedback convergence rate (Figure 7.4) does not change significantly.*

7.3 Summary

In this chapter, we have developed a low-rank stabilization algorithm, TSQR, via Newton's method to solve the DRE. If used correctly, TSQR can be used to approximate the DRDE for large scale optimal filtering problems where all the parameters are predetermined. For stabilization purposes, TSQR can find a stabilizing feedback efficiently by manipulating parameters Q, R and the initial guess with more freedom than the SQR and CSQR algorithms discussed earlier. TSQR also unifies our algorithms and others in the literature such as [37].

Many open questions remain. The first is the convergence properties of the approximate Newton's iteration (Algorithm 11) for an arbitrary initial condition. The second is the convergence properties of TSQR for both stabilization and solution of the DRDE. The current form of TSQR does not attempt to exploit techniques that were used to transform SQR into CSQR, i.e., computing the difference rather than the square root. Considering extensions of TSQR along those lines is promising.

The approximation of the solution to the Lyapunov equation required on each step must also be considered further in terms of efficiency and its overall role in the convergence of the iteration. Finally, the approximate Newton's iteration and TSQR may be very useful when applied to solving the continuous-time Riccati Equation for various purposes. Their adaptation to this case should be considered. Further research on these issues will benefit not only stabilization in optimal control, but also optimal filtering and data assimilation problems where solving a large scale DRDE might be impractical.

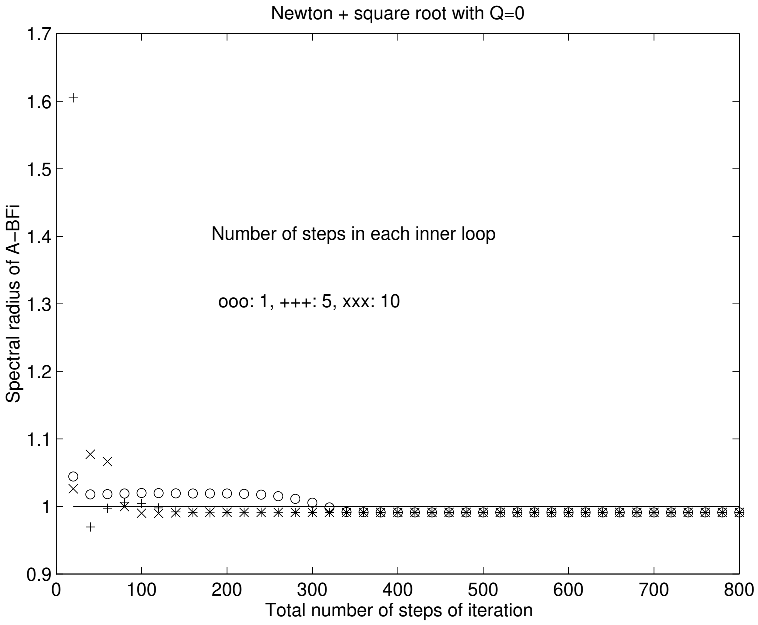

Figure 7.1. TSQR($Q = 0$) vs. SQR: Spectral radii of $A - BF_i$, Example 19

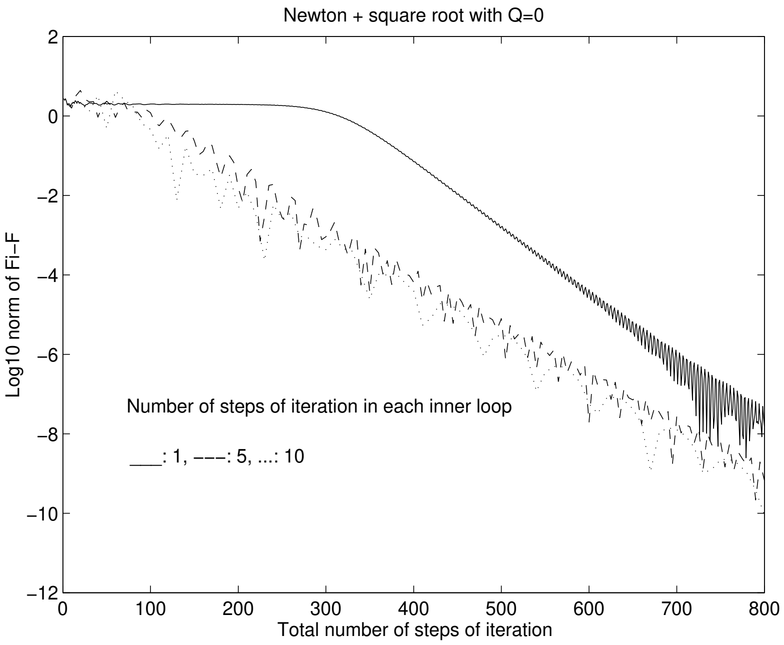

Figure 7.2. TSQR($Q = 0$) vs. SQR: Feedback convergence, Example 19

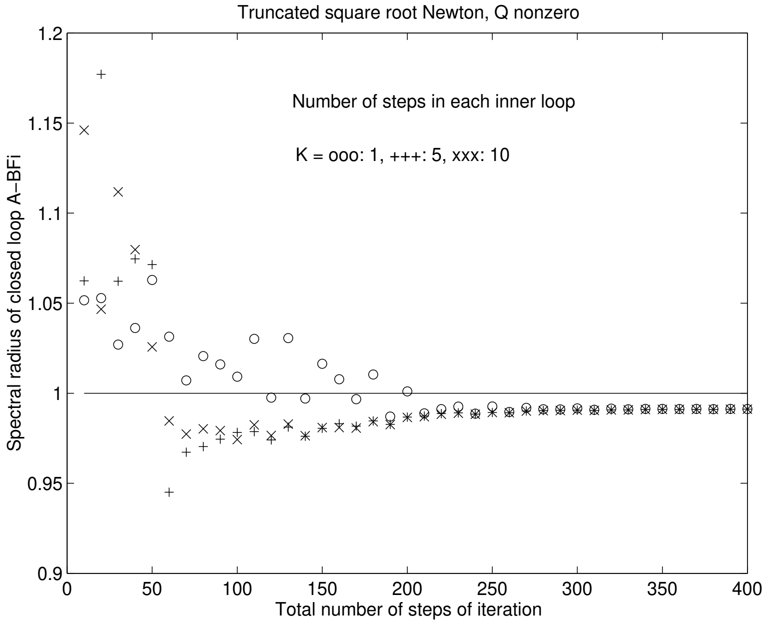

Figure 7.3. TSQR: Spectral radii of $A - BF_i$, Example 20

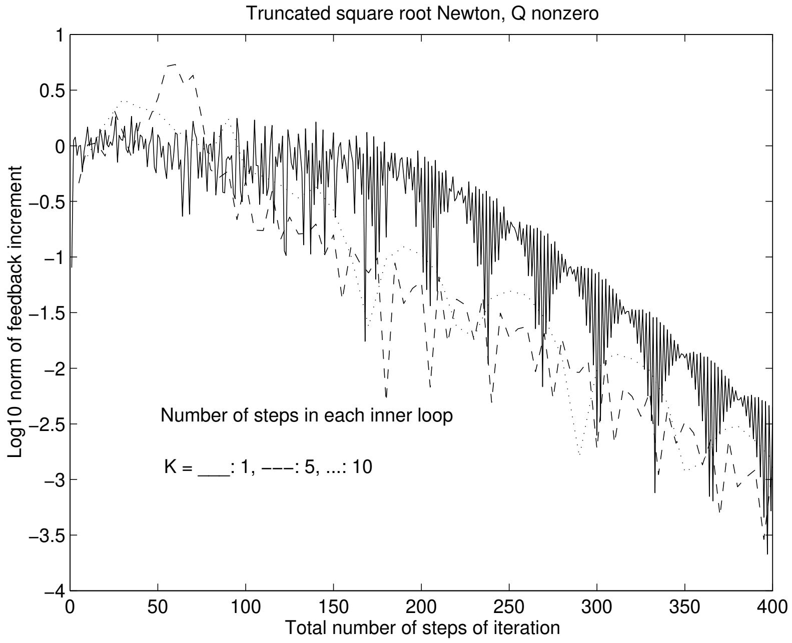

Figure 7.4. TSQR: Feedback convergence, Example 20

CHAPTER 8

STABILIZATION OF CONTINUOUS-TIME SYSTEMS

In this chapter we investigate stabilizing large-scale continuous-time dynamical systems

$$\dot{x} = Ax + Bu \tag{8.1}$$

with feedback F where A and B are $n \times n$, and $n \times p$ real matrices respectively and F is a $p \times n$ real matrix to be determined. Usually n is very large while $p \ll n$. The feedback F stabilizes the system if all eigenvalues of $A - BF$ have negative real parts. As with discrete-time systems, eigenvalue assignment (pole placement) algorithms can be used (see Chapter 1). The difficulties encountered in stabilizing large scale continuous-time systems are summarized in [10] by Saad and [8] by He and Mehrmann.

As with discrete-time systems, Saad has explored the use of projection methods to stabilize continuous-time [10]. The difference lies in the characterization of the unstable subspace and the iterative method that produces its approximate basis. A projection method is also implemented by He and Mehrmann in [8]. The use the sign function to compute the unstable invariant subspace basis. The results from [10] and [8] are summarized in this chapter.

