

**A CONJUGATE GRADIENT METHOD FOR SOLVING MATRIX ALGEBRAIC RICCATI
EQUATIONS**

by

Lixin Liu

B.Sc., Beijing Normal University, 1985

**THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE**

in the Department

of

Mathematics and Statistics

© Lixin Liu 1988

SIMON FRASER UNIVERSITY

September 1988

All rights reserved. This work may not be reproduced in whole or in part, by photocopy or other means, without permission of the author.

APPROVAL


Name: Lixin Liu

Degree: Master of Science

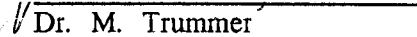
Title of thesis: A CONJUGATE GRADIENT METHOD FOR SOLVING MATRIX ALGEBRAIC
RICCATI EQUATIONS

Examining Committee:

Chairman: Dr. G. Bojadziev



Dr. R. D. Russell
Senior Supervisor



Dr. M. Trummer

Dr. R. Lardner

Dr. B. Bhattacharya
External Examiner
Department of Computing Science
Simon Fraser University

Date Approved: September 26, 1988

PARTIAL COPYRIGHT LICENSE

I hereby grant to Simon Fraser University the right to lend my thesis, project or extended essay (the title of which is shown below) to users of the Simon Fraser University Library, and to make partial or single copies only for such users or in response to a request from the library of any other university, or other educational institution, on its own behalf or for one of its users. I further agree that permission for multiple copying of this work for scholarly purposes may be granted by me or the Dean of Graduate Studies. It is understood that copying or publication of this work for financial gain shall not be allowed without my written permission.

Title of Thesis/Project/Extended Essay

A conjugate ~~gradient~~ gradient method for
solving algebraic Riccati equations

Author: _____

(signature)

LIXIN LIU

(name)

Nov. 22, 1988

(date)

ABSTRACT

Recently, algebraic Riccati equations (AREs) have been widely solved in many fields. In this thesis, some applications will be discussed. These applications include computing ill-conditioned eigenproblems (refining invariant subspaces); solving optimal control problems (computing optimal control functions for both continuous and discrete time systems, and solving singular perturbation problems of dynamic systems); and decoupling boundary value problems for ordinary differential equations.

Several numerical methods for solving AREs have been developed in the past 20 years. These methods can be divided into two classes, direct methods (the Schur method and the symplectic method) and iterative methods (linear iterative methods, Newton's method and the generalized secant method). We will review these methods with some comparisons and implementations. In the case of large sparse AREs, these methods will lose their efficiency because matrix factorizations (LU-decomposition or QR-factorization) are needed. A "new" method for solving AREs which may be particularly suitable for solving large sparse problems, the conjugate gradient method, will be suggested in this thesis. We will compare this method to the other methods (the Schur method, the generalized secant method and Newton's method) in terms of convergence, cost and storage requirements.

DEDICATION

To my parents and my sister

ACKNOWLEDGEMENTS

I would like to express my sincere thanks to my supervisor, Dr. R. D. Russell, for his suggestion and guidance about the research of this thesis.

I would also like to thank Dr. L. Dieci and Dr. M. Trummer for their help and cooperation. And finally, thank Simon Fraser University and Department of Mathematics & Statistics for giving me the opportunity to study here, especially thank Mrs. S. Holmes for her help.

Financial support for the research of this thesis received from Dr. Russell's research grant is also appreciated.

TABLE OF CONTENTS

Approval		ii
Abstract		iii
Dedication		iv
Acknowledgements		v
1. Introduction		1
2. Applications of Solving Algebraic Riccati Equations		7
2.1 Computing Invariant Subspaces		7
2.1.1 Methods for Computing Invariant Subspaces		9
2.1.2 The Gap between Invariant Subspaces		11
2.2 Optimal Control Problems		12
2.2.1 Continuous-time Optimal Control Problems		12
2.2.2 Discrete-time Optimal Control Problems		13
2.2.3 Time-optimal Control Systems with Slow and Fast Modes		14
2.3 Decoupling Boundary Value Problems		17
3. Methods for Solving Algebraic Riccati Equations		20
3.1 Direct Methods for Solving Algebraic Riccati Equations		20
3.1.1 Schur Method		20
3.1.2 Symplectic Method		22
3.2 Iterative Methods for Solving Algebraic Riccati Equations		23
3.2.1 Kokotovic's Linear Iterative Method		23
3.2.2 Stewart's Linear Iterative Method		24
3.2.3 Newton's Method		25
3.2.4 Generalized Secant Method		27
3.3 Conditioning of Algebraic Riccati Equations		29
3.4 Implementations		30

3.4.1	Iterative Refinement Method	31
3.4.2	Steepest Descent Technique	31
3.4.3	Reordering Eigenvalues	32
4.	Conjugate Gradient Method for Solving AREs	34
4.1	Conjugate Gradient Method for Solving Nonlinear Optimization problems	34
4.1.1	Quadratic Case	34
4.1.2	General Cases	36
4.1.3	Implementations for Conjugate Gradient Method	38
4.2	Conjugate Gradient Method for Solving Algebraic Riccati Equations	40
4.2.1	Conjugate Gradient Algorithm	40
4.2.2	Line Search Algorithm	42
4.2.3	Comparisons and Implementations	45
5.	Numerical Examples	50
6.	Conclusions	58
	References	60

CHAPTER 1

INTRODUCTION

Consider the matrix quadratic equation

$$F(X) := A_{21} + A_{22}X - XA_{11} - XA_{12}X = 0 \quad (1.1)$$

where $A_{11} \in \mathbf{R}^{n \times n}$, $A_{22} \in \mathbf{R}^{m \times m}$, $A_{21} \in \mathbf{R}^{m \times n}$, $A_{12} \in \mathbf{R}^{n \times m}$, and $X \in \mathbf{R}^{m \times n}$. The equation (1.1) is called a matrix *algebraic Riccati equation* (ARE). Particularly, if

$$\hat{F}(X) := \hat{A}_{21} + \hat{A}_{11}^T X + X \hat{A}_{11} - X \hat{A}_{12} X = 0 \quad (1.2)$$

where $m=n$, \hat{A}_{12} and \hat{A}_{21} are $n \times n$ positive semi-definite matrices, then (1.2) is called a *symmetric algebraic Riccati equation*.

The purpose of this thesis is to discuss some numerical methods for solving algebraic Riccati equations. In this chapter, we will review some important concepts and results. Some applications for solving AREs will be discussed in the next chapter. These applications include solving ill-conditioned eigenproblems or computing invariant subspaces, solving optimal control problems, and decoupling boundary value problems (BVPs) for ordinary differential equations (ODEs). In chapter 3, we will review several numerical methods (both *direct* methods and *iterative* methods) for solving AREs. These methods include the *Schur* method, *linear iterative* methods, the *generalized secant* method, and *Newton's* method. Some important theorems which are associated with these methods will be stated in this chapter as well as comparisons and implementations. A "new" method for solving AREs, the *conjugate gradient* (C-G) method, will be proposed and discussed in chapter 4, with the comparisons of some methods in chapter 3 (most comparisons are among *Newton's* method, the *generalized secant* method, the *Schur* method and the *conjugate gradient* method). Numerical examples and results will be given in chapter 5, and finally, the conclusions are in chapter 6.

Definition 1.1 Let the matrix $A = [a_{ij}] \in \mathbf{R}^{n \times n}$.

- (1) A is called *symmetric* if $A^T = A$.
- (2) A is called *upper (lower) triangular* if $a_{ij} = 0$ for all $1 \leq j < i \leq n$ ($1 \leq i < j \leq n$).
- (3) A is called *upper Hessenberg* if $a_{ij} = 0$ for all $1 \leq j < i - 1 \leq n - 1$.
- (4) A is called *orthogonal* if $A^T A = I$.
- (5) If there exists $\lambda \in \mathbf{C}$ and $\mathbf{x} \in \mathbf{C}^n$, $\mathbf{x} \neq 0$ such that $A\mathbf{x} = \lambda\mathbf{x}$, then λ is called an eigenvalue of A and \mathbf{x} is called an eigenvector of A corresponding to the eigenvalue λ . The pair (λ, \mathbf{x}) is called an eigenpair. The set of all eigenvalues of the matrix A is denoted by $\Lambda(A)$.
- (6) If A is symmetric and $\mathbf{x}^T A \mathbf{x} > 0$ (≥ 0) for any $\mathbf{x} \in \mathbf{R}^n$, $\mathbf{x} \neq 0$, then A is called *positive (semi-)definite*.

Definition 1.2

- (1) Let $\mathbf{x} \in \mathbf{R}^n$. The *Euclidean norm* (l_2 -norm) of the vector \mathbf{x} is defined by $\|\mathbf{x}\|_2 = \sqrt{\mathbf{x}^T \mathbf{x}}$.
- (2) Let $A \in \mathbf{R}^{m \times n}$. The *spectral norm* (l_2 -norm) of the matrix A is defined by

$$\|A\|_2 = \max_{\mathbf{x} \in \mathbf{R}^n} \left\{ \frac{\|A\mathbf{x}\|_2}{\|\mathbf{x}\|_2} \mid \mathbf{x} \neq 0 \right\}.$$

The *Frobenius norm* (F -norm) of A is defined by

$$\|A\|_F = \sqrt{\sum_{i=1}^m \sum_{j=1}^n a_{ij}^2}.$$

Theorem 1.3 (QR-Factorization) [19] Let $A \in \mathbf{R}^{m \times n}$, $m \geq n$. Then there exist an orthogonal matrix $Q \in \mathbf{R}^{m \times m}$ and upper triangular matrix $R \in \mathbf{R}^{n \times n}$ such that

$$A = Q \begin{bmatrix} R \\ 0 \end{bmatrix}.$$

Theorem 1.4 (Singular Value Decomposition) [19] Let $A \in \mathbf{R}^{m \times n}$ and $k = \min(m, n)$. Then there exist two orthogonal matrices $U \in \mathbf{R}^{m \times m}$ and $V \in \mathbf{R}^{n \times n}$ such that

$$U^T A V = \text{diag} (\sigma_1, \sigma_2, \dots, \sigma_k) \quad (1.3)$$

where $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_k \geq 0$.

Definition 1.5 The σ_i ($i = 1, 2, \dots, k$) in (1.3) are called the *singular values* of A .

Theorem 1.6 (Real Schur Decomposition) [19] Let A be an $n \times n$ real matrix. Then there exists an orthogonal matrix $Q \in \mathbb{R}^{n \times n}$ such that

$$\tilde{A} = Q^T A Q = \begin{bmatrix} A_{11} & A_{12} & \cdots & A_{1,n-1} & A_{1n} \\ 0 & A_{22} & \cdots & A_{2,n-1} & A_{2n} \\ \cdot & \cdot & \cdots & \cdot & \cdot \\ 0 & 0 & \cdots & A_{n-1,n-1} & A_{n-1,n} \\ 0 & 0 & \cdots & 0 & A_{nn} \end{bmatrix} \quad (1.4)$$

where each diagonal block \tilde{A}_{ii} is either a 1×1 real matrix or a 2×2 real matrix having complex conjugate eigenvalues. Moreover, these diagonal block matrices can appear in any desired order on the diagonal.