We use results from the stabilization of large scale discrete-time system to develop continuous-time stabilization algorithms via the projection methods we have discussed in Chapter 4. Based on the relationships between the general DRE and the general CRE Equation developed by Hung and Chu in [40], we also extend our discussions from Chapters 5, 6 and 7 to develop algorithms to stabilize large scale continuous-time system.

8.1 Direct Discretization Methods

Almost all of the algorithms to stabilize large scale continuous-time system are linked to a discrete-time system method, directly or indirectly. A natural idea is to discretize the continuous-time system and stabilize the resulting discrete-time system. This approach is discussed in this section.

Let x_n denote $x(n\Delta t)$ and approximate $\dot{x}(n\Delta t)$ by $(x_{n+1}-x_n)/\Delta t$. We then have an discrete-time system which is an approximation of Equation (8.1)

$$x_{n+1} = (I + \Delta tA)x_n + \Delta tBu_n. \tag{8.2}$$

It is easy to verify that if we can find a stabilizing feedback F of discrete-time system $(I + \Delta tA,\ \Delta tBF)$ such that all eigenvalues of $(I + \Delta tA - \Delta tBF)$ are inside unit circle, F will be a stabilizing feedback of continuous-time system (A, B) such that all eigenvalues of $(A - BF)$ are located in the left half complex plane.

Let $\lambda = r(\cos(\theta) + i\sin(\theta))$ be any eigenvalue of A. If $\cos(\theta) \geq 0$, i.e., λ is a continuous-time unstable eigenvalue of A, and it is obvious $1+\Delta t\lambda$ is a discrete-time unstable eigenvalue of $I + \Delta tA$. However, in order for a continuous-time stable eigenvalue, λ, to map to a discrete-time stable eigenvalue of $I + \Delta tA$ $1 + \Delta t\lambda$ a constraint on Δt must be satisfied. Specifically, we must have $\Delta t < (-2\cos(\theta))/r$. It is not easy to determine such a Δt since it is not easy to estimate the spectral radius $\rho(A)$ of A and the smallest $|\cos(\theta)|$ over all eigenvalues of A. In practice, we choose a constant $0 < \alpha \leq 0.1$ and hope that $\Delta t \leq \alpha/\rho(A)$ transforms a few as possible continuous-time stable eigenvalues of A to discrete-time unstable eigenvalues of $(I+\Delta tA)$. While this direct discretization method maintains the sparse characteristics for large sparse systems (A, B) and avoids the expensive solutions of large scale systems involved in other algorithms, estimating Δt causes some difficulties.

The optimal Δt should not be too small, or $I + \Delta t A$ will have too many eigenvalues (both stable and unstable) very close to unit circle, which will limit our choice of discrete-time stabilization algorithms and slow both convergence and the stabilization process. Δt should be chosen as large as possible under the condition that $I + \Delta t A$ only has very few unstable eigenvalues. After discretization, SSI or SQR might not be the ideal algorithm to compute feedback from the discrete-time stabilization problem, since the rank of SSI and SQR depends on the number of unstable eigenvalues of $I + \Delta t A$, which could be much larger than the number of continuous-time unstable eigenvalues of A and is not easy to estimate. As we have discussed in previous chapters on discrete-time system, CSQR is a true low-rank algorithm and the rank is independent of the number of unstable eigenvalues. Furthermore, CSQR can more effectively deal with a system with eigenvalues very close to unit circle. Without any knowledge of the eigenvalue distribution of A, CSQR with scaling techniques introduced in Chapter 6 is strongly recommended over other algorithms to compute the stabilizing feedback F of the transformed discrete-time system $(I + \Delta t A, \Delta t B)$. The algorithm to stabilize continuous-time system via direct discretization is summarized as follows.

Algorithm 13 Direct discretization algorithm

1. **Discretize:** *Estimate the spectral radius ρ of A. Choose a constant $0 < c \leq 0.1$. Take Δt less than c/ρ.*

2. - **Stabilization via projection:** *Apply stabilization algorithms introduced in Chapter 4 to compute stabilizing feedback F of discrete-time system $(I + \Delta t A, \Delta t B)$. This feedback F will stabilize continuous-time system (A, B), too. Or*

- **Stabilization with DRE based algorithm:** *Apply any stabilization algorithm (SQR, CSQR, Newton's iteration and their modifications) introduced in Chapters 5, 6 and 7 to find a stabilizing feedback F of discrete-time system $(I + \Delta tA, \Delta tB)$. This feedback F will stabilize continuous-time system (A, B), also.*

Example 21 *(Figure 8.1 and Figure 8.2) System (A, B) is generated in the following way: We create a 100×100 matrix with MATLAB RANDN function and shift by 7.2 to get A which has 6 unstable eigenvalues and all stable eigenvalues are located in $[-20, 0] \times [-10, 10]$. B is generated by RAND as a random vector. Δt is taken as 0.03, 0.05 and 0.1 respectively. The Rank 1 Scaled CSQR algorithm is used to stabilize the discrete-time system $(\tilde{A}, \tilde{B}) = (I + \Delta tA, \Delta tB)$ and compute the feedback F. We use $(\tilde{A} - \tilde{B}F_i) \cdots (\tilde{A} - \tilde{B}F_1)\tilde{B}$ as the stopping criterion as in Chapter 6. From Figure 8.1 and Figure 8.2, we see that with larger Δt, the direct discretization plus Rank 1 CSQR iteration with the larger Δt needs fewer iterations to get a stabilizing feedback with smaller norm. With Δt as 0.03, 0.05 and 0.1, after 400 steps of iteration, $\|F\|_2$ is 3.6×10^8, 2.6×10^6 and 3.6×10^4 respectively. Compare this to the results in Example 23 where a linear rational transformation or its approximation is used to transform (A, B) to a discrete-time system, and $\|F\|_2$ is 2.6×10^3 with SQR and 848.7693 with Chandrasekhar algorithm.) The stabilization is detected earlier with the larger Δt as well. The main research task for this class of methods is therefore determining heuristically the appropriate value of Δ and making sure it is large enough to avoid these difficulties while not being so large as to destroy stabilization.*

8.2 Projection Method Theory

As with the stabilization of large scale discrete-time system, there are two classes of algorithms to stabilize large scale continuous-time system: stabilization via projection and stabilization via the control matrix equations. With projection methods, we first compute a left invariant subspace V of A which is a superspace of the continuous-time left unstable invariant subspace of A (and hopefully of low dimension), then project the system (A, B) onto V, stabilize the reduced continuous-time system $(V'AV, V'B)$ with feedback F_u and lift F_u back to the original space to get a stabilizing feedback $F = F_uV'$ (see [10] and Chapter 4). A stabilizing solution of a continuous-time Riccati Equation (CRE) will also produce a stabilizing feedback. With some preprocessing on A, e.g., shifting A to be stable, a linear system can even be stabilized by a continuous-time Lyapunov equation [9]. In this section, we focus on projection methods.

The key for stabilization via a projection method is the computation of the left unstable invariant subspace of A. While all the principles of projection methods in Chapter 4 can be used both in discrete-time and continuous-time system, we have not discussed any useful method to compute continuous-time unstable invariant subspace, especially for large scale system.

The most recent method to compute continuous-unstable invariant subspace of A is using the sign function of A (see [41] and references therein). The work of He and Mehrmann [8] combines projection via the sign function and a reduced-order CRE that is transformed into an equivalent reduced-order continuous-time Lyapunov equation to stabilize continuous-time system.

8.3 The Sign Function Method

The sign function of matrix A can be defined by

$$sign(A) = W' \begin{bmatrix} -I_{n-p} & Y \\ 0 & I_p \end{bmatrix} W \tag{8.3}$$

where W is a matrix of Schur vectors of A such that

$$W'AW = \begin{bmatrix} A_{11} & A_{12} \\ 0 & A_{22} \end{bmatrix} \tag{8.4}$$

where A_{11} contains all stable eigenvalues of A and A_2 contains all unstable eigenvalues of A, and Y is a solution of $YA_{22} - A_{11}Y = 2A_{12}$.

The sign function of A can be computed with Newton's method or its refinements as follows.

- Start from $A_0 = A$
- $A_i = \frac{1}{2c}(A_{i-1} + c^2 A_{i-1}^{-1}), \quad c = |detA_{i-1}|^{1/n}$ until some stopping criterion tolerance is reached.

The convergence of the iteration depends on the distance of the eigenvalues of A to imaginary axis. If convergent, the limit A_∞ will only have -1 and 1 as eigenvalues. With a shift, for example, $A_\infty + I$, the two sets (1 and -1 of A_∞) of eigenvalues are well separated. With Stewart's SRR refinement for SSI, it should not take too many iterations on $A_\infty + I$ to get the invariant subspace of A_∞ with eigenvalue 1, or the unstable invariant subspace of A.

The update of A_i is the main impediment to using this algorithm for large scale sparse system since it destroys sparsity. For large sparse problems, an alternate version of the iteration that avoids the update or a version that "sparsifies" A_i, must be developed. We know of no current work along these lines. On the other hand, the sign function can be used to compute the stabilizing solution of low-order continuous-time or discrete-time Riccati equations that might be needed to stabilize the reduced continuous-time or discrete-time system after projection. The details of how to solve Riccati equations via sign function and the computations of sign function are discussed in [42].

8.4 Polynomial Methods

For discrete-time systems, usually we compute the invariant subspace from eigenvectors (or Schur vectors) or from SSI. This idea can be used for continuous-time system. The usual method to compute eigenvalues or invariant subspace (power method, orthogonal iteration, Arnoldi, etc) tends to identify eigenvalues (or the associated eigenvectors) with the largest absolute values. To compute continuous-time unstable invariant subspace, we need to get eigenvectors corresponding to right most eigenvalues in the complex plane, so some transformation is required. Under such a transformation, a continuous-time unstable eigenvalue of A should be transformed to a discrete-time unstable eigenvalue of a new matrix $\tilde{A}$ and the number of stable eigenvalues of A which are transformed into unstable eigenvalues of $\tilde{A}$ should be as small as possible. A frequently used transformation has the form of $u = (a + bz)/(c + dz)$ where $a,\ b,\ c,\ d$ are complex. With such a transformation, we transform A to $\tilde{A} = (aI + bA)(cI + dA)^{-1}$. Any prior knowledge could be helpful to determine the parameters $a,\ b,\ c,\ d$ such that $\tilde{A}$ has the simplest form in A. One example is the direct discretization algorithm mentioned earlier, where we $d = 0,\ c = 1$ and $a = 1,\ b = \Delta t$. Another choice is $a = 1,\ b = 1$ and $c = -1,\ d = 1$, that is, $\tilde{A} = (A + I)(A - I)^{-1}$ that transforms the right half complex plane to outer of unit circle. Applying SSI or eigen-solvers such as the Arnoldi method on $\tilde{A}'$ should produce the invariant subspace containing the eigenvalues of A' with positive real parts. This transformation has another advantage, it can transform some continuous-time control equations such as Lyapunov equation and Riccati Equation into equivalent discrete-time forms ([43], [36] and [40]).

For a large sparse matrix A, the disadvantage to use a $\tilde{A}$ as described above is the computation of matrix inverse of $cI + dA$. A standard approach is that we do not compute $(cI + dA)^{-1}$, instead we solve a linear system $(cI + dA)x = v$ when computation of the form $(cI + dA)^{-1}v$ is needed. Even so, such an approach does increase the operations required considerably. It is often possible to avoid the computation of the matrix inverse or the solution of a system via some simple approximations. For example, in [34] Meerbergen computes the rightmost eigenvalues of general eigenvalue problem $A = \lambda B$ with the Arnoldi method on a matrix with form $M_c(A - \lambda B)$ where M_c is an approximation of $(A - \alpha B)^{-1}$. We can also use lower order polynomials to approximate the linear rational transformation $(aI+bA)(cI+dA)^{-1}$, for example, we can use a Taylor expansion of degree k. Usually, we use $\tilde{A} = (I + \alpha A)(I - \alpha A)^{-1}$ where α is used to scale A so that the degree k of the approximated polynomial can be kept as small as possible (e.g., we can choose $\alpha = \frac{1}{2\rho(A)}$ where $\rho(A)$ is the spectral radius of A). Such an approximation of $\tilde{A}$ can be used to compute the rightmost eigenvalues/eigenvalues of A and the continuous-time unstable invariant subspace. If $\rho(A)$ is not too large, say less than 100, and we choose $k = 8$, such a polynomial approximation is very efficient both in computation and storage. As with discrete-time SSI, we can compute continuous-time invariant subspace for large sparse system as follows.

Algorithm 14 Subspace iteration with linear rational transformation

1. **Compute $\tilde{A}$:** *Estimate an $\alpha > 0$ such that $(I - \alpha A)$ is nonsingular and any unstable eigenvalue of αA has absolute value less than 1. Specify a small integer k and compute $\tilde{A}$ from degree k Taylor expansion of $(I + \alpha A)(I - \alpha A)^{-1}$. Note we should leave $\tilde{A}$ in the form of polynomial of A. (If such an α cannot be found, use $\tilde{A} = (I + \alpha A)(I - \alpha A)^{-1}$ such that $I - \alpha A$ is nonsingular.)*

2. **Run SSI on $\tilde{A}'$:** *Specify an integer m and a random vector v, run the Arnoldi algorithm on $\tilde{A}'$ and v to get m orthogonal vectors V_0. Specify an integer N as the maximum steps of iteration and specify an accuracy tolerance. For i from 1 up to N, V_i is updated with economical QR decomposition of $\tilde{A}'V_{i-1}$.*

3. **Implement SRR refinement and compute residual:** *In selected steps of the above iteration, compute Schur vectors Q_i and eigenvalues of $V_i'A'V_i$ (the diagonal elements of $Q_i'(V_i'A'V_i)Q_i$ are ordered such that rightmost eigenvalues appear on the top). Take r as the number of eigenvalues with positive real parts (or including the eigenvalues with negative real parts which are very close to*

imaginary axis) and compute $\tilde{V}_i$ from V_i and first r columns of Q_i. Compute the norm of residual $A'\tilde{V}_i - \tilde{V}_i(\tilde{V}_i' A'\tilde{V}_i)$, if it reaches the specified tolerance, stop. (Note once this step is executed, it is better to run several consecutive steps of this type because of the possible wrong choice of Schur vectors as we discussed in Chapter 4.)

The implementations of this algorithm vary and depend on the form of $\tilde{A}$ chosen. If the polynomial approximation is chosen, we leave $\tilde{A}$ in form of a polynomial of A and use k iterations to compute $\tilde{A}'V_i$ from $A'V_i,\ A'(A'V_i), \cdots$. If $\tilde{A}$ is chosen as linear rational function of A and A is dense, or A is large sparse but $\tilde{A}$ can be computed effectively, it is better to compute $\tilde{A}$ at the beginning of the iteration. Otherwise, we should not compute $\tilde{A}$. Instead, we compute $(I+\alpha A)'(I-\alpha A')^{-1}V_i$ by solving linear systems in each step. For a large scale system with very few unstable eigenvalues, the size of V_i could be very small compared to size of the system, so the computation and storage of $(I+\alpha A)'(I-\alpha A')^{-1}V_i$ could be much cheaper than directly computing $\tilde{A}$ if the corresponding subspace iteration converges fast enough.

When using SSI to compute continuous-time unstable invariant subspace, we need to perform some transformations that convert the continuous-unstable eigenvalues to discrete-time unstable while mapping the continuous-time stable eigenvalues (except possibly very few) to discrete-time stable. A linear rational transformation is one example but, the computations involve undesired large scale matrix inversion. Low-order polynomial transformations have advantages over linear rational transformation for large scale sparse system. An example of a low-order polynomial transformation is a low-order Taylor expansion of $(I+\alpha A)(I-\alpha A)^{-1}$. Note SSI is in fact extracting the orthogonal basis from $((I+\alpha A)(I-\alpha A)^{-1})^k V_{i-1}$ to update V_i. Another type of SSI with polynomials uses a polynomial sequence to replace $((I+\alpha A)(I-\alpha A)^{-1})^k$, e.g., Chebyshev iteration. Chebyshev SSI, which is discussed by Saad in [44], performs a minimax approximation of an ellipse $E(c, e, a)$, which has center c, foci $c+e,\ c-e$, and major semi-axis a, with Chebyshev polynomials. The asymptotically best minimax polynomial is the polynomial

$$p_k(\lambda) = \frac{C_k[(\lambda - c)/e]}{C_k[(\lambda_1 - c)/e]}, \tag{8.5}$$

where C_k is the Chebyshev polynomial of degree k of the first kind. For SSI to extract continuous unstable subspace, λ_1 should be chosen as the estimate of unstable eigenvalue with smallest real part, or the one closest to $E(c, e, a)$. With the recurrence for the Chebyshev polynomials, we can get the recurrence for p_k as following:

$$\begin{aligned} p_{k+1}(\lambda) &= 2\sigma_{k+1}\frac{\lambda - c}{e}p_k(\lambda) - \sigma_k\sigma_{k+1}p_{k+1}(\lambda) \\ \sigma_1 &= \frac{e}{\lambda_1 - c} \\ \sigma_{k+1} &= \frac{1}{2/\sigma_1 - \sigma_k}\ k = 1, \cdots. \end{aligned} \tag{8.6}$$

With the transformation of $p_k(\lambda)$, the eigenvalues located in $E(c, e, a)$ tend to have smaller magnitudes than eigenvalues outside $E(c, e, a)$,i.e., the ellipse $E(c, e, a)$ is the region containing unwanted eigenvalues. For stabilization purposes, $E(c, e, a)$ should contain all stable eigenvalues, or at most only very few stable eigenvalues could be outside of $E(c, e, a)$, and all unstable eigenvalues should be well outside $E(c, e, a)$. In practice, c is chosen as real, and e could be real or pure imaginary. As we see from the recurrence of p_k, no a is explicitly used. Now, we can use $p_k(A)$ the same way as A^k in SSI. Directly using $p_k(A)$ is not practical, instead, we use the recurrence relation.