Definition 1.7 The $n \times n$ matrix \tilde{A} in (1.3) is called an *upper-Schur form matrix* or *quasi-upper-triangular form matrix*.¹

To compute an upper-Schur form matrix \tilde{A} from a matrix A , one can use the *QR-algorithm* (that is, use *Householder transformations* to transform A to an *upper Hessenberg form* and repeat *Francis' QR-steps* to obtain the *upper-Schur form* \tilde{A}). The total cost of the QR-algorithm is about $10n^3$ flops.

Definition 1.8 The linear matrix equation

¹ A *Lower-Schur form matrix* can be defined accordingly.

$$C X - X D = F \quad (1.5)$$

is called the *Sylvester equation* where $C \in \mathbf{R}^{m \times m}$, $D \in \mathbf{R}^{n \times n}$ and $X, F \in \mathbf{R}^{m \times n}$.

Lemma 1.9 The *Sylvester equation* (1.5) has a unique solution for any F if and only if

$$\Lambda(C) \cap \Lambda(D) = \emptyset .$$

In order to solve the AREs, it is important to understand the methods for solving the *Sylvester equations*. Bartels and Stewart [3] gave the following method

Algorithm 1.10 (B-S Algorithm) [3]

- (1) Transform C to upper-Schur form matrix \tilde{C} and D to lower-Schur form matrix \tilde{D} by using the Schur decomposition: $\tilde{C} = U^T C U$, $\tilde{D} = V^T D V$;
- (2) Solve the transformed system: $\tilde{C}\tilde{X} - \tilde{X}\tilde{D} = U^T F V$;
- (3) Transform back to the solution: $X = U \tilde{X} V^T$.

The operation count of the B-S algorithm is $10(m^3+n^3) + \frac{5}{2}(m^2n+mn^2)$ flops and the storage requirements are $2(m^2+n^2+mn)$. If C and D are already in required Schur form, then the cost of the B-S algorithm is only $O(m^2n+mn^2)$.

Golub, Nash and vanLoan [18] developed another method for solving this equation. In their method, C is only required to be transformed to upper Hessenberg form instead of upper Schur form. The operation count of their algorithm is $\frac{5}{3}m^3 + 5m^2n + \frac{5}{2}mn^2 + 10n^3$ flops, but an extra m^2 storage locations are required.

We will only use the *B-S algorithm* in this thesis, since the operation counts for both methods are $O(m^3+n^3)$ and the storage requirements are $O(m^2+n^2)$.

Definition 1.11 Let T be a linear operator $T: X \rightarrow CX - XD$, where $C \in \mathbb{R}^{m \times m}$, $D \in \mathbb{R}^{n \times n}$, and $X \in \mathbb{R}^{m \times n}$. The *norm* of the operator T is defined by

$$\|T\| = \max_{X \neq 0} \frac{\|TX\|}{\|X\|}. \quad (1.6)$$

The *separation* of the matrices C and D is defined by

$$\text{sep}(C, D) = \begin{cases} \|T^{-1}\|^{-1} = \min_{X \neq 0} \frac{\|TX\|}{\|X\|} & \lambda(C) \cap \lambda(D) = \emptyset, \\ 0 & \text{otherwise.} \end{cases} \quad (1.7)$$

Property 1.12 [33] Let $C \in \mathbb{R}^{m \times m}$ and $D \in \mathbb{R}^{n \times n}$. The separation $\text{sep}(C, D)$ has the following properties:

- (1) $\text{sep}(C, D) = \text{sep}(D, C)$;
- (2) $\text{sep}(C, D) \leq \min \left\{ |\lambda - \mu| \mid \lambda \in \lambda(C), \mu \in \lambda(D) \right\}$;
- (3) If $M \in \mathbb{R}^{m \times m}$ and $N \in \mathbb{R}^{n \times n}$, then

$$\text{sep}(C+M, D+N) \geq \text{sep}(C, D) - \|M\| - \|N\|.$$

If the Frobenius norm is used in (1.5) and (1.6), one can show that $\|T\|$ and $\text{sep}(C, D)$ are the largest and the smallest singular values of the $mn \times mn$ matrix $I_n \otimes C - D^T \otimes I_m$, respectively.

Definition 1.13

- (1) $\{\alpha_n\}$ is said to *converge* to α if $\lim_{n \rightarrow \infty} \|\alpha_n - \alpha\| = 0$;
- (2) $\{\alpha_n\}$ is said to be *q-linearly convergent* to α if there exist $c \in [0, 1)$ and an integer $k \geq 0$ such that for all $k \geq \hat{k}$, $\|\alpha_{k+1} - \alpha\| \leq c \|\alpha_k - \alpha\|$;
- (3) $\{\alpha_n\}$ is called *q-superlinearly convergent* to α if there exist $\{c_k\} \rightarrow 0$ such that $\|\alpha_{k+1} - \alpha\| \leq c_k \|\alpha_k - \alpha\|$;

- (4) $\{ \alpha_n \}$ is said to converge to α with *q-order at least p* if there exist constants $p > 1$, $c > 0$ and $\hat{k} \geq 0$ such that for all $k \geq \hat{k}$, $\| \alpha_{k+1} - \alpha \| \leq c \| \alpha_k - \alpha \|^p$; if $p=2$, the convergence is said to be q-quadratic.

Definition 1.14 [27] Let Φ and Ψ be linear subspaces of \mathbb{R}^n . The *angle* between these two subspaces is defined by

$$\cos \angle (\Phi, \Psi) = \max \left\{ \max_{\mathbf{u} \in \Phi} \min_{\mathbf{v} \in \Psi} \left\{ |\mathbf{u}^T \mathbf{v}| \right\}, \max_{\mathbf{v} \in \Psi} \min_{\mathbf{u} \in \Phi} \left\{ |\mathbf{u}^T \mathbf{v}| \right\} \right\}$$

where \mathbf{u} and \mathbf{v} are unit vectors.

Let $\theta \in [0, \frac{\pi}{2}]$ such that

$$\cos \theta = \max_{\mathbf{u} \in \Phi} \max_{\mathbf{v} \in \Psi} \left\{ |\mathbf{u}^T \mathbf{v}| \mid \|\mathbf{u}\| = 1, \|\mathbf{v}\| = 1 \right\}.$$

Then the *gap* between two linear subspaces Φ and Ψ is defined by

$$\text{gap} (\Phi, \Psi) = \sin \theta$$

and the *distance* between two linear subspaces Φ and Ψ is defined by

$$\text{dist} (\Phi, \Psi) = \max_{\mathbf{u} \in \Phi} \left\{ \text{gap} (\text{span}(\mathbf{u}), \Psi) \right\}.$$

CHAPTER 2

APPLICATIONS OF SOLVING ALGEBRAIC RICCATI EQUATIONS

Solving algebraic Riccati equations has been used in many fields in the past 20 years and plays an important role in engineering and scientific research. We will discuss some of its applications in this chapter. These applications range from (1) solving ill-conditioned eigenproblems (refining invariant subspaces), (2) solving optimal control problems for both continuous-time and discrete-time cases (computing optimal control functions), and (3) decoupling boundary value problems for ordinary differential equations.

2.1 Computing Invariant Subspaces

Definition 2.1 Let $A \in \mathbf{R}^{l \times l}$ and $X \subseteq \mathbf{R}^l$. If

$$\mathbf{x} \in X \implies A \mathbf{x} \in X$$

then X is called an invariant subspace of A .

It is well-known that for ill-conditioned eigenproblems (some eigenvalues are close together or some eigenvectors are almost parallel), it is more stable to compute the invariant subspaces with respect to the eigenvalues which are close together or eigenvectors which are almost parallel rather than compute each individual eigenvalue and eigenvector. Computing invariant subspaces can be done by solving AREs. In this section, we will discuss several methods for refining invariant subspaces. These methods are: (1) method [S] developed by Stewart [33], (2) method [DMW] by Dongarra, Moler and Wilkinson [13], and (3) method [C] by Chatelin [6]. Demmel [8] shows that all of these methods are essentially solving the ARE (1.1) under certain transformations. Also in this section, we will discuss some results about the *angle* and the *gap* between invariant subspaces.

Before discussing the methods of computing invariant subspaces, let us review the following important theorems.

Theorem 2.2 Let

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \in \mathbf{R}^{l \times l} \quad (2.1)$$

where $l=m+n$, $A_{11} \in \mathbf{R}^{n \times n}$, $A_{22} \in \mathbf{R}^{m \times m}$, $A_{21} \in \mathbf{R}^{m \times n}$ and $A_{12} \in \mathbf{R}^{n \times m}$, and suppose that the nonsingular matrix

$$T = \begin{bmatrix} T_1 & T_2 \end{bmatrix} \in \mathbf{R}^{l \times l}$$

satisfies

$$T^{-1} A T = \begin{bmatrix} \tilde{A}_{11} & \tilde{A}_{12} \\ \tilde{A}_{21} & \tilde{A}_{22} \end{bmatrix} \quad (2.2)$$

where $T_1 \in \mathbf{R}^{l \times n}$ and $T_2 \in \mathbf{R}^{l \times m}$. Then the columns of T_1 span an invariant subspace of A if and only if $\tilde{A}_{21} = 0$.

Theorem 2.3 Let $A \in \mathbf{R}^{l \times l}$ and $T_1 \in \mathbf{R}^{l \times n}$. Then the columns of T_1 span an invariant subspace of A if and only if there exists $B \in \mathbf{R}^{n \times n}$ such that

$$A T_1 = T_1 B. \quad (2.3)$$

The above two theorems provide some basic ideas for computing invariant subspaces. We can either use similarity transformations to obtain (2.2) or solve (2.3). If T is chosen to be the Riccati transformation in (2.2), i.e.

$$T = \begin{bmatrix} I & 0 \\ X & I \end{bmatrix}, \quad T^{-1} = \begin{bmatrix} I & 0 \\ -X & I \end{bmatrix}$$

we obtain

$$\begin{aligned}
 \begin{bmatrix} \tilde{A}_{11} & \tilde{A}_{12} \\ \tilde{A}_{21} & \tilde{A}_{22} \end{bmatrix} &= \begin{bmatrix} I & 0 \\ -X & I \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} I & 0 \\ X & I \end{bmatrix} \\
 &= \begin{bmatrix} A_{11} + A_{12}X & A_{12} \\ A_{21} + A_{22}X - XA_{11} - XA_{22}X & A_{22} - XA_{12} \end{bmatrix} \\
 &= \begin{bmatrix} A_{11} + A_{12}X & A_{12} \\ F(X) & A_{22} - XA_{12} \end{bmatrix}. \tag{2.4}
 \end{aligned}$$

According to theorem 2.2, the columns of the matrix $\begin{bmatrix} I \\ X \end{bmatrix}$ span an invariant subspace of A if and only if $\tilde{A}_{21} = 0$ which leads to the ARE (1.1). Moreover, if A is a *Hamiltonian* matrix¹

$$A = \begin{bmatrix} \hat{A}_{11} & -\hat{A}_{12} \\ -\hat{A}_{21} & -\hat{A}_{11}^T \end{bmatrix}, \tag{2.5}$$

then, (2.4) gives the *symmetric* ARE (1.2).

2.1.1 Methods of Computing Invariant Subspaces

Now let us review the following three methods for refining invariant subspaces.