Algorithm 15 Chebyshev SSI

1. *Start: Choose c, e and estimate λ_1, an orthogonal basis of initial subspace V_0 (so, size of V_0 is implicitly chosen) and compute*

$$\sigma_1 = \frac{e}{\lambda_1 - c}, \tag{8.7}$$

$$\tilde{V}_1 = \frac{\sigma_1}{e}A - cIV_0, \tag{8.8}$$

Perform a QR decomposition of $\tilde{V}_1$ to get V_1.

2. Iterate: For $k = 1, 2, \cdots$, until convergence or some specified number, do

$$\sigma_{k+1} = \frac{1}{2/\sigma_1 - \sigma_k}, \tag{8.9}$$

$$\tilde{V}_{k+1} = 2\frac{\sigma_{k+1}}{e}(A - cI)V_k - \sigma_k\sigma_{k+1}V_{k-1} \tag{8.10}$$

Perform a QR decomposition on $\tilde{V}_{k+1}$ to get V_{k+1}. Compute the norm of residual $AV_{k+1} - V_{k+1}(V'_{k+1}AV_{k+1})$, stop if reaches an acceptable level. The SRR method may be needed to accelerate or refine the process if the desired and undesired subspaces can not be separated very well.

A special and interesting case is $c = 0$ and $e = -i$ where the Chebyshev subspace iterations can be used to extract discrete-unstable eigenvalues if the size of V_0 is not less than the number of discrete-time unstable eigenvalues.

We have discussed two algorithms related to SSI to approximate continuous-time unstable invariant subspace: linear rational transformations and Chebyshev polynomials. For large scale sparse systems, linear rational transformations may not be efficient in terms of storage or total computations because the matrix inverse involved. However, linear rational transformations are very easy to implement when only scaling is needed to guarantee nonsingularity of $I - \alpha A$. Furthermore, if the eigenvalues (especially unstable ones) of A are concentrated in a relative small region, we can use some low-order polynomial to approximate $(I + \alpha A)(I - \alpha A)^{-1}$ and such an approximation will eliminate the matrix inverse.

Chebyshev polynomial iteration may have the advantage on total computations and storage, since only computations like Av are involved and no matrix inverse is needed. However, it is not easy to determine the parameters such as λ_1. More significantly , since a is not in the parameters used in Chebyshev iteration, separating wanted eigenvalues from unwanted ones is difficult since we can not know the exact shape of the ellipse, even if we have prior knowledge about eigenvalues distribution.

Although the Chebyshev iteration is typically used as an acceleration technique for eigensolvers like Arnoldi, the difficulty in determining parameters may make it most useful in conjunction with the deflation method to find the rightmost eigenvalues/eigenvectors one by one with Chebyshev polynomial iteration.

8.5 DRE-based Methods

Usually, the Lyapunov equation is used to determine stability or is used as part of a solution method for the CRE. With some preprocessing of A, the Lyapunov equation can be used to compute a stabilizing feedback (Armstrong in [9]), where the stabilizing feedback $F = B'Z^{+}$ is constructed from the positive semidefinite solution $Z \geq 0$ of the continuous-time Lyapunov equation

$$-(A + \beta I_n)Z + Z(-(A + \beta I_n))' = -2BB' \tag{8.11}$$

where $\beta > 0$ is chosen such that $-(A+\beta I_n)$ is stable and Z^{+} means the pseudoinverse of Z. Since B is of rank p, Equation (8.11) can be solved with a low-rank square root algorithm introduced by Penzl in [36]. The trouble in using Penzl's algorithm is that we need to compute a matrix inverse of form $(I-\alpha A)^{-1}$ or we have to solve a sequence of linear systems. Another problem is the Z^{+}. If we use a low-rank approximation of $Z^{1/2}$, we need further research of the effect on Z^{+}, especially whether Z^{+} can keep the appropriate parts of Z^{+} needed to construct a stabilizer.

Another method to determine a stabilizing feedback is through the CRE

$$A'X + XA - XBR^{-1}B'X + C'C = 0. \tag{8.12}$$

An result analogous to Theorem 3.4 in Chapter 3 on the existence of a stabilizing solution from the CRE is given in [23].

The CRE can be solved via a generalized eigenvalue solver, sign function method or Newton's method. A generalized eigenvalue solver method must compute n stable eigenvectors (or Schur vectors) of a matrix $2n \times 2n$, and it is only useful for small scale problems. The matrix sign function method in fact needs to construct the same $2n \times 2n$ Hamiltonian and compute its sign function. Computation of sign matrix with size of $2n \times 2n$ is also not cheap and is not directly practical for large scale problem.

Newton's method produces a sequences of continuous-time Lyapunov equations:

$$(A - BR^{-1}B'X_i)'X_{i+1} + X_{i+1}(A - BR^{-1}B'X_i) = -X_iBR^{-1}B'X_i - C'C. \quad (8.13)$$

If the initial condition X_0 is positive semidefinite and $A - BR^{-1}B'X_0$ is stable, we can get a sequence of nonincreasing X_i that converges to X. Each Lyapunov equation's solution can be approximated by the square root algorithm proposed by Penzl in [36], where any continuous-time Lyapunov equation is first transformed into its discrete-time equivalent.

The biggest problem in using Newton's method is the condition on X_0, where $A - BR^{-1}B'X_0$ is required to be stable. As with discrete-time approximate Newton's method, we need to do some modifications to develop an alternative. If we define

$$\begin{aligned} F_i &= R^{-1}B'X_i \\ \tilde{A}_i &= (I - A + BF_i)^{-1}(I - A + BF_i), \end{aligned}$$

then the continuous-time Lyapunov equation sequence can be transformed into

$$X_{i+1} = \tilde{A}_i'X_{i+1}\tilde{A}_i + \tilde{F}_i'R\tilde{F}_i + \tilde{C}_i'\tilde{C}_i \quad (8.14)$$

where $\tilde{F}_i = \sqrt{2}F_i(I - A + BF_i)^{-1}$ and $\tilde{C}_i = \sqrt{2}C(I - A + BF_i)^{-1}$. The idea of the TSQR method in Chapter 7 can be adapted to this situation as follows.

Algorithm 16 TSQR for the CRE

1. *Decompose $X_i = S_iS_i'$. The initial guess S_0 can be formed by the Arnoldi algorithm on $(I - A)^{-1}(I + A)$ and a random vector v, or can be chosen as a low-rank random matrix. Note the rank should be at least the number of unstable eigenvalues.*

2. *Specify integers N and K. For each i from 0 to $N-1$,*

   - *Let $S_{i+1}^{(0)} = S_i$. $F_i = R^{-1}B'S_iS_i'$ and compute $\tilde{A}_i$, $\tilde{F}_i$ and $\tilde{C}_i$.*
   - *For j from 0 to $K-1$, use SVD decomposition on $(\tilde{A}_i'S_{i+1}^{(j)},\ R^{-1/2}\tilde{F}_i',\ \tilde{C}_i')$ and truncation to update $S_{i+1}^{(j+1)}$ and $S_{i+1} = S_{i+1}^{(K)}$.*

3. *Stop if $\|F_{i+1} - F_i\|$ reaches a specified tolerance.*

Of course, we do not compute $\tilde{A}_i$ explicitly. If we take C as zero, the rank of the solution X of the CRE will be the number of unstable eigenvalues of A, so Algorithm 16 should give the exact solution. If we choose C nonzero, the truncated version could have trouble producing a stabilizing feedback. This can be improved by applying SSI on C as a preprocessing. See Chapter 7 for a discussion on choosing the rank of iteration and C.

8.6 Stabilization via the DRE

In previous chapters, we have discussed the use of the DRE to find a stabilizing feedback F. Earlier in this chapter, it was mentioned that the continuous-time Lyapunov equation can be transformed into discrete-time Lyapunov equation. If a similar technique could be developed to transform the CRE into DRE then all of the techniques of the previous chapters relative to the DRE can be used to stabilize continuous-time systems.

In [40] Hung and Chu develop a relationship between the general DRE and the general CRE with the linear rational transformations we discussed earlier. The general DRE has the form

$$P = \tilde{A}'P\tilde{A} - (\tilde{A}'P\tilde{B} + \tilde{N})(\tilde{R} + \tilde{B}'P\tilde{B})^{-1}(\tilde{B}'P\tilde{A} + \tilde{N}') + \tilde{Q} \tag{8.15}$$

and general CRE has the form

$$PA + A'P + Q - (PB + N)R^{-1}(B'P + N') = 0. \tag{8.16}$$

The following transformation links them:

$$\begin{aligned}
\tilde{A} &= \alpha(I - A)^{-1}(I + A), \\
\tilde{B} &= \sqrt{2}(I - A)^{-1}B, \\
\tilde{Q} &= 2(I - A)^{-T}Q(I - A)^{-1}, \\
\tilde{N} &= (I - \alpha A')^{-1}(\sqrt{2}N + \alpha Q\tilde{B}), \\
\tilde{R} &= R + \frac{\alpha}{\sqrt{2}}(N'\tilde{B} + \tilde{B}'N) + \frac{1}{2}\tilde{B}'Q\tilde{B}.
\end{aligned}$$

where α has absolute value 1 and $-\alpha$ can not be any eigenvalue of $\tilde{A}$. Under such a transformation, the general DRE and the general CRE will have the same solution whence a stabilizing feedback can be derived.