(1) Method [S]

The method [S] was proposed by Stewart [34] who originally considered perturbation theory for the invariant subspaces. In this method, Stewart assumed that $R(T_1)$ be an approximate invariant subspace and \tilde{A}_{21} be merely small instead of zero in (2.2). By using the transformation

$$\tilde{T} := \begin{bmatrix} \tilde{T}_1 & \tilde{T}_2 \end{bmatrix} = T U. \tag{2.6}$$

¹ A *Hamiltonian* matrix will be defined in chapter 3.

where the orthogonal matrix

$$U = \begin{bmatrix} I_n & -X^T \\ X & I_m \end{bmatrix} \begin{bmatrix} (I + X^T X)^{-\frac{1}{2}} & 0 \\ 0 & (I + X X^T)^{-\frac{1}{2}} \end{bmatrix}, \quad (2.7)$$

and letting $\tilde{A}_{21} = 0$, he obtained the invariant subspace $R(\tilde{T}_1)$ of A . It is easy to show that X is a solution of the ARE (1.1).

(2) Method [DMW]

The method [DMW] was suggested by Dongarra, Moler and Wilkinson [13] who attempted to solve (2.3) by using Gaussian elimination with partial pivoting where n rows of T_1 were fixed. Demmel [8] shows that this method is also equivalent to solving the ARE (1.1) if T_1 is chosen by

$$T_1 = \begin{bmatrix} I \\ X \end{bmatrix}. \quad (2.8)$$

(3) Method [C]

The method [C] was proposed by Chatelin [6] who considered (2.3) in the form

$$A T_1 = T_1 B, \quad S^T T_1 = I_n \quad (2.9)$$

where $S^T \in \mathbb{R}^{n \times l}$. Clearly, (2.9) is equivalent to the *generalized decoupling equation*

$$A T_1 - T_1 (S^T A T_1) = 0. \quad (2.10)$$

Demmel [8] shows that if T_1 is given by (2.8) and $S^T = \begin{bmatrix} I_n & 0 \end{bmatrix}$, then (2.10) will become the ARE (1.1).

Remark 2.4 [12] By taking $S = T_1$, one can easily obtain the *orthogonal iteration method*

$$A X_{k-1} = Z_k, \quad A_k = Q_k R_k, \quad X_k = Q_k \quad k = 1, 2, \dots,$$

or the *inverse orthogonal iteration* method

$$A Z_k = X_{k-1} \text{ (find } Z_k), \quad Z_k = Q_k R_k \text{ (QR-factorization)} \quad k = 1, 2, \dots$$

2.1.2 The Gap between Invariant Subspaces

The *angle* and *gap* between two invariant subspaces have been discussed by many authors, and some important results have been obtained. Let us suppose that X is an invariant subspace of a matrix $A \in \mathbb{R}^{n \times n}$ and let \bar{X} be its approximation generated by a certain method. What can we say about $\angle(X, \bar{X})$ and *gap* (X, \bar{X}) ? Davis and Kahan [7] discussed this problem for the self-adjoint matrix A ($A = A^H$) and obtained the following result.

Theorem 2.5 [7] Let λ_i be an eigenvalue of the $n \times n$ real symmetric matrix A , and α_i be an approximation of λ_i . If

$$\delta = \min_{1 \leq i \leq p < j \leq n} \left\{ |\alpha_j - \lambda_i| \right\}$$

then

$$\delta \mid \sin \angle(X, \bar{X}) \mid \leq \|A\bar{X} - \bar{X}(\bar{X}^T A \bar{X})\|. \quad (2.11)$$

Note: The right-hand side of (2.11) is the residual of the generalized decoupling equation (2.10).

But when A is a nonsymmetric matrix, this problem becomes much more difficult. Kahan, Parlett and Jiang [22] give an example which shows that (2.11) may not hold for some nonsymmetric A . Stewart [33] uses the transformation (2.6) and (2.7) to approximate the invariant subspace by using his iterative method for solving the ARE (1.1)²; He also proves the convergence

² Stewart's method for solving AREs will be discussed in chapter 3.

property of his method ³.

2.2 Optimal Control Problems

Riccati equations have been widely used for solving optimal control problems. We will discuss its applications for seeking an optimal control function for both continuous-time and discrete-time problems, and for solving singular perturbation optimal control problems.

2.2.1 Continuous-time Optimal Control Problems [26]

Consider the linear-quadratic (Gaussian) optimal control problem

$$\dot{\mathbf{x}}(t) = \frac{d}{dt} \mathbf{x}(t) = A \mathbf{x}(t) + B \mathbf{u}(t) \quad , \quad \mathbf{x}(0) = \mathbf{x}_0 \quad . \quad (2.12)$$

We seek the solution as the optimal control function

$$\mathbf{u}(t) = H \mathbf{x}(t)$$

which minimizes the quadratic constraint

$$J(\mathbf{u}) = \int_0^{\infty} [\mathbf{x}^T(t) K \mathbf{x}(t) + \mathbf{u}^T(t) R \mathbf{u}(t)] dt$$

where $K = C^T C \in \mathbf{R}^{n \times n}$ and $R \in \mathbf{R}^{n \times n}$ are positive semi-definite matrices. Assume that the pair (A, B) is stabilizable ⁴, and the pair (C, A) is detectable ⁵. Then

$$\mathbf{u}(t) = -R^{-1} B X \mathbf{x}(t)$$

is the optimal control function where X is the unique positive semi-definite solution of the

³ See theorem 3.9 in chapter 3.

⁴ (A, B) is called to be stabilizable if there exists a constant matrix $M \in \mathbf{R}^{n \times n}$ such that all the real parts of the eigenvalues of $A - BM$ are negative.

⁵ (C, A) is called detectable if (A^T, C^T) is stabilizable.

continuous-time ARE

$$-X (B R^{-1} B^T) X + X A + A^T X + K = 0 \quad (2.13)$$

Clearly, (2.13) is a *symmetric* ARE.

2.2.2 Discrete-time Control Problems [26]

Consider the linear-quadratic discrete-time optimal control problem

$$\mathbf{x}_{k+1} = A \mathbf{x}_k + B \mathbf{u}_k \quad (2.14)$$

where $A \in \mathbf{R}^{n \times n}$, $B \in \mathbf{R}^{n \times m}$, $\mathbf{x}_k \in \mathbf{R}^n$ and $\mathbf{u}_k \in \mathbf{R}^m$, and the associated performance index

$$J_i = \frac{1}{2} \mathbf{x}_N^T X_N \mathbf{x}_N + \frac{1}{2} \sum_{k=i}^{N-1} (\mathbf{x}_k^T Q \mathbf{x}_k + \mathbf{u}_k^T R \mathbf{u}_k) \quad (2.15)$$

where $Q, X_N \in \mathbf{R}^{n \times n}$, $R \in \mathbf{R}^{m \times m}$, and Q, R are positive semi-definite matrices.

We will seek the solution of control sequences $\mathbf{u}_i, \mathbf{u}_{i+1}, \dots, \mathbf{u}_{N-1}$ to minimize the quadratic functions J_i . Substituting

$$\mathbf{u}_k = -R^{-1} B^T \mathbf{z}_{k+1}$$

into (2.14) and (2.15), we obtain the *Hamiltonian system*

$$\begin{bmatrix} \mathbf{x}_{k+1} \\ \mathbf{z}_k \end{bmatrix} = \begin{bmatrix} A & -BR^{-1}B^T \\ Q & A^T \end{bmatrix} \begin{bmatrix} \mathbf{x}_k \\ \mathbf{z}_{k+1} \end{bmatrix} \quad (2.16a)$$

or

$$\begin{bmatrix} I & BR^{-1}B^T \\ 0 & A^T \end{bmatrix} \begin{bmatrix} \mathbf{x}_{k+1} \\ \mathbf{z}_{k+1} \end{bmatrix} = \begin{bmatrix} A & 0 \\ -Q & I \end{bmatrix} \begin{bmatrix} \mathbf{x}_k \\ \mathbf{z}_k \end{bmatrix} \quad (2.16b)$$

It can be shown that the solution of this system is

$$\mathbf{u}_k = -K_k \mathbf{x}_k \quad , \quad \mathbf{x}_{k+1} = (A - B K_k) \mathbf{x}_k$$

where

$$K_k = (B^T X_{k+1} B + R)^{-1} B^T X_{k+1} A$$

$$X_k = A^T [X_{k+1} - X_{k+1} B (B^T X_{k+1} B + R)^{-1} B^T X_{k+1}] A + Q \quad .$$

If the system is steady, that is, $\lim_{k \rightarrow \infty} X_k = X$, we obtain the *discrete-time ARE*

$$X = A^T X A - A^T X B (B^T X B + R)^{-1} B^T X A + Q \quad (2.17a)$$

or

$$A^T X A - X - A^T X B (B^T X B + R)^{-1} B^T X A + Q = 0 \quad . \quad (2.17b)$$

The unique positive semi-definite solution X of (2.17) will give the *optimal control function* \mathbf{u}_k .

2.2.3 Time-optimal Control Systems with Slow and Fast Modes [23]

Consider the singular perturbation system

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{z}, \mathbf{u}, \varepsilon, t) \quad (2.18a)$$

$$\varepsilon \dot{\mathbf{z}} = \mathbf{g}(\mathbf{x}, \mathbf{z}, \mathbf{u}, \varepsilon, t) \quad (2.18b)$$

where $\mathbf{u} = \mathbf{u}(t)$ is a control input function, $0 < \varepsilon \ll 1$ and $t \in [t_0, T]$. The initial conditions are

$$\mathbf{x}(t_0) = \mathbf{x}_0 \quad , \quad \mathbf{z}(t_0) = \mathbf{z}_0 \quad . \quad (2.18c)$$

By using a two-time singular perturbation method (slow-time scale t and fast-time scale $\tau = \frac{t-t_0}{\varepsilon}$), we obtain the solution of this system

$$\mathbf{x} = \bar{\mathbf{x}}(t) + O(\epsilon)$$

$$\mathbf{z} = \bar{\mathbf{z}}(t) + \hat{\mathbf{z}}(\tau) - \mathbf{z}(t_0) + O(\epsilon)$$

where $\bar{\mathbf{x}}$ and $\bar{\mathbf{z}}$ satisfy

$$\dot{\bar{\mathbf{x}}} = \mathbf{f}(\bar{\mathbf{x}}, \bar{\mathbf{z}}, \bar{\mathbf{u}}, 0, t)$$

$$\mathbf{g}(\bar{\mathbf{x}}, \bar{\mathbf{z}}, \bar{\mathbf{u}}, 0, t) = 0$$

and $\hat{\mathbf{z}}$ is the solution of

$$\frac{d\hat{\mathbf{z}}}{d\tau} = \mathbf{g}(\mathbf{x}(t_0), \hat{\mathbf{z}}(\tau), \mathbf{u}, 0, t_0) \quad , \quad \hat{\mathbf{z}}(0) = \mathbf{z}(t_0) .$$