Theorem 8.1 *([40])Let $|\alpha| = 1$ and $-\alpha$ is not any eigenvalue of $\tilde{A}$. Then the following two statements are equivalent:*

1. *P is a solution to the general DRE (8.15).*

2. *P is a solution to the general CRE (8.16).*

Define $\tilde{\Delta}(P) = \tilde{A} - \tilde{B}(\tilde{R} + \tilde{B}'P\tilde{B})^{-1}(\tilde{B}'P\tilde{A} + \tilde{N}')$ and $\Delta(P) = A - BR^{-1}(B'P + N')$. If z is an eigenvalue of $\tilde{\Delta}(P)$, then $s = (z - \alpha)/(z + \alpha)$ is an eigenvalue of $\Delta(P)$, and vice versa. Clearly we also have $|z| < 1$ is equivalent to $REAL(s) < 0$.

To use the DRE to find a continuous-time stabilizing feedback, we need to build a DRE with parameters derived from the CRE and to transform back with above theorem. To stabilize the continuous-time system (A, B), first take $\alpha = 1$. If 1 is an eigenvalue of A, then scale A by a constant, α, such that $I - \alpha A$ is nonsingular. (A, B) is transformed to discrete-time system $(\tilde{A}, \tilde{B})$ by:
$\tilde{A} = (I - \alpha A)^{-1}(I + \alpha A)$,
$\tilde{B} = \sqrt{2}(I - \alpha A)^{-1}B$.
We then use $(\tilde{A}, \tilde{B})$ to build a DRE that can produce a stabilizing solution with the

DRDE-based or Newton-based algorithms we have discussed in previous chapters. The $\tilde{R}$, $\tilde{Q}$ (and L_0 or C) are chosen as needed by SQR, CSQR or Newton's iteration to achieve best results. The $\tilde{Q}, \tilde{R}, \tilde{N}$ are transformed back to Q, R, N. Theorem 8.1 will then give us a continuous-time stabilizing feedback for the system (A, B). Note the DRE used is the form we have used in previous chapters, but the CRE is in the general form.

There are several algorithms that use the DRE and DRDE to stabilize a continuous-time system. Since we do not use the general DRE, $\tilde{N}$ is taken as zero here. If we also choose $\tilde{Q}$ as zero on the discrete-time side, we have Q and N as zero on continuous-time side. With $\tilde{N}$ and $\tilde{Q}$ as zero, we can build the SQR and TSQR iterations with the rank the same as the number of continuous-time unstable eigenvalues of A. The advantage with SQR or TSQR is that these two algorithms can generate the invariant subspace basis sequence with only small additional computations. From these bases we can extract an approximate continuous-time unstable invariant subspace. This also give us a good way to measure the stability of corresponding closed-loop matrix. This stability measure is achieved by comparing the eigenvalues of the projections of A and corresponding closed loop on the approximated unstable invariant subspace of A (see Chapter 5 for details). The disadvantage is that we do not have an effective method to accelerate both the stabilization process and convergence.

If we use the CSQR algorithm, implying nonzero $\tilde{Q}$, we have to be very careful about Q and N since both will be nonzero. The advantage of CSQR is its true low-rank that is independent of the number of unstable eigenvalues. The acceleration of CSQR for continuous-time system needs further research, since the feedback increment is no longer a good stopping criterion for some acceleration methods of CSQR and the criterion based on $(A - BF_i) \cdots (A - BF_1)B$ no longer works in the continuous-time case.

Regardless of the algorithm used, we cannot compute continuous-time feedback F from $\tilde{K}$ and R^{ϵ}. we must exploit the linear relationship $F = R^{-1}B'P$ and compute F_i from $R^{-1}B'P_i^{1/2}(P_i^{1/2})'$ in SQR and TSQR, and ΔF_i from $R^{-1}B'L_iL_i'$ in CSQR. Since $B'P_i^{1/2}$ is required by SQR and TSQR and $B'L_i$ is required by CSQR, the computation of F_i increases the computational cost insignificantly.

The biggest disadvantage in using DRE-based algorithms to stabilize continuous-time systems is the computations of form $\tilde{A}v$ require the solution of large sparse linear systems. For some special cases, where the spectral radius of A is relative small, we can use low-order Taylor expansions with scaling on A to approximate a linear rational transformation, as we discussed earlier for SSI. Numerical experiments show the difference of feedback and corresponding closed loop from using linear fraction transformation to get $\tilde{A}$ from A and using a polynomial of degree 8 which is from Taylor expansion of the same linear fraction transformation is not significant.

However, for a problem like Example 22 where the spectral radius of A is very large, we must use the linear rational transformation and use reliable iterative methods to solve the resulting linear systems.

Another way to use the DRE to stabilize a continuous-time system without the trouble of matrix inverse computation is "pole assignment in a specified disk" from [38]. With this method, we first transform (A, B) with a shift and scaling to $((A - \alpha)/r, B/r)$. If the system $((A - \alpha)/r, B/r)$ is stabilized by feedback F in discrete-time such that all eigenvalues of $(A-\alpha)/r-(B/r)F$ are inside unit circle, all eigenvalues of $A - BF$ will be in a disk with radius r centered at α. The DRE built on the new system $((A - \alpha)/r, B/r)$ and stabilizes it in discrete-time. Comparing to above algorithms which use linear rational transformation or its approximation, this algorithm has a big advantage for large sparse systems. In fact, when applied to stabilize a continuous-time system, the "specified disk pole assignment" algorithm is exactly the direct discretization algorithm we have discussed at the beginning of this chapter. So, this method has the same advantage and disadvantage as the direct discretization method. Whether this method and direct discretization method are effective depends on whether we can find a disk in the left complex plane such that only very few eigenvalues (including all unstable eigenvalues) of A are outside the disk.

8.7 Empirical Results

In this section, we compare SSI, SQR, CSQR, Chebyshev, and direct discretization methods for the stabilization of continuous-time systems. Efficient versions of the Newton-based approaches are not yet to the point of meaningful empirical testing.

Example 22 *(Figure 8.3) The continuous-time linear system (A, B) in this example is taken from the work of He and Mehrmann [8]. It has the form $A = Q(D + N)Q'$, where Q is a random orthogonal matrix and $N(i, i + 30) = \alpha$ and all others are zero with $\alpha = 10$. D has the form $D = diag(D_1, \cdots, D_{350}, 1, 2, \cdots, 10)$ with $D_k = \begin{bmatrix} x_k & y_k \\ -y_k & x_k \end{bmatrix}$ where $x_k = -0.1 * y_k^2$, $y_k = -0.1k$, $k = 1, \cdots, 350$. B is taken as a random sparse matrix of 15 columns. The only unstable eigenvalues of A are $1, 2, \cdots, 10$. $P_0^{1/2}$ is taken as random sparse matrix of 10 columns.*

First, we transform this system to discrete-time form and use SQR algorithm to compute $P_i^{1/2}$ and F_i. Note we need scale A by some constant to avoid singularity of $I - \alpha A$. We run 200 iterations of SQR. The results are shown in Figure 8.3. The convergence and stopping criteria are measured by $\log_{10} \|F_i - F_{i-1}\|_2/\|F_i\|_2$, $\log_{10} \|A'V_i - V_i(V_i'A'V_i)\|_2$ and the eigenvalue comparison between projection of A and $A - BF_i$ on V_i where V_i is approximated unstable invariant subspace of A extracted from sequence $P_i^{1/2}$. (Using the eigenvalue comparison as a stopping criterion is detailed in Chapter 5 and we need only change it to the continuous-time equivalent). Within 100 steps, the unstable eigenvalues of A are stabilized to $-1, -2, \cdots, -10$ to an accuracy 10^{-6}, as expected, with feedback $\|F\|_2 = 42.9988$. Both the associated subspace sequence an feedback sequence converge to an accuracy 10^{-6} within 100 steps.

Increasing the rank of $P_i^{1/2}$ *does not help feedback convergence but, it does accelerate convergence of unstable subspace sequence extracted from* $P_i^{1/2}$.

When using CSQR to compute δP_i*, the feedback convergence rate is close to the feedback convergence rate of SQR with rank higher than 10, where it is the spectral radius of corresponding discrete-time closed-loop matrix that dominates, as discussed in Chapter 5. However, CSQR does not generate the approximate unstable invariant subspace that can help detect stabilization much earlier than feedback convergence.*

If we use a linear rational transformation to transform the system to its equivalent DRE and solve using SQR, 100 iterations suffices for an accuracy of 10^{-13} *for all unstable eigenvalues* $1, \cdots, 10$*, and a residual* $A'V - V(V'A'V)$ *with norm* 10^{-5}.