Now we specify our system (2.18) to be a linear system

$$\dot{\mathbf{x}} = A \mathbf{x} + B \mathbf{z} \quad (2.19a)$$

$$\epsilon \dot{\mathbf{z}} = C \mathbf{x} + D \mathbf{z} \quad (2.19b)$$

and the initial conditions remain unchanged. Introducing the fast variable

$$\boldsymbol{\eta}(\tau) = \mathbf{z} - D^{-1} C \mathbf{x} \quad , \quad \tau = \frac{t-t_0}{\epsilon} \quad ,$$

we get the solution

$$\mathbf{x} = \bar{\mathbf{x}}(t) + O(\epsilon)$$

$$\mathbf{z} = \bar{\mathbf{z}}(t) + \boldsymbol{\eta}(\tau) + O(\epsilon)$$

where $\bar{\mathbf{x}}, \bar{\mathbf{z}}$ satisfy

$$\dot{\bar{\mathbf{x}}} = A \bar{\mathbf{x}} + B \bar{\mathbf{z}}$$

$$C \bar{\mathbf{x}} + D \bar{\mathbf{z}} = 0$$

and $\boldsymbol{\eta}$ satisfies

$$\frac{d\eta}{d\tau} = D \eta , \quad \eta(0) = z_0 + D^{-1} C x_0 .$$

Therefore our solution is

$$z = -D^{-1} C e^{(A-BD^{-1}C)t} x_0 + e^{\frac{D t}{\epsilon}} (z_0 + D^{-1} C x_0) + O(\epsilon) . \quad (2.20)$$

Let us focus on the fast variable $\eta(\tau)$. Substituting $\eta = z - X x$ into (2.19), our system becomes

$$\dot{x} = (A + BX) x + D \eta \quad (2.21a)$$

$$\epsilon \dot{\eta} = (C + DX - \epsilon XA - \epsilon XB X) x + (D - \epsilon XB) \eta . \quad (2.21b)$$

Now let X be a solution of the ARE

$$C + D X - \epsilon X A - \epsilon X B X = 0 , \quad (2.22)$$

and introduce another variable $\xi = x - Y \eta$. We obtain

$$\dot{\xi} = (A + BX) \xi + [\epsilon(A + BX)Y - Y(D - \epsilon XB) + B] \eta \quad (2.23a)$$

$$\epsilon \dot{\eta} = (D - \epsilon XB) \eta . \quad (2.23b)$$

By solving the Sylvester equation

$$\epsilon (A + BX) Y - Y (D - \epsilon XB) = -B \quad (2.24)$$

for Y , our system (2.19) will be decoupled into two systems: the fast-time system

$$\epsilon \dot{\eta} = (D - \epsilon XB) \eta , \quad \eta(0) = z_0 - X x_0 \quad (2.25)$$

and the slow-time system

$$\dot{\xi} = (A + BX) \xi , \quad \xi(t_0) = x_0 - Y \eta(0) . \quad (2.26)$$

2.3 Decoupling Boundary Value Problems [2]

Let us consider the *boundary value problem (BVP) for ordinary differential equations*

$$\frac{dx}{dt} = A(t)x(t) \quad 0 < t < 1 \quad (2.27a)$$

$$\begin{bmatrix} B_1 \\ 0 \end{bmatrix} x(0) + \begin{bmatrix} 0 \\ B_2 \end{bmatrix} x(1) = \beta \quad (2.27b)$$

where $A(t) = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$ and $A_{11} \in \mathbb{R}^{n \times n}$, $A_{22} \in \mathbb{R}^{m \times m}$, $(l = m+n)$, $x(t) \in \mathbb{R}^l$,
 $B_1 \in \mathbb{R}^{m \times l}$, $B_2 \in \mathbb{R}^{n \times l}$.

Definition 2.7 A nonsingular matrix $X \in \mathbb{R}^{l \times l}$ is called a fundamental solution of the differential equation (2.27) if X satisfies

$$\frac{dX}{dt} = AX.$$

Definition 2.8 Suppose X is a fundamental solution of (2.27) which satisfies

$$Q = \begin{bmatrix} B_1 \\ 0 \end{bmatrix} X(0) + \begin{bmatrix} 0 \\ B_2 \end{bmatrix} X(1) = I, \quad P = Q^{-1} \begin{bmatrix} B_1 \\ 0 \end{bmatrix} X(0) = \begin{bmatrix} I_n & 0 \\ 0 & 0 \end{bmatrix}.$$

If there exist constants K and λ, μ such that

$$\|X(t)P X^{-1}(t)\| \leq Ke^{-\lambda(t-s)}, \quad 0 \leq s \leq t \leq 1 \quad (2.28a)$$

$$\|X(t)(I-P) X^{-1}(t)\| \leq Ke^{-\mu(s-t)}, \quad 0 \leq t \leq s \leq 1 \quad (2.28b)$$

then K is called a conditioning constant for the BVP (2.27). The BVP is called well-conditioned if K is moderate, and the dichotomic structure for the fundamental solution matrix X is given by (2.28).

The solution space (column space of the fundamental solution matrix) can be written as a direct sum of two subspaces, the nonincreasing subspace $S_1 = \{ \phi = X(t) P c , c \in \mathbb{R}^n \}$ and the nondecreasing subspace $S_2 = \{ \phi = X(t) (I-P) c , c \in \mathbb{R}^n \}$. Our purpose is to decouple the solution space into two subspaces $S = S_1 \oplus S_2$, or to decouple the variables into two groups, the *nonincreasing variables* and the *nondecreasing variables*. To do this, we introduce the new variable $y = T^{-1}x$ where the matrix $T(t) \in \mathbb{R}^{l \times l}$ and the new fundamental solution Y has the structure $Y(t) = \begin{bmatrix} Y_{11} & Y_{12} \\ 0 & Y_{22} \end{bmatrix}$ such that Y_{11} and Y_{22} characterize the nonincreasing and nondecreasing dynamics of the original problems. If there exists $T(t)$ such that

$$\frac{dy}{dt} = \bar{A}(t)y , \quad \bar{A}(t) = \begin{bmatrix} \bar{A}_{11} & \bar{A}_{12} \\ 0 & \bar{A}_{22} \end{bmatrix} ,$$

and moreover, if, say, for every $\lambda_1 \in \Lambda(\bar{A}_{11})$ and $\lambda_2 \in \Lambda(\bar{A}_{22})$, $\lambda_1 > \lambda_2$ holds and they are well-separated, then the above form for $Y(t)$ becomes possible and the matrices A , T , \bar{A} have the relation

$$\bar{A} = T^{-1} A T - T^{-1} \frac{dT}{dt} . \quad (2.29)$$

The equation (2.29) is called a *decoupling equation* of the BVP (2.27). Assume that

$$T(t) = [T_1, T_2] , \quad T^{-1}(t) = \begin{bmatrix} S_1^T \\ S_2^T \end{bmatrix}$$

where $T_1 \in \mathbb{R}^{l \times n}$ and $S_1^T \in \mathbb{R}^{m \times l}$. It is easy to show that the decoupling equation can also be written as

$$\frac{dT_1}{dt} = A T_1 - T_1 \bar{A}_{11} , \quad \bar{A}_{11} = S_1^T \left(A T_1 - \frac{dT_1}{dt} \right) . \quad (2.30)$$

If we again use the Riccati transformation

$$T(t) = \begin{bmatrix} I_n & 0 \\ R(t) & I_m \end{bmatrix}, \quad T^{-1}(t) = \begin{bmatrix} I_n & 0 \\ -R(t) & I_m \end{bmatrix},$$

our decoupling equation (2.29) will become the *Riccati differential equation (RDE)*

$$\frac{dR}{dt} = A_{21} + A_{22}R - RA_{12} - RA_{12}R, \quad R(0) = R_0. \quad (2.31)$$

If R is a constant matrix, (2.31) is identical to the ARE (1.1).

CHAPTER 3

METHODS FOR SOLVING ALGEBRAIC RICCATI EQUATIONS

Methods of solving AREs have been studied for a long time and several successful methods have been developed in the past 20 years. These methods can be divided into two classes: *direct* methods and *iterative* methods. In this chapter, several important methods for solving AREs will be discussed as well as their implementations and the concept of conditioning of AREs..

3.1 Direct Methods for Solving Algebraic Riccati Equations

The direct methods are based upon a matrix factorizations (QR-factorization or LU-decomposition). The principle behind these methods is given in

Theorem 3.1 Let A be given in (2.1) and

$$T = \begin{bmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \end{bmatrix}. \quad (3.1)$$

If T satisfies

$$T^{-1} A T = \begin{bmatrix} \bar{A}_{11} & \bar{A}_{12} \\ 0 & \bar{A}_{22} \end{bmatrix} \quad (3.2)$$

and T_{11} is nonsingular, then $X = T_{21} T_{11}^{-1}$ is a solution of the ARE (1.1).

3.1.1 Schur Method

The Schur method was introduced by Laub [25] who considered solving linear-quadratic-Gaussian control problems for both the continuous-time and the discrete-time cases. This method

is based on applying the QR-factorization to A , i.e., T in (3.1) is chosen to be an orthogonal matrix.

(1) Continuous-time AREs

Consider the continuous-time ARE (2.13) and let

$$Z = \begin{bmatrix} A & -N \\ -K & -A^T \end{bmatrix} \in \mathbf{R}^{2n \times 2n} \quad (3.3)$$

where $N = BR^{-1}B^T$. If the orthogonal matrix $U \in \mathbf{R}^{2n \times 2n}$ satisfies

$$U^T Z U = \begin{bmatrix} S_{11} & S_{12} \\ 0 & S_{22} \end{bmatrix}$$

where $\text{Re } \Lambda(S_{11}) < 0$, $\text{Re } \Lambda(S_{22}) > 0$, then $X = U_{21}U_{11}^{-1}$ is the unique positive semi-definite solution of ARE (2.13).

Theorem 3.2 [25] With respect to the notation and assumptions above,

- (i) U_{11} is nonsingular and $X = U_{21}U_{11}^{-1}$ is a symmetric positive semi-definite solution of the ARE (2.13).
- (ii) $\Lambda(S_{11}) = \Lambda(A - NX)$ = the *closed-loop spectrum*.

(2) Discrete-time AREs

Consider the discrete-time ARE (2.17) and let

$$Z = \begin{bmatrix} A + (BR^{-1}B^T)A^{-T}Q & -(BR^{-1}B^T)A^{-T} \\ -A^{-T}Q & A^{-T} \end{bmatrix} \quad (3.4)$$

and let $U \in \mathbf{R}^{2n \times 2n}$ be an orthogonal matrix such that

$$U^T Z U = \begin{bmatrix} S_{11} & S_{12} \\ 0 & S_{22} \end{bmatrix}$$

where $|\Lambda(S_{11})| < 1$, $|\Lambda(S_{22})| > 1$. Then $X = U_{21}U_{11}^{-1}$ is the unique positive semi-definite solution of (2.17).

Theorem 3.3 [25] With respect to the notation and assumptions above,

- (i) U_{11} is nonsingular and $X = U_{21}U_{11}^{-1}$ is a symmetric positive semi-definite solution of the ARE (2.17).
- (ii) $\Lambda(S_{11}) = \Lambda(A - B(R + B^T X B)^{-1} B^T X A) =$ the *closed-loop spectrum*.

Remark 3.4 For both continuous-time AREs and discrete-time AREs, Laub uses the EISPACK subroutines BALANC, ORTHES, ORTRAN, BALBAK and Stewart's software HQR3, EXCHNG to get the $2n \times 2n$ orthogonal matrices U , and uses the LINPACK subroutines DECOMP and SOLVE to obtain $U_{21}U_{11}^{-1}$.

For both continuous-time AREs and discrete-time AREs, the cost of the Schur method is about $81n^3$ flops because the QR-algorithm is applied to a $2n \times 2n$ matrix. The storage requirements are at least a $2n \times 2n$ array.