The Chebyshev SSI does not perform well when used to compute approximate unstable invariant subspace since these stable eigenvalues are spread over too large an area and it is hard to find a suitable small ellipse to contain them as required by Chebyshev iteration. For example, taking the initial subspace V_0 *as a* 710×40 *orthogonal matrix and* $\lambda_1,\ c,\ e$ *as* $1,\ -70,\ 45$ *and using Chebyshev SSI along with SRR refinement, yields after 200 steps good accuracy on all unstable eigenvalues except* $\lambda = 1$ *which has an error of* -0.02 *and the norm of residual* $A'V - V(V'A'V)$ *for the approximate unstable subspace* V *is 0.9251. Even after 400 steps, the is only* 10^{-3} *for* $\lambda = 1$ *and the norm of residual* $A'V - V(V'A'V)$ *is 0.0071. The trouble stems from the inability of the parameters* $\lambda_1,\ c,\ e$ *to determine an ellipse that can separate the eigenvalue* $\lambda = 1$ *from the ellipse. Direct discretization also works poorly for this example.*

Example 23 *In this example, we create a* 100×100 *matrix with MATLAB RANDN function and shift by 7.2 to get* A *which has 6 unstable eigenvalues and all stable eigenvalues are located in* $[-20, 0] \times [-10, 10]$*.* B *is generated by RAND as a random vector. Figure 8.4 and Figure 8.5 show the results for several methods.*

Using Chebyshev with λ_1, c, e *as* $3, -10, 10i$*, and taking the initial subspace as a* 100×10 *matrix, yields a norm of the residual of unstable subspace of* 10^{-4} *after 200 iterations, and a norm of* 10^{-8} *within 400 iterations.*

Using a linear rational transformation followed by a rank 6 SSI to generate the approximate unstable invariant subspace reaches a subspace residual accuracy of 10^{-5} *within 100 iterations* 10^{-10} *within 200 iterations So once again, projection methods based on the rational transformation produce superior convergence at the cost of increased work per iteration to solve the linear system. (Indeed, controlling this extra work is the key performance issue for this and any other approach that requires the solution of linear systems.)*

If we transform the CRE to a DRE using a linear rational transformation and stabilize via a rank 6 SQR, the feedback increment reaches 10^{-12} *in 200 iterations. Furthermore, by comparing the eigenvalues of projections of* A *and* $A - BF_i$ *on the approximate unstable invariant subspace, stabilization is detected within 50 steps (circles in Figure 8.5).*

If the same CRE to DRE transformation is done and then solved with rank 2 CSQR, feedback converges slightly slower and reaches 10^{-6} *in 200 iterations. However, as noted in our discussions we have no simple stabilization detection heuristic for CSQR other than feedback convergence and therefore cannot detect stabilization early. In fact, Figure 8.4 shows that SQR stabilizes the system* (A, B) *in 30 iterations and CSQR stabilizes the system in 60 iterations.*

8.8 Summary

In order to stabilize large scale continuous-time systems, most algorithms, except Chebyshev polynomial SSI, must transform to some discrete-time form, directly or indirectly. Direct discretization and the Chebyshev method have the advantage of maintaining the sparse pattern of A but, both are limited to the situation where the stable eigenvalues of A are concentrated in a ellipse of with a relatively small

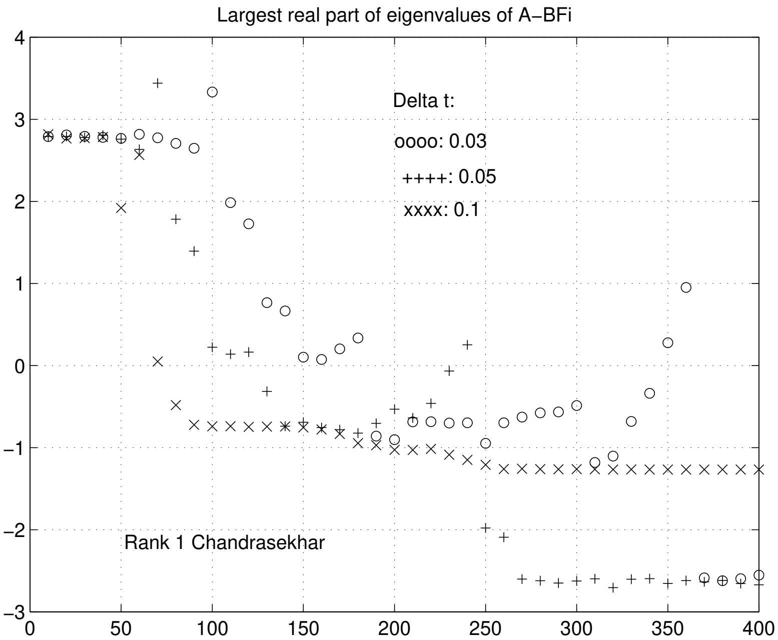

Figure 8.1. Stabilization of continuous-time system via direct discretization + rank 1 Chandrasekhar, stopping criterion, Example 21

semi-axis or a disk of relatively small radius and it is very hard to estimate the parameters used to construct the algorithms. Other algorithms which use linear rational transformations are easier to implement but, the computations to solve the linear systems for each iteration can be expensive. When the spectral radius of A is small (for example, less than 10), the linear rational transformation can be replaced by low-order polynomials such as a Taylor expansion, that save storage and computations, especially for large sparse system. When using direct discretization, CSQR is recommended to stabilize the corresponding discrete-time system because of its true low-rank nature. When using linear rational transformations to generate a DRE from a CRE, SQR is preferred because it can generate the same subspace sequence as SSI and stabilization is much easier to detect via the ΔV criterion.

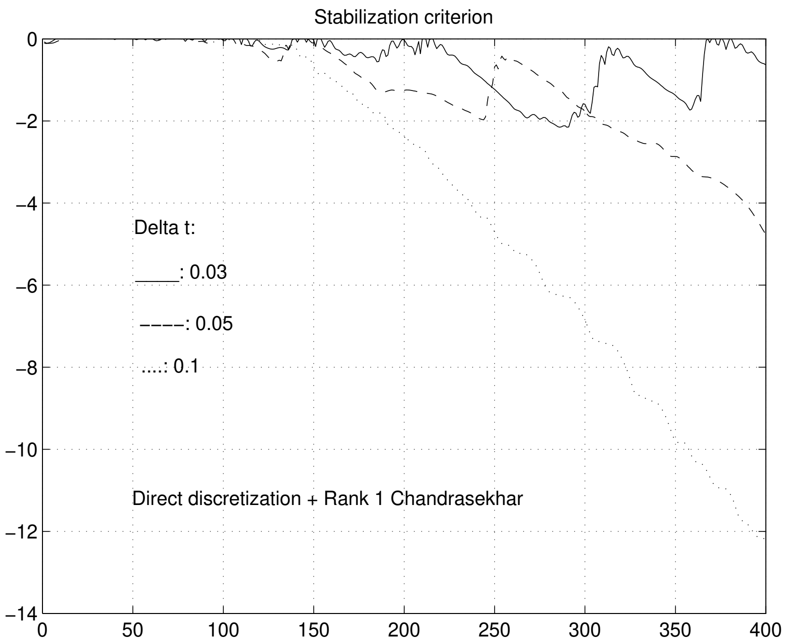

Figure 8.2. Stabilization of continuous-time system via direct discretization + rank 1 Chandrasekhar, Example 21

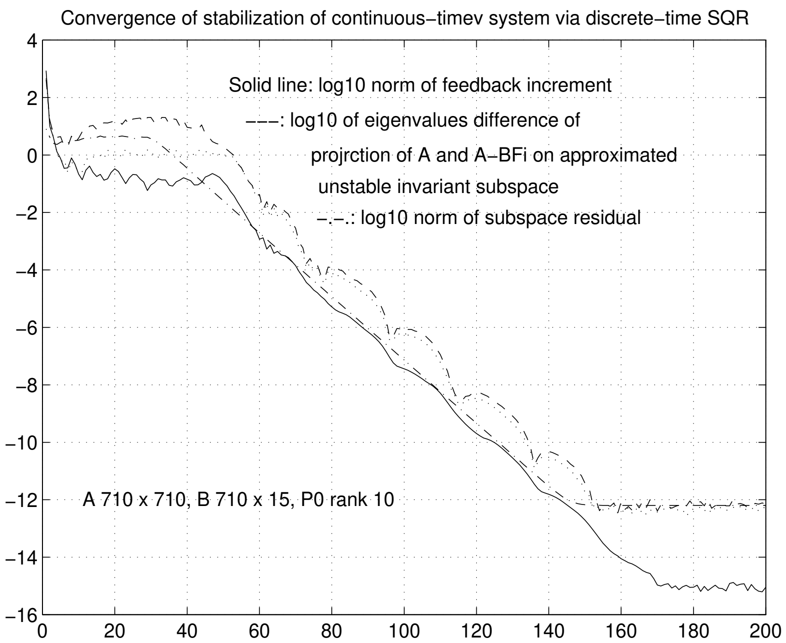

Figure 8.3. Rank 10 SQR: Stabilization of continuous-time system via discrete-time SQR, Example 22