3.1.2 Symplectic Method

The symplectic method was discussed by vanLoan [34], Bunge-Gerstner and Mehrmann [4] and Bunge-Gerstner, Mehrmann and Watkins [5] for Hamiltonian systems.

Definition 3.5 Let A be given by (2.1) with $m=n$ and

$$J = \begin{bmatrix} 0 & I_n \\ -I_n & 0 \end{bmatrix}.$$

- (1) A is called *symplectic* if $A^T J A = J$;
- (2) A is called *Hamiltonian* if $J A = (J A)^T$;
- (3) A is called *J-Hessenberg* if A_{11}, A_{21}, A_{22} are upper triangular and A_{12} is upper Hessenberg;
- (4) A is called *J-triangular* if A_{11}, A_{12}, A_{21} and A_{22} are upper triangular and A_{21} has a zero diagonal;
- (5) A is called *J-tridiagonal* if A_{11}, A_{21}, A_{22} are diagonal and A_{12} is tridiagonal.

Definition 3.6 Let $A \in \mathbb{R}^{2n \times 2n}$. A decomposition $A = SR$ is called an *SR-decomposition* if S is symplectic and R is J-triangular.

The symplectic method is particularly suitable for solving continuous-time AREs where an SR-decomposition is performed on the Hamiltonian matrix instead of using a QR-factorization. Bunse-Gerstner, Mehrmann and Watkins [5] gave a method for computing the SR-decomposition by using the Gaussian elimination method. The cost of their method is only about $66n^3$ flops. But as with the Schur method, the storage requirements are also at least a $2n \times 2n$ array.

3.2 Iterative Methods for Solving Algebraic Riccati Equations

Let us consider the Taylor expansion of the ARE (1.1) around X_i

$$\begin{aligned}
 F(X) = F(X_i) + [A_{22}(X-X_i) - (X-X_i)A_{12}X_i - X_i A_{12}(X-X_i)] \\
 + [-2(X-X_i)A_{12}(X-X_i)] = 0 .
 \end{aligned}
 \tag{3.5}$$

Several iterative methods for solving AREs can be considered as arising from appropriate approximations of (3.5).

3.2.1 Kokotovic's Linear Iterative Method

Kokotovic [23] considered solving the singular perturbation system (2.19). In order to decouple the fast and slow variables, the ARE (2.22) has to be solved. He suggests that the following linear iterative method can be used for solving the ARE (1.1):

Algorithm 3.7 (Kokotovic's Linear Iterative Method) Given $X_0 = -A_{21}A_{11}^{-1}$;

$$X_{i+1} = -A_{22}^{-1} (A_{21} - X_i A_{11} - X_i A_{12} X_i) , \quad i = 0, 1, 2, \dots \quad (3.6)$$

Kokotovic shows that the order of convergence of this method is linear and the convergence contraction constant is $O(\epsilon)$, $0 < \epsilon \ll 1$ if A_{22} is $O(\frac{1}{\epsilon})$. The cost of each iteration of this method is very cheap (only $2m^2n + mn^2$ flops) and the storage requirements are also minimal.

This method can be considered as the approximation of the Taylor expansion

$$F(X_i) + A_{22}(X_{i+1} - X_i) = 0 .$$

An alternative choice of Kokotovic's linear method is: Given $X_0 = A_{21}A_{11}^{-1}$,

$$X_{i+1} = (A_{21} + A_{22}X_i - X_i A_{12} X_i) A_{11}^{-1} , \quad i = 0, 1, 2, \dots ,$$

which is the approximation of the Taylor expansion

$$F(X_i) - (X_{i+1} - X_i)A_{11} = 0 .$$

3.2.2 Stewart's Linear Iterative Method

Stewart [33] discussed the error bound for refining the invariant subspaces and obtained the ARE (1.1). He proposes that one can use another linear iterative method to solve the ARE (1.1).

Algorithm 3.8 (Stewart's Linear Iterative Method) Given $X_0 = 0$; for $i = 0, 1, 2, \dots$, solve the Sylvester equation

$$A_{22} X_{i+1} - X_{i+1} A_{11} = -A_{21} + X_i A_{12} X_i \quad (3.7)$$

by using algorithm 1.10.

Stewart also proved

Theorem 3.9 [33] Let

$$\kappa = \frac{\|A_{12}\|_F \|A_{21}\|_F}{\text{sep}^2(A_{11}, A_{22})} \quad (3.8)$$

If $\kappa < \frac{1}{4}$, then the iteration (3.7) converges linearly to the unique solution X of the equation (1.1) inside the ball

$$\|X\|_F \leq \frac{1 - \sqrt{1 - 4\kappa}}{2\kappa} \cdot \frac{\|A_{21}\|_F}{\text{sep}(A_{11}, A_{22})} < 2 \frac{\|A_{21}\|_F}{\text{sep}(A_{11}, A_{22})} \quad (3.9)$$

and the convergence contraction constant is no more than $1 - \sqrt{1 - 4\kappa}$.

The cost of Stewart's linear iterative method is only $\frac{9}{2}m^2n + \frac{5}{2}mn^2$ flops after the first iteration, which costs $10(m^3+n^3) + \frac{9}{2}m^2n + \frac{5}{2}mn^2$ flops, since the coefficient matrices of the Sylvester equations have already been transformed to the required form for all $i > 0$. The storage requirements are $2(m+n)^2$.

It is easy to show that the approximation of the Taylor expansion (3.5)

$$F(X_i) + A_{22}(X_{i+1} - X_i) - (X_{i+1} - X_i)A_{11} = 0$$

yields Stewart's linear iterative method.

3.2.2 Newton's Method

Chatelin [6] studied Newton's method for refining the invariant subspaces. She applied Newton's method directly to the generalized decoupling equations (2.10): given $T_1^{(0)}$

$$T_1^{(k+1)} = T_1^{(k)} - [DF(T_1^{(k)})]^{-1} F(T_1^{(k)}) \quad (3.10a)$$

where the Frechet derivative

$$DF(T_1)Y = (I - T_1 S^T)A Y - Y (S^T A T_1) . \quad (3.10b)$$

By choosing $T_1 = \begin{bmatrix} I \\ X \end{bmatrix}$ and $S^T = [I_n, 0]$, Demmel [8] obtains Newton's method for solving the ARE (1.1).

Algorithm 3.10 (Newton's Method) Given $X_0 = 0$; for $i = 0, 1, 2, \dots$, do

- (1) Compute $C_i = A_{22} - X_i A_{12}$, $D_i = A_{11} + A_{12} X_i$ and $F_i = -A_{21} - X_i A_{12} X_i$;
- (2) Solve the Sylvester equation

$$C_i X_{i+1} - X_{i+1} D_i = F_i \quad (3.11)$$

for X_{i+1} .

Demmel also shows that Newton's method converges quadratically to the solution.

Theorem 3.11 [8] Let κ be given by theorem 3.9. If $\kappa < \frac{1}{12}$, then Newton's iteration (3.11) converges quadratically to the unique solution X inside the ball given by (3.9), and

$$\|X_{i+1} - X\|_F \leq \frac{\|A_{12}\|_F}{\text{sep}(A_{11}, A_{22})} \cdot \frac{3}{2} \cdot \|X_i - X\|_F^2 .$$

Each Newton iteration is very expensive because a different Sylvester equation has to be solved at each iteration; therefore, the coefficient matrices must be transformed to the required (Schur or Hessenberg) forms. The operation count for each Newton iteration is

$10(m^3+n^3) + \frac{11}{2}m^2n + \frac{7}{2}mn^2$ and the storage requirements are $2(m+n)^2 + (m^2+n^2)$. For symmetric AREs, because $C_i = -D_i^T$, only half the work and half the storage of the general case are required.

It is easy to show that Newton's method is the approximation

$$F(X_i) + A_{22}(X_{i+1}-X_i) - (X_{i+1}-X_i)A_{11} - (X_{i+1}-X_i)A_{12}X_i - X_iA_{12}(X_{i+1}-X_i) = 0$$

to the Taylor expansion (3.5).

3.2.4 Generalized Secant Method

The generalized secant method was proposed by Dieci and Russell [12] who considered the approximation of the Frechet derivative

$$\begin{aligned} DF(X_i)(X_i-X_{i-1}) &\approx F(X_i) - F(X_{i-1}) \\ &\approx A_{22}(X_i-X_{i-1}) - (X_i-X_{i-1})A_{11} - X_iA_{12}(X_i-X_{i-1}) - (X_i-X_{i-1})A_{12}X_{i-1} . \end{aligned}$$

Substituting it into the Newton iteration (3.11), we obtain

$$(A_{22}-X_iA_{12})X_{i+1} - X_{i+1}(A_{11}+A_{12}X_{i-1}) = -A_{21} - X_iA_{12}X_{i-1} . \quad (3.12a)$$

Alternatively, if our approximation is made by

$$\begin{aligned} DF(X_i)(X_i-X_{i-1}) &\approx F(X_i) - F(X_{i-1}) \\ &\approx A_{22}(X_i-X_{i-1}) - (X_i-X_{i-1})A_{11} - X_{i-1}A_{12}(X_i-X_{i-1}) - (X_i-X_{i-1})A_{12}X_i , \end{aligned}$$

we get another secant iteration

$$(A_{22}-X_{i-1}A_{12})X_{i+1} - X_{i+1}(A_{11}+A_{12}X_i) = -A_{21} - X_{i-1}A_{12}X_i . \quad (3.12b)$$

Dieci and Russell suggest that one can use the following generalized secant method to solve the ARE (1.1):

Algorithm 3.12 (Generalized Secant Method)

- (1) Given $X_0 = 0$, solve X_1 by using Stewart's linear iterative method.
- (2) For $i = 1, 3, 5, \dots$, solve the Sylvester equations

$$(A_{22} - X_{i-1}A_{12})X_{i+1} - X_{i+1}(A_{11} + A_{12}X_i) = -A_{21} - X_{i-1}A_{12}X_i, \quad (3.13a)$$

$$(A_{22} - X_{i+1}A_{12})X_{i+2} - X_{i+2}(A_{11} + A_{12}X_i) = -A_{21} - X_{i+1}A_{12}X_i. \quad (3.13b)$$

Theorem 3.13 [12] Let κ be given by theorem 3.9 and $\kappa < \frac{1}{12}$. Then the generalized secant method (3.16) converges superlinearly to a unique solution X inside the ball given by (3.9). Moreover,

$$\lim_{i \rightarrow \infty} \frac{\|X_{i+1} - X\|}{\|X_i - X\|^p} = \text{constant}$$

where $p = \frac{1+\sqrt{5}}{2}$.

The operation count for each iteration of the generalized secant method is only about half work of each Newton iteration — $10n^3 + \frac{7}{2}m^2n + \frac{7}{2}mn^2$ flops if i is an even number and $10m^3 + \frac{11}{2}m^2n + \frac{5}{2}mn^2$ flops if i is an odd number — because only one matrix needs to be transformed to the required (upper Schur or Hessenberg) form, but extra mn storage locations are required ($2(m+n)^2 + (m^2+n^2+mn)$) compared to Newton's method.

It seems that the generalized secant method is often more efficient than Newton's method although its order of convergence is smaller. Dieci and Russell use an efficiency index to analyze the efficiency of these two methods. In their results, the efficiency index of the generalized secant method is $\frac{1+\sqrt{5}}{2} \approx 1.6$ while the efficiency index of Newton's method is $\sqrt{2} \approx 1.4$.

However, this method is not suitable for solving symmetric AREs since the same amount of work per iteration is needed compared to the nonsymmetric case.

3.3 Conditioning of Algebraic Riccati Equations

The conditioning of the algebraic Riccati equations has been discussed by several authors. We have already seen earlier in this chapter that κ and $\text{sep}(A_{11}, A_{22})$ play important roles in the convergence properties for most iterative methods (Stewart's method, Newton's method, and the generalized secant method). An important result for the conditioning of the ARE (1.1) is given by Dieci [10].

Let us consider the perturbed ARE

$$(A_{21} + \varepsilon B_{21}) + (A_{22} + \varepsilon B_{22})X(\varepsilon) - X(\varepsilon)(A_{11} + \varepsilon B_{11}) - X(\varepsilon)(A_{12} + \varepsilon B_{12})X(\varepsilon) = 0 \quad (3.14)$$

where $0 < \varepsilon \ll 1$ and the initial condition

$$X(0) = X.$$

We suppose the solution of the perturbed ARE (3.14) to be

$$X(\varepsilon) = X + \varepsilon \dot{X}(0) + O(\varepsilon^2).$$

By differentiating (3.14) with respect to ε and letting $\varepsilon = 0$, we obtain

$$\phi(X) \dot{X}(0) = R$$

where

$$\phi(X) Y = (A_{22} - XA_{12}) Y - Y (A_{11} + A_{12}X)$$

$$R = -(B_{21} + B_{22}X - XB_{11} - XB_{12}X).$$

If the Sylvester operator $\phi(X)$ is nonsingular, then

$$\dot{X}(0) = \phi^{-1}(X) R .$$

Therefore, the relative error is

$$\begin{aligned} \frac{\|X(\epsilon) - X\|}{\|X\|} &= \epsilon \frac{\|\dot{X}(0)\|}{\|X\|} \\ &\leq \epsilon \frac{\|\phi^{-1}(X)\| \cdot \|R\|}{\|X\|} + O(\epsilon^2) \\ &\leq \epsilon \text{cond}(\phi(X)) \left[\frac{\|B_{21}\|}{\|X\|} + \|B_{12}\| \|X\| + \|B_{11}\| + \|B_{22}\| \right] \cdot \frac{1}{\|\phi(X)\|} \quad (3.15) \end{aligned}$$

where

$$\text{cond}(\phi(X)) = \|\phi(X)\| \cdot \|\phi^{-1}(X)\|$$

is called the *condition number* of the ARE (1.1). If the Frobenius norm is used, the condition number of the ARE (1.1) is

$$\text{cond}(\phi(X)) = \|\phi(X)\|_F \cdot \|\phi^{-1}(X)\|_F = \frac{\sigma_{\max}}{\sigma_{\min}} = \frac{\sigma_{\max}}{\text{sep}_F(A_{11} + A_{12}X, A_{22} - XA_{12})}$$

where σ_{\max} and σ_{\min} are the largest and smallest singular values of the matrix $(A_{22} - XA_{12}) \otimes I_m - I_n \otimes (A_{11} + A_{12}X)$.

It is clear that the relative error strictly depends upon the size of the condition number or $\text{sep}(A_{22} - XA_{12}, A_{11} + A_{12}X)$. If $\Lambda(A_{11} + A_{12}X)$ and $\Lambda(A_{22} - XA_{12})$ are close enough together, one can hardly solve the ARE (1.1) numerically.

3.4 Implementations

We have already introduced several important methods for solving AREs in the first two sections of this chapter, and discussed the conditioning of AREs in the last section. In this section, we will discuss their implementations in more detail.

3.4.1 Iterative Refinement Method

In order to estimate the condition number $cond(\phi(X))$ and to correct the solution for the ill-conditioned ARE (1.1), Dieci [10] uses the iterative refinement method for refining the solution of the Sylvester equation (1.5).

Algorithm 3.14 (Iterative Refinement)

- (1) Solve the Sylvester equation (1.5) in t-digit arithmetic by using algorithm 1.10;
- (2) Compute the residual matrix $R = F - (CX - XD)$ in 2t-digit arithmetic;
- (3) Solve the Sylvester equation $CX - XD = R$ in t-digit arithmetic;
- (4) Estimate the condition number

$$cond(\phi(X)) \approx \frac{1}{EPS} \cdot \frac{\|Y\|}{\|X\|}$$

where EPS is the machine precision for t-digit arithmetic.

- (5) Correct the solution $X := X + Y$.

The iterative refinement scheme is very cheap because C , D have already been transformed to the desired forms.

3.4.2 Steepest Descent Technique

It is well-known that Newton's method and the secant method converge only locally. It is very important to obtain a good initial approximation by using some global methods. Dieci, Lee and Russell [11] suggest that one can use the *steepest descent* technique to get a good initial approximation for the solution of the ARE (1.1). The framework for this method is to minimize the function

$$G(X) = \frac{1}{2} \|F(X)\|_F^2 = \sum_{i=1}^m \sum_{j=1}^n f_{ij}^2$$

by using the steepest descent method.

Algorithm 3.15 (Steepest Descent Technique) Given X_0 , for $i = 0, 1, 2, \dots$

(1) Compute the gradient

$$H_i = \nabla G(X_i) = (A_{22} - X_i A_{12})^T F(X_i) - F(X_i)(A_{11} + A_{12} X_i)^T ;$$

(2) Compute $\alpha_i > 0$ such that

$$G(X_i - \alpha_i H_i) = \min_{\alpha > 0} G(X_i - \alpha H_i) ;$$

(3) Let $X_{i+1} = X_i - \alpha_i H_i$.

The local convergence property of the steepest descent method is linear and the contraction constant is no more than $\frac{\lambda_{\max} - \lambda_{\min}}{\lambda_{\max} + \lambda_{\min}}$ where λ_{\max} and λ_{\min} are the largest and smallest eigenvalues, respectively, of the matrix $\nabla^2 G(X)$. Therefore, the convergence of the steepest descent method can be very slow if $\lambda_{\min} \ll \lambda_{\max}$.

The cost of each steepest descent iteration strongly depends on the efficiency of the line search method in step (2) where the function $G(X_i)$ must be evaluated several times. The cost of each function evaluation is $2m^2n + mn^2$ flops. For the case $m=n$, this is about $\frac{1}{11}$ of each Newton iteration.

For symmetric AREs, it is easy to show that if X_0 is symmetric, then each $H_i = H_i^T$ and $F(X_i) = [F(X_i)]^T$ is too; therefore, the cost is only about half of the nonsymmetric case.

3.4.3 Reordering Eigenvalues

It often happens that our numerical solutions of the AREs via iterative methods are not the ones we need. Can one obtain the desired solution from another known solution? It is not easy in general. Dieci [10] discussed this problem. Since some solutions are known, one usually can

easily obtain the eigenvalues from the solutions. These eigenvalues can be reordered by using EISPACK subroutines; therefore, we can obtain our desired solutions after reordering eigenvalues. To be more specific let X be a solution of the ARE (1.1) satisfying

$$Q^T T^{-1} A T Q =: S = \begin{bmatrix} S_{11} & S_{12} \\ 0 & S_{22} \end{bmatrix}.$$

Suppose we seek the solution X^* satisfying

$$\operatorname{Re}(\Lambda(A_{11} + A_{12}X^*)) > \operatorname{Re}(\Lambda(A_{22} - X^*A_{12})).$$

Let \tilde{Q} be an orthogonal matrix such that

$$\tilde{S} := \tilde{Q}^T S \tilde{Q} = \begin{bmatrix} \tilde{S}_{11} & \tilde{S}_{12} \\ 0 & \tilde{S}_{22} \end{bmatrix}$$

where $\operatorname{Re}(\Lambda(\tilde{S}_{11})) > \operatorname{Re}(\Lambda(\tilde{S}_{22}))$. It can be shown that the desired solution is

$$X^* = \tilde{H}_{21} \tilde{H}_{11}^{-1}$$

where $\tilde{H} = T Q \tilde{Q}$.

CHAPTER 4

CONJUGATE GRADIENT METHOD FOR SOLVING ARE'S

The conjugate gradient method was first introduced by Hestenes and Stiefel [20] for solving systems of linear equations, or equivalently, solving quadratic unconstrained optimization problems. Fletcher and Reeves [15], and Polak and Ribiere [30] developed conjugate gradient methods for solving generalized nonlinear optimization problems.

The conjugate gradient method has been studied by many authors and it is considered as one of the best methods for solving large sparse linear systems of equations, nonlinear systems of equations and nonlinear optimization problems because other methods need matrix factorizations where storage resources may not be available.

In this chapter, we will introduce this method for solving algebraic Riccati equations. But first of all, let us review the conjugate gradient method for solving nonlinear unconstrained optimization problems.

4.1 Conjugate Gradient Method for Solving Nonlinear Optimization Problems

Originally, the conjugate gradient method was applied to solve quadratic unconstrained optimization problems or linear systems of equations [20]. Several conjugate gradient methods have been developed for solving generalized nonlinear optimization problems (the Fletcher-Reeves method, the Polak-Ribiere method). Also, many implementation aspects have been studied, such as the line search method, the Hessian matrix update, and the preconditioning technique. We will begin our discussion from quadratic functions.

4.1.1 Quadratic Case

Consider the quadratic unconstrained optimization problem

$$\text{Min } f(\mathbf{x}), \quad f(\mathbf{x}) := \frac{1}{2} \mathbf{x}^T A \mathbf{x} - \mathbf{b}^T \mathbf{x} + c \quad (4.1)$$

where A is an $n \times n$ positive definite matrix and $\mathbf{x}, \mathbf{b} \in \mathbf{R}^n$. It is not difficult to show that solving (4.1) is equivalent to solving the linear system of equations

$$A \mathbf{x} = \mathbf{b} . \quad (4.2)$$

Algorithm 4.1 (Conjugate Gradient Method) Let $f(\mathbf{x})$ be given by (4.1). Given any initial guess $\mathbf{x}_0 \in \mathbf{R}^n$:

- (1) If $\mathbf{g}_0 = \nabla f(\mathbf{x}_0) = 0$, then the solution of (4.1) is $\mathbf{x}^* = \mathbf{x}_0$; otherwise, compute $\mathbf{d}_0 = -\mathbf{g}_0$.
- (2) For $k = 0, 1, 2, \dots, n-1$, let

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k$$

where

$$\alpha_k = \frac{\mathbf{g}_k^T \mathbf{g}_k}{\mathbf{d}_k^T A \mathbf{d}_k} \quad (4.3)$$

and

$$\mathbf{g}_{k+1} = \nabla f(\mathbf{x}_{k+1}) .$$

If $\mathbf{g}_{k+1} = 0$, then \mathbf{x}_{k+1} is the solution; otherwise, let

$$\mathbf{d}_{k+1} = -\mathbf{g}_{k+1} + \beta_k \mathbf{d}_k$$

where

$$\beta_k = \frac{\mathbf{g}_{k+1}^T \mathbf{g}_{k+1}}{\mathbf{g}_k^T \mathbf{g}_k} . \quad (4.4)$$

An advantage of the Algorithm 4.1 is that the solution can be found within n iterations. In fact, one can show that the conjugate gradient method for solving the optimization problem (4.1) is equivalent to the Gaussian elimination method for solving the linear system of equations (4.2).