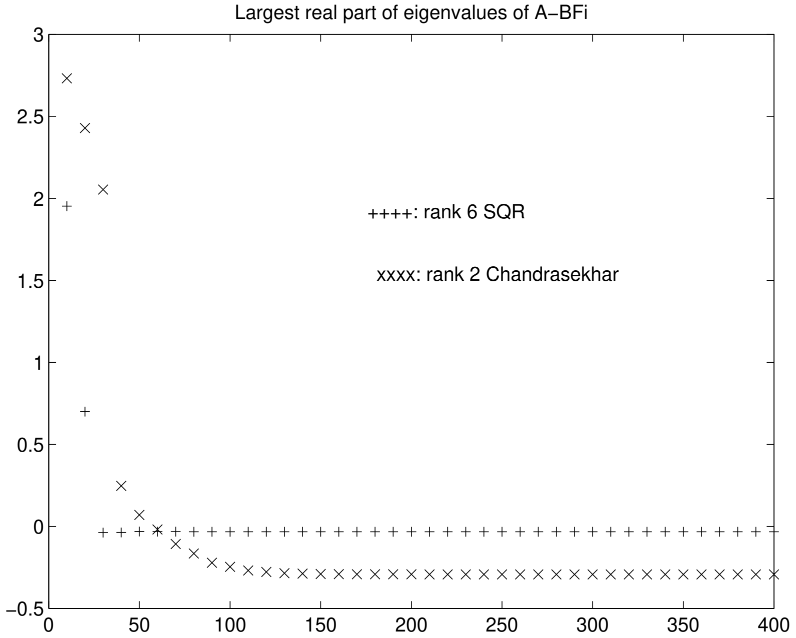

Figure 8.4. Stabilization of continuous-time system via discrete-time SQR and CSQR, Example 23

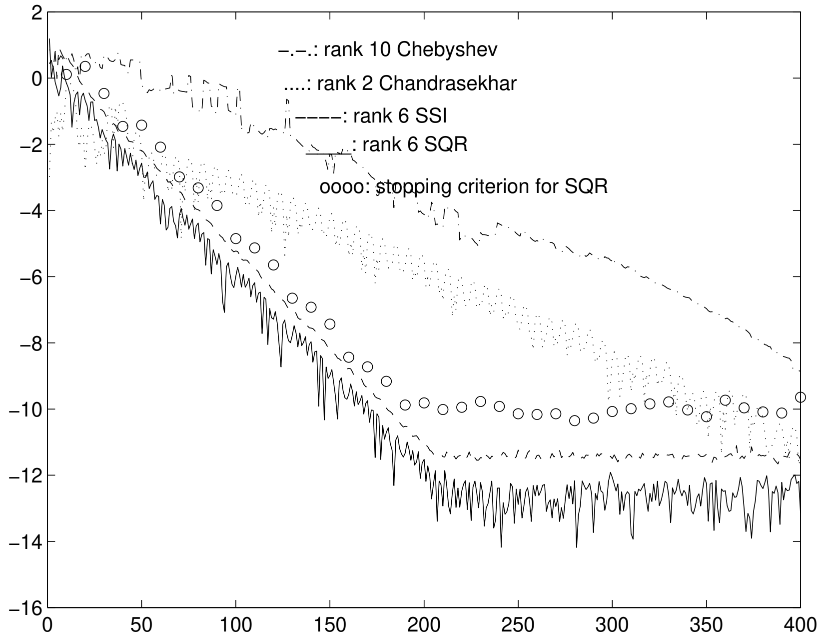

Figure 8.5. Convergence of different algorithms used to stabilize continuous-time system: SQR, SSI, CSQR and Chebyshev, Example 23

CHAPTER 9

CONCLUSIONS AND FUTURE WORK

In this thesis, we have discussed, in detail, the stabilization of large linear dynamical systems. We have discussed projection methods and proposed a hybrid algorithm which combines SSI, projection and feedback computing and changed the typical projection method from two phases to one phase. We have successfully implemented and improved algorithms used in Kalman filtering such as DRE-based square root algorithms SQR and CSQR to effectively stabilize large scale systems. We also proposed a new method TSQR based on approximation of a Newton iteration on the DRE that can be used to stabilize large systems and approximate the solution of the DRE and solve Kalman filtering problems. Different methods to measure the difficulty of stabilizing linear systems were discussed and two metrics, the condition number for stabilizability of (A, B) and stability of the matrix triple (A, B, F) were developed.

For a well-conditioned discrete-time system where A has only very few unstable eigenvalues and the eigenvalues of A are well-separated from the unit circle, all of algorithms in this thesis, SSI, SQR, CSQR and TSQR, perform very well. If A has many eigenvalues close to unit circle, the efficiency of SSI, SQR, CSQR and TSQR can degrade but, two refinements of CSQR, restarted and scaled perform well. The rank of projection-based algorithms, SQR and TSQR is at least the number of unstable eigenvalues of A, so these algorithms are not scalable. CSQR is a true low-rank algorithm. Its rank is at most the maximal multiplicity of unstable eigenvalues of A, which is independent of the number of unstable eigenvalues.

The difficulty in stabilizing a system can be predicated by the condition number of stabilizability defined in this thesis. If all algorithms in this thesis fail to reach a stabilizing feedback, this condition number should be very large. To compare the results from different stabilization algorithms and different parameters, the stability radius of (A, B, F) provides a way to choose a reasonable feedback, i.e., the one that has the largest stability radius.

We have shown the relationship between SSI and SQR and the advantage of SQR over SSI. We have developed a stopping criterion based on the projections of A and $A - BF_i$ onto the approximate invariant subspace sequence extracted from SQR. This stopping criterion can detect the stability of $A - BF_i$ much earlier than any other standard criterion such as feedback convergence and subspace residual. For CSQR, where the the spectral radius of $A - BF_i$ is not close to 1, we also developed a new stopping criterion based on the fact that $\prod_{j=1}^{i}(A - BF_j)B \to 0$. This stopping criterion was successfully used in scaled CSQR and restarted CSQR.

Continuous-time systems can be stabilized by transforming into discrete-time via direct discretization or linear rational transformation. While direct discretization can exploit sparsity of A more efficiently than a linear rational transformation, stronger constraints are imposed on the eigenvalue distribution of A and the optimal Δt is yet to be found. With linear rational transformation and the relationship between the general CRE and the general DRE, the stabilization algorithms used for discrete-time systems such as SSI, SQR, CSQR and TSQR can be adapted to a continuous-time system.

There are several possible avenues of future research.

1. A general framework unifying and possible extending all of the methods discussed in this thesis appears possible and promising for new insights into this and related problems such as optimal filtering, optimal control and data assimilation.

2. Much work remains on the efficient implementation of the most promising algorithms in the thesis.

3. In Chapter 2.2, $\sigma_n(S)$ and $r_{\mathbb{C}}(A, B, F)$ are presented to measure conditioning of stabilization problems. While some estimates are presented, for large scale systems, we need to find efficient algorithms to compute both metrics.

4. Analytical work is needed on the perturbation of feedback and stability radii $r_{\mathbb{C}}(A, B, F)$ for different stabilization algorithms.

5. A metric or metrics to compare the quality of the feedbacks from different algorithms is needed.

6. In Chapter 5, the condition on P_0 for DRDE convergence is relaxed for the special case where $Q = 0$. Research is needed on the conditions on P_0 for a general DRDE where $Q \neq 0$.

7. Since all of the algorithms we have presented are iterative algorithms, we need efficient stopping criterion to terminate the iteration as early as possible. For stabilization purposes, whatever algorithm we use, our focus should be to find a stabilizer, not to assess convergence of the iteration. For SQR, we have proposed a stabilization criterion that can detect stability very early but, the correctness of this criterion has not been theoretically proved. For SSI, we need to determine how small the subspace residuals should be so that the projection method on the approximate unstable invariant subspace can produce a stabilizer in the original space. For CSQR stabilization occurs much earlier than feedback convergence. While the stabilization criterion from $(A - BF_i) \cdots (A - BF_1)B$ may detect the stability of $A - BF_i$ much earlier than feedback convergence if the spectral radius of corresponding closed-loop is not too close to 1, it is not as reliable as the methods developed for SQR and projection-based methods.

8. Some stopping criteria need estimates of the (sub)optimal stability radius of the system (A, B), that is, the optimal stability radius of $A - BF$ over all stabilizers F. Even for small systems, the estimation of such a stability radius is not easy because of the existence of multiple local minima or maxima. Efficient algorithms for large sparse systems are needed.

9. SQR has the form $P_{i+1}^{1/2} = (A'P_i^{1/2})U_i^{22}$ from which its similarity to SSI and other eigensolvers is seen. It should be possible to borrow something from eigenvalue algorithms to accelerate this iteration. For example, as in subspace iteration where large eigenvalues converge first, SQR stabilizes large eigenvalues first and fast, but the total convergence depends on the worst (smallest unstable) eigenvalue. If we first stabilize and filter out these eigenvalues that are easy to stabilize and then focus on the tough eigenvalues, the total computation cost should be saved. Some deflation/restarted eigenvalue algorithms might be useful to construct a deflation/restarted SQR.

10. CSQR is a true low-rank stabilization algorithm (the rank is independent to the number of unstable eigenvalues). There are some efficient modifications such as preprocessing C, restarting and scaling A. To preprocess C, we should further consult the method to preprocess initial vector for Arnoldi (see [10]).

11. For restarted CSQR, research to determine the number of iterations in the inner loop to balance stabilization speed and closed-loop eigen-conditioning is needed.

12. The most efficient CSQR method for a general system should combine all or several modifications of CSQR together and further research on doing this combination efficiently is needed.