Alternatively, instead of using (4.3), one can use a line search method to obtain α_k such that

$$f(\mathbf{x}_k + \alpha_k \mathbf{d}_k) = \min_{\alpha > 0} f(\mathbf{x}_k + \alpha \mathbf{d}_k) .$$

Theoretically, this method will also terminate within n iterations if all line searches are exact.

Another advantage of the conjugate gradient method is that there is no matrix factorization performed. Therefore, when the matrix A is large and sparse, we do not need to store any full matrix.

4.1.2 General Cases

For generalized nonlinear unconstrained optimization problems, we can still use the conjugate gradient algorithm 4.1 where the matrix A is replaced by the Hessian matrix $\nabla^2 f(\mathbf{x}_k)$ at each iteration, or α_k is determined by a line search. There are two important conjugate gradient methods for solving generalized nonlinear optimization problems: one is developed by Fletcher and Reeves [15], and the other is by Polak and Ribiere [30]. They are slightly different.

Algorithm 4.2 (*Generalized Conjugate Gradient Method*) Let $f(\mathbf{x}) \in C^2(D)$ for all $\mathbf{x} \in D \subseteq \mathbb{R}^n$:

(1) Given an initial approximation \mathbf{x}_0 , let $k = 0$ and $\mathbf{d}_0 = -\nabla f(\mathbf{x}_0)$.

(2) Repeat (i) — (iv)

(i) stop if $\nabla f(\mathbf{x}_k) = 0$; otherwise,

(ii) compute

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k ,$$

(iii) let

$$\mathbf{d}_{k+1} = -\nabla f(\mathbf{x}_{k+1}) + \beta_k \mathbf{d}_k ,$$

(iv) let $k := k + 1$.

In algorithm 4.2, α_k is determined either by

$$\alpha_k = \frac{[\nabla f(\mathbf{x}_k)]^T \nabla f(\mathbf{x}_k)}{\mathbf{d}_k^T \nabla^2 f(\mathbf{x}_k) \mathbf{d}_k}$$

or by a line search method

$$f(\mathbf{x}_k + \alpha_k \mathbf{d}_k) = \min_{\alpha > 0} f(\mathbf{x}_k + \alpha \mathbf{d}_k) ,$$

while β_k can be chosen by the Fletcher-Reeves conjugate gradient method

$$\beta_k = \frac{[\nabla f(\mathbf{x}_{k+1})]^T \nabla f(\mathbf{x}_{k+1})}{[\nabla f(\mathbf{x}_k)]^T \nabla f(\mathbf{x}_k)} \quad (4.5)$$

or by the Polak-Ribiere conjugate gradient method

$$\beta_k = \frac{[\nabla f(\mathbf{x}_{k+1})]^T [\nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k)]}{[\nabla f(\mathbf{x}_k)]^T \nabla f(\mathbf{x}_k)} \quad (4.6)$$

Remark 4.3 If $f(\mathbf{x})$ is a quadratic function, (4.5) and (4.6) are identical and the algorithm will terminate within n iterations when all of the line searches are exact or (4.1) is used.

Both of the above methods have been discussed and implemented by many authors. In general, since the n -step termination property will no longer hold, the convergence properties of the conjugate gradient method become a major issue. From our numerical experience, the Polak-Ribiere method seems much more efficient than the Fletcher-Reeves method in terms of convergence. But only the Fletcher-Reeves method has been shown to converge if a line search method is used.

Powell [31] discusses the convergence properties for both methods and gives a clear explanation. In his work, he shows that the Algorithm 4.2 will converge to a local stationary point when $\beta_k \geq 0$. This is always true for the Fletcher-Reeves method but may not hold for the Polak-Ribiere method. Therefore, he suggests that one can use

$$\beta_k = \max \left\{ 0, \frac{[\nabla f(\mathbf{x}_{k+1})]^T [\nabla f(\mathbf{x}_{k+1}) - \nabla f(\mathbf{x}_k)]}{[\nabla f(\mathbf{x}_k)]^T \nabla f(\mathbf{x}_k)} \right\} \quad (4.7)$$

instead of using the Polak-Ribiere method.

Remark 4.4 In (4.7), when $\beta_k > 0$, it is the Polak-Ribiere method; when $\beta_k = 0$, it is the steepest descent method.

4.1.3 Implementations for Conjugate Gradient Method

Because the conjugate gradient method has its advantages for solving large sparse problems, it has been recognized as one of the best methods for solving large sparse problems. The implementation aspects include line search methods, the Hessian matrix approximations and the convergence property improvements or preconditioning technique.

(1) Line Search Method

When a line search conjugate gradient method is used to solve a nonlinear optimization problem, the efficiency of the method is strongly dependent upon the efficiency of the line search, because this step may be very expensive. The method of line search is not only applied to the conjugate gradient method, but also to the other optimization methods such as Newton's method, the secant method and the steepest descent method.

Let us suppose that $\mathbf{p} \in \mathbb{R}^n$ is a descent direction of the function $f(\mathbf{x})$, that is, \mathbf{p} satisfies $\nabla f(\mathbf{x})^T \mathbf{p} < 0$. Then there exists $\alpha > 0$ such that, when $0 < \tau < 1$ and $\tau < \sigma < 1$, the following two conditions hold:

$$f(\mathbf{x}_{k+1}) \leq f(\mathbf{x}_k) + \tau \nabla f(\mathbf{x}_k)^T (\mathbf{x}_{k+1} - \mathbf{x}_k), \quad (4.8)$$

$$\nabla f(\mathbf{x}_{k+1})^T \mathbf{p} \geq \sigma \nabla f(\mathbf{x}_k)^T \mathbf{p}, \quad (4.9)$$

where $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha \mathbf{p}$. A small σ gives a relatively inexact line search while a bigger σ (close to 1) provides a fairly exact line search.

Most authors agree that the strategy of the line search is either the condition (4.8) or both conditions (4.8) and (4.9). The practical method for line search can be any one dimensional optimization method, such as the golden section search method, Newton's method or the secant method. However, these methods may be very expensive (some of them may be even impossible) because we need to evaluate the function and its derivative many times.

A successful line search method is the *backtracking* line search method incorporating cubic interpolations (For more detail, see Dennis and Schnabel [9]).

(2) Update Hessian Matrix

The conjugate gradient method without line search needs to evaluate the Hessian matrix at each iteration and this work may be very expensive or even impossible. Instead of this, one can use some Hessian matrix update methods to approximate the Hessian matrix. For example, the positive definite secant update (the BFGS update) method can be used, i.e.

$$H_{k+1} = H_k + \frac{y_k y_k^T}{y_k^T s_k} - \frac{H_k s_k s_k^T H_k}{s_k^T H_k s_k}$$

where $s_k = x_{k+1} - x_k$ and $y_k = g_{k+1} - g_k$. This method needs $O(n^2)$ storage locations in general.

(3) Convergence Properties and Preconditioning Technique

The convergence properties of the conjugate gradient method still remain poorly understood although n -step termination is true for quadratic functions. Our numerical examples show that this method seems linearly convergent, and works poorly sometimes. Furthermore, if a restarting technique (reset the line search direction to be the steepest descent direction every n iterations) is used, one can show that the Fletcher-Reeves conjugate gradient method is n -step q -superlinearly convergent when the line search is exact. But when n is large enough, the order of convergence is essentially only linear.

However, it is well-known that the conjugate gradient method is at least as good as the steepest descent method. In fact, just as the steepest descent method, the rate of convergence of the conjugate gradient method depends on how separate the eigenvalues of the Hessian matrix

are. When all these eigenvalues are close together, the conjugate gradient method can be surprisingly efficient. In another word, if the Hessian matrix is close to a unit matrix, this method can converge very fast to a solution.

In order to improve convergence properties, a preconditioning technique can be used. The idea of preconditioning originated in partial differential equation research, but it has been applied to solve both quadratic and nonquadratic optimization problems.

When we consider solving a linear system of equations, or a quadratic optimization problem, the Hessian matrix remains unchanged everywhere. This matrix can be modified so that all its eigenvalues are close together (or the Hessian matrix is close to a unit matrix) and the method will often be much more efficient than without using it. For the general nonlinear functions, some type of Hessian matrix approximations can be used. For example, we can use the preconditioned BFGS method

$$d_k = -\bar{g} + \frac{s_{k-1}^T \bar{g}_k y_{k-1} + y_{k-1}^T \bar{g}_k s_{k-1}}{y_{k-1}^T s_{k-1}} - \frac{s_{k-1}^T \bar{g}_k}{y_{k-1}^T s_{k-1}} \left[1 + \frac{y_{k-1}^T y_{k-1}}{y_{k-1}^T s_{k-1}} \right] s_{k-1} ,$$

where $\bar{g}_k = H_k^{-1} g_k$ and H_k is given by the BFGS formula. Since the Hessian matrices are needed, $O(n^2)$ storage locations are required in general.

4.2 Conjugate Gradient Method for Solving Algebraic Riccati Equations

We now introduce a "new" method, the *conjugate gradient* method, for solving algebraic Riccati equations.

4.2.1 Conjugate Gradient Algorithm

Let us still consider the ARE (1.1)

$$F(X) = [F_{ij}(X)] = A_{21} + A_{22}X - XA_{12} - XA_{12}X$$

and let

$$G(X) = \frac{1}{2} \|F(X)\|_F^2 = \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^n F_{ij}^2(X) . \quad (4.10)$$

Then the gradient of the function G is [11]

$$\nabla G(X) = [D F(X)]^T \cdot F(X) = (A_{22} - XA_{12})^T F(X) - F(X) (A_{11} + A_{12}X)^T . \quad (4.11)$$

Since the gradient of the function G is easy to compute from (4.11), we can use a line search conjugate gradient method to locate a local minimal point of the function G , which is usually (not always) a solution of the ARE (1.1).

Algorithm 4.5 (Conjugate Gradient Method for Solving AREs)

- (1) Give an initial guess $X^{(0)} \in \mathbb{R}^{m \times n}$, let $k = 0$
- (2) Repeat (i) — (iv)
 - (i) Stop if some stopping criterion is satisfied; otherwise, do (ii) — (iv)
 - (ii) Determine $\alpha^{(k)}$ such that

$$G(X^{(k)} + \alpha^{(k)} D^{(k)}) = \min_{\alpha > 0} \left\{ G(X^{(k)} + \alpha D^{(k)}) \right\}$$

(iii) Let $X^{(k+1)} = X^{(k)} + \alpha^{(k)} D^{(k)}$ and $H^{(k+1)} = \nabla G(X^{(k+1)})$.

(iv) Compute the new conjugate gradient direction

$$D^{(k+1)} = -H^{(k+1)} + \beta^{(k)} D^{(k)} .$$

Remark 4.6

- (1) Our stopping criterion is $\|F(X^{(k)})\|_F < \varepsilon$ or $\frac{\|X^{(k+1)} - X^{(k)}\|_F}{\|X^{(k)}\|_F} < \varepsilon$.
- (2) $\alpha^{(k)}$ is determined by a line search method.
- (3) $\beta^{(k)}$ is chosen according to Powell's suggestion (4.7)

$$\beta^{(k)} = \max \left\{ 0 , \frac{\sum_{i=1}^m \sum_{j=1}^n [H_{ij}^{(k+1)} (H_{ij}^{(k+1)} - H_{ij}^{(k)})]}{\|H^{(k)}\|_F^2} \right\} .$$

4.2.2 Line Search Algorithm

The efficiency of each conjugate gradient iteration is critically dependent upon the cost of the line search method, where the function G has to be evaluated several times. The exact line search conjugate gradient method is n -step q -superlinearly convergent, but in practice, it is impossible to do the exact line search within a finite number of iterations. Instead, we use an inexact line search method. It can be shown that an inexact line search conjugate gradient method is globally convergent (Al-Baali, [1]), and the order of the convergence is linear.

The strategy of the line search has been discussed in the last section. Although a more exact line search gives the faster convergence for the conjugate gradient method, this may not be very efficient because more function evaluations are required for a more exact line search. Therefore, our line search algorithm should do a fairly exact line search and meanwhile, keep the number of function evaluations as small as possible. Obviously, the condition (4.8) needs more function evaluations to satisfy. So our line search strategy is only the condition (4.7) rather than both conditions. One of the best choices for the line search is still the backtracking line search with cubic interpolation.

Because $\nabla G(X^{(k)})$ is already known, it is easy to do line search by minimizing the cubic approximation of the function G . If we let

$$g(\alpha) = G(X^{(k)} + \alpha D^{(k)}),$$

then

$$g(0) = G(X^{(k)}), \quad g'(0) = \sum_{i=1}^m \sum_{j=1}^n [H_{ij}^{(k)} \cdot D_{ij}^{(k)}].$$

It is reasonable to use

$$g(\alpha) = a \alpha^3 + b \alpha^2 + g'(0) \alpha + g(0)$$

as a cubic approximation of the function $g(\alpha)$, where the coefficients a , b are determined by choosing two different value of α , namely α_1 and α_2 , so

$$a = \frac{1}{\alpha_1 - \alpha_2} \left[\frac{g(\alpha_1) - g(0) - g'(0)\alpha_1}{\alpha_1^2} - \frac{g(\alpha_2) - g(0) - g'(0)\alpha_2}{\alpha_2^2} \right],$$

$$b = \frac{1}{\alpha_1 - \alpha_2} \left[\frac{\alpha_1 (g(\alpha_2) - g(0) - g'(0)\alpha_2)}{\alpha_2^2} - \frac{\alpha_2 (g(\alpha_1) - g(0) - g'(0)\alpha_1)}{\alpha_1^2} \right].$$

Therefore, the minimal point of the function $\hat{g}(\alpha)$,

$$\hat{\alpha}^{(k)} = \frac{-b + \sqrt{b^2 - 3ag'(0)}}{3a},$$

is our approximate $\alpha^{(k)}$.

In the line search algorithm suggested in [9], initially, $\alpha_1 = 1$ and $\hat{\alpha}^{(k)}$ is determined by minimizing a quadratic fit. If $\hat{\alpha}^{(k)}$ does not satisfy the condition

$$g(\hat{\alpha}^{(k)}) < g(0),$$

then successive backtracking is used: let $\alpha_2 := \alpha_1$, $\alpha_1 := \hat{\alpha}^{(k)}$, and minimize the cubic interpolant to obtain a new $\hat{\alpha}^{(k)}$. A disadvantage of this method is that the initial α_1 may be too small and, consequently, the line search may not be exact enough. This disadvantage will definitely affect the convergence of the conjugate gradient method. On the other hand, if we choose a large initial α_1 , then the procedure of locating the acceptable $\hat{\alpha}^{(k)}$ may require many function evaluations and the cost of each conjugate gradient iteration can be very expensive.

In order to overcome this disadvantage, we would rather use "foretracking" than backtracking when the above case happens, that is, we increase the value of α_1 so that we can find another $\hat{\alpha}^{(k)}$ which is more accurate than the old one. We summarize the whole procedure in the Algorithm 4.7.

Algorithm 4.7 (Line Search Algorithm)

- (1) Compute $c = \sum_{i=1}^m \sum_{j=1}^n [H_{ij}^{(k)} \cdot D_{ij}^{(k)}]$ and let $g_0 = G(X^{(k)})$;
- (2) Choose $0 < \alpha_1 < \alpha_2$ such that

$$g_1 < g_0 < g_2$$

where $g_1 = G(X^{(k)} + \alpha_1 D^{(k)})$ and $g_2 = G(X^{(k)} + \alpha_2 D^{(k)})$;

(3) Compute the cubic approximation

$$\hat{g}(\alpha) = a \alpha^3 + b \alpha^2 + c \alpha + g_0$$

where

$$\begin{bmatrix} a \\ b \end{bmatrix} = \frac{1}{\alpha_1 - \alpha_2} \begin{bmatrix} \frac{1}{\alpha_1^2} & -\frac{1}{\alpha_2^2} \\ -\frac{\alpha_2}{\alpha_1^2} & \frac{\alpha_1}{\alpha_2^2} \end{bmatrix} \begin{bmatrix} g_1 - g_0 - c \alpha_1 \\ g_2 - g_0 - c \alpha_2 \end{bmatrix} ;$$

(4) Determine $\alpha^{(k)}$: if $a = 0$, $\alpha^{(k)} = -\frac{c}{2b}$, otherwise $\alpha^{(k)} = \frac{-b + \sqrt{b^2 - 3ac}}{3a}$.

Remark 4.8

- (1) Other one dimensional minimization techniques (such as Newton's method and the secant method) can also be used to evaluate $\alpha^{(k)}$, but it would be more expensive because both $g(\alpha)$ and $g'(\alpha)$ (or its approximation) are needed at each point.
- (2) The existence of α_1 and α_2 is guaranteed because $D^{(k)}$ is a descent direction and $g(\alpha)$ is bounded below.

The cost of this line search method is strictly dependent on the number evaluations of the function G . Each function evaluation needs $m^2n + 2mn^2$ flops. When $m = n$, this is about $\frac{1}{11}$ of the cost of a Newton iteration. Therefore, reducing the number of function evaluations is the major work of our algorithm.

In practice, we use $\alpha_2 = 1$ at the beginning. If $g(\alpha_2) \leq g(0)$, then $\alpha_1 := \alpha_2$ and $\alpha_2 := 10\alpha_2$; otherwise, let $\alpha_1 = 0.1\alpha_2$. We keep doing this procedure until the requirement is met. Our numerical experience shows that the average number of function evaluations for this method is about three (this is good, because we need at least two function evaluations in each line search) and the line search is fairly exact.

4.2.3 Comparisons and Implementations

In this section, we will compare our conjugate gradient method to some other methods (Newton's method, the secant method and the Schur method) in terms of operation counts, storage requirements and convergence properties.

(1) Operation Counts

As we know, the cost of the conjugate gradient method is critically dependent upon the line search efficiency or the average number of function evaluations. Because one gradient value is required, if we save $A_{21} - XA_{11}$, the total cost of our conjugate gradient iteration is $(m^2n + 2mn^2) + k(2m^2n + mn^2)$ flops where k is the number of function evaluations. In our line search method, the average number of function evaluations is assumed to be 3 (this is from numerical experience); therefore, the cost of each iteration is $7m^2n + 5mn^2$ flops. Table 4.1 gives the comparison of the cost among Newton's method, the generalized secant method, and the conjugate gradient method.

Iteration i	Newton	Secant	C-G
$i = \text{odd}$	$10(m^3 + n^3) + \frac{11}{2}m^3n + \frac{7}{2}mn^2$	$10m^3 + \frac{11}{2}m^2n + \frac{5}{2}mn^2$	$7m^2n + 5mn^2$
$i = \text{even}$	$10(m^3 + n^3) + \frac{11}{2}m^3n + \frac{7}{2}mn^2$	$10n^3 + \frac{7}{2}m^2n + \frac{7}{2}mn^2$	$7m^2n + 5mn^2$

Particularly, when the Riccati equation is the symmetric ARE (1.2)

$$F(X) = \hat{A}_{21} + \hat{A}_{11}^T X + X\hat{A}_{11} - X\hat{A}_{12}X = 0$$

where $m = n$ and $\hat{A}_{12}, \hat{A}_{21}$ are symmetric positive semi-definite matrices, then the matrix $F(\hat{X})$ will also be symmetric if the solution \hat{X} is symmetric. Therefore, the gradient matrix $\nabla G(\hat{X})$ will be symmetric as well:

$$\nabla G(\hat{X}) = (\hat{A}_{11}^T - \hat{X}\hat{A}_{12})^T F(\hat{X}) + F(\hat{X})(\hat{A}_{11} - \hat{A}_{12}\hat{X})^T = [\nabla G(\hat{X})]^T .$$

In practice, we seek a symmetric positive semi-definite solution of the ARE (1.2). Beginning with a symmetric initial guess $\hat{X}^{(0)}$, all of our $\hat{X}^{(k)}$ will also be symmetric. Then the cost of each conjugate gradient iteration is only $(2k + 1)n^3$ flops. When $k = 3$, this is about half the work of Newton's method.

Iteration i	Newton	Secant	C-G
$i=odd$	$16.5n^3$	$19.5n^3$	$7n^3$
$i=even$	$16.5n^3$	$12n^3$	$7n^3$

In the case of $m \gg n$ (or $m \ll n$), the cost of one conjugate gradient iteration is only $O(m^2)$ (or $O(n^2)$) flops where Newton's method needs $O(m^3)$ (or $O(n^3)$) flops.

Iteration i	Newton	Secant	C-G
$i=odd$	$O(m^3)$	$O(m^3)$	$O(m^2)$
$i=even$	$O(m^3)$	$O(m^2)$	$O(m^2)$

(2) Storage Consideration

On the aspect of storage requirements, the conjugate gradient method needs only $5mn$ additional storage locations since we do not perform any matrix factorization. The comparisons of the storage requirements among Newton's method, the secant method, the Schur method and the conjugate gradient method are in table 4.4 for the nonsymmetric case and table 4.5 for the symmetric

case.

Table 4.4: Storage requirements for Solving ARE (1.1)			
Newton	Secant	Schur	C-G
$3m^2+3n^2+4mn$	$3m^2+3n^2+5mn$	$2m^2+2n^2+4mn$	m^2+n^2+7mn

Table 4.5: Storage requirements for Solving ARE (1.2)			
Newton	Secant	Schur	C-G
$7n^2$	$8n^2$	$8n^2$	$5n^2$

Also in the case of $m \gg n$, all the other methods require $O(m^2)$ extra storage locations because some matrix factorizations (QR-factorization or LU-decomposition) must be carried out. But the conjugate gradient method requires only $O(m)$ extra storage spaces (the steepest descent method can be considered as a special case of the conjugate gradient method, i.e., $\beta_k = 0$).

Table 4.6: Additional storage requirements for Solving ARE (1.1) ($m \gg n$)			
Newton	Secant	Schur	C-G
$O(m^2)$	$O(m^2)$	$O(m^2)$	$O(m)$

(3) Convergence Properties

We have already discussed the convergence properties of the conjugate gradient method for solving general nonlinear unconstrained optimization problems in Section 4.1. These properties will also hold for our conjugate gradient method for solving AREs.

Since for each iteration, we do not increase the function value $G(X)$, the global convergence property can be guaranteed for any initial value $X^{(0)}$. In fact, as we already know, the conjugate gradient method is at least as good as the steepest descent method. This suggests