13. In CSQR, we have limited one freedom of parameter choice of DRDE: we choose $P_0 = 0$. The implication is, we have to accumulate P_i from A and

Q with something close to SSI. If we can freely choose P_0, the total steps of iteration could be reduced significantly. Further research on the condition of P_0 such that CSQR converges and stabilizes rapidly is needed.

14. In Chapter 7, inexact Newton's algorithm and TSQR are presented, where the approximate $P_{i+1}^{1/2}$ is updated by truncating the SVD of
$$\begin{bmatrix} (A-BF_i)'P_i^{1/2} & Q^{1/2} & F_i'R^{1/2} \end{bmatrix}.$$
The rank of $P_i^{1/2}$ and the conditions for the algorithms to converge and stabilize needs further research.

15. Efficient implementation of TSQR needs investigation. This centers on the role of the approximation of the Lyapunov equation required on each step. Since the system is unstable iterative methods for the Lyapunov method will tend to diverge. We must understand the way in which this divergence is overcome and convergence of the TSQR iteration ensured.

16. In Chapter 8, most algorithms to stabilize continuous-time system require conformal transformations that transform the continuous-time stabilization problem to a discrete-time system problem. Unfortunately, undesired large scale linear equation solutions are often required for this type of transformation. The effect of the approximation arising from the use of iterative methods to solve the linear systems or replacing them by polynomial approximations must be analyzed.

17. Efficient implementations of Newton-based DRE methods for continuous-time problems must be considered.

18. Effective stabilization detection heuristics for CSQR applied to the DRE arising from the general CRE associated with continuous-time problems must be investigated.

REFERENCES

[1] N. K. Nichols J. Kautsky and P. van Dooren. Robust pole assignment in linear state feedback. *Int'l. J. Contr.*, 41(5), 1985.

[2] V. Mehrmann and H. Xu. Choosing poles so that the single-input pole placement problem is well conditioned. *SIAM J. Matrix Anal. Appl.*, 19(3), 1998.

[3] M. Arnold and B. N. Datta. Single-input eigenvalue assignment algorithms: a close look. *SIAM J. Matrix Anal. Appl.*, 19(2), 1998.

[4] G. Miminis and C. C. Paige. An algorithm for pole assignment of time invariant linear systems. *Int'l. J. Contr.*, 35, 1981.

[5] G. Miminis and C. C. Paige. An direct algorithm for pole assignment of time invariant multi-input linear systems using state feedback. *IEEE Trans. Automat. Contr.*, 34(6), 1989.

[6] N. D. Christov P. H. Petkov and M. M. Konstantinov. A computational algorithm for pole assignment of linear multiinput systems. *IEEE Trans. Automat. Contr.*, 31(11), 1986.

[7] M. Arnold and B. N. Datta. An algorithm for the multiinput eigenvalue assignment problem. *IEEE Trans. Automat. Contr.*, 35(10), 1990.

[8] C. Y. He and V. Merhmann. Stabilization of large linear systems. Technical report, TU Chemnitz-Zwichau, 1997.

[9] E. S. Armstrong. An extension of bass's algorithm for stabilizating linear contiunuous constant systems. *IEEE Trans. Automat. Contr.*, 20, 1975.

[10] Y. Saad. Projection and deflation methods for partial pole assignment in linear state feedback. *IEEE Trans. Automat. Contr.*, 33(3), 1988.

[11] G. W. Stewart. Simultaneous iteration for computing invariant subspace of non-hermitian matrices. *Numer. Math.*, 25, 1976.

[12] M. R. Gevers C. E. De Souza and G. C. Goodwin. Riccati equations in optimal filtering of nonstabilizable systems having singular transition matrices. *IEEE Trans. Automat. Contr.*, 31(9), 1986.

[13] M. Morf and T. Kailath. Square-root algorithms for least-squares estimation. *IEEE Trans. Automat. Contr.*, 20(4), 1975.

[14] W. M. Wonham. *Linear multivariable control : a geometric approach.* New York: Springer-Verlag, 1985.

[15] J. G. Sun. Perturbation analysis of system hessenberg and hessenberg-triangular forms. *Lin. Alg. Appl.*, 241-243, 1996.

[16] J. G. Sun. Perturbation theory for algebraic riccati equations. *SIAM J. Matrix Anal. Appl.*, 19(1), 1998.

[17] J. G. Sun. Perturbation analysis of the pole assignment problem. *SIAM J. Matrix Anal. Appl.*, 17(2), 1998.

[18] G. H. Golub and C. F. Van Loan. *Matrix computations.* The John Hopkins Univ. Press, 1983.

[19] P. Van Dooren and V. Vermaut. On stability radii of generalized eigenvalue problems. Technical report, Belgian Programme on Interuniversity Poles of Attraction, 1999.

[20] P. E. Caines and D. Q. Mayne. On the discrete time matrix riccati equation of optimal control. *Int'l. J. Contr.*, 12(5), 1970.

[21] W. W. Hager and L. L. Horowitz. Convergence and stability properties of discrete riccati operator equation and the associated optimal control and filtering problems. *SIAM J. Contr. Optimiz.*, 14(2), 1976.

[22] B. D. O. Anderson and J. B. Moore. *Optimal Filtering.* Englewood Cliffs, N.J. : Prentice-Hall, 1979.

[23] G. D. Nicolao and M. R. Gevers. Difference and differential riccati equations: A note on the convergence to the strong solution. *IEEE Trans. Automat. Contr.*, 37(7), 1992.

[24] P. Van Dooren. A generalized eigenvalue approach for solving riccati equations. *SIAM J. Sci. Stat. Comput.*, 2(2), 1981.

[25] A. J. Laub. A schur method for solving algebraic riccati equations. *IEEE Trans. Automat. Contr.*, 24(6), 1979.

[26] A. J. Laub T. Pappas and Jr. N. Sandell. On the numerical solution of the discrete-time algebraic riccati equations. *IEEE Trans. Automat. Contr.*, 25(4), 1980.

[27] M. Verhaegen and P. Van Dooren. Numerical aspects of different kalman filter implementations. *IEEE Trans. Automat. Contr.*, 31(10), 1986.

[28] C. Kenny T. Gudmundsson and A. J. Laub. Scaling of the discrete-time algebraic riccati equation to enhance stability of the schur solution method. *IEEE Trans. Automat. Contr.*, 37(4), 1992.

[29] L.V. Kantorovich and G. P. Akilov. *Functional Analysis.* Oxford ; New York : Pergamon Press, second edition, 1982.

[30] J. Demmel. Three methods for refining estimates of invariant subspace. *Computing*, 38, 1987.

[31] S. Hammarling J. J. Dongarra and J. H. Wilkinson. Numerical considerations in computing invariant subspace. *SIAM J. Matrix Anal. Appl.*, 13(1), 1992.

[32] R. B. Lehoucq and D. C. Sorensen. Deflation techniques for an implicitly restarted arnoldi iteration. *SIAM J. Matrix Anal. Appl.*, 17(4), 1996.

[33] G. W. Stewart. Error and perturbation bounds for subspace associated with certain eigenvalue problems. *Numer. Math.*, 15, 1973.

[34] K. Meerbergen and D. Roose. The restarted arnoldi method applied to iterative linear system solvers for the computation of rightmost eigenvalues. *SIAM J. Matrix Anal. Appl.*, 18(1), 1997.

[35] C. Y. He P. Byers and V. Mehrmann. The matrix sign function method and the computation of invariant subspace. *SIAM J. Matrix Anal. Appl.*, 18(3), 1997.

[36] T. Penzl. A cyclic low rank smith method for large sparse lyapunov equations with applications in model reduction and optimal control. Technical report, Technische Universität Chemnitz, 1998.

[37] M. Verlaan. *Efficient Kalman Filtering Algorithms for Hydrodynamic Models.* PhD thesis, Delft University of Technology, 1998.

[38] K. Furuta and S. B. Kim. Pole assignment in a specified disk. *IEEE Trans. Automat. Contr.*, 32(5), 1987.

[39] Chun-Hua Guo. Newton's method for discrete algebraic riccati equations when the closed-loop matrix has eigenvalues on the unit circle. *SIAM J. Matrix Anal. Appl.*, 20(2), 1999.

[40] Y. S. Hung and D. L. Chu. Relationships between discrete-time and continuous-time algebraic riccati inequalities. *Lin. Alg. Appl.*, 270, 1998.

[41] Z. Bai and J. Demmel. Using the matrix sign function to compute invariant subspace. *SIAM J. Matrix Anal. Appl.*, 19(1), 1998.

[42] J. D. Gardiner and A. J. Laub. A generalization of the matrix-sign-function solution for algebraic riccati equations. *Int'l. J. Contr.*, 44(3), 1986.

[43] S. J. Hammarling. Numerical solution of the stable, non-negative definite lyapunov equation. *J. of Numerical Analysis*, 2, 1982.

[44] Y. Saad. *Numerical methods for large eigenvalue problems.* Halstead Press (John Wiley), 1992.

BIOGRAPHICAL SKETCH

Xiang Rao

Xiang Rao received his BS degree in mathematics from Peking University, China, in 1985 and MS degree in applied mathematics from Northern Jiaotong University, China, in 1988.

From 1988 to 1995, he held lecturer position at Northern Jiaotong University. From 1995 to 1999, he was teaching assistant and research assistant at Florida State University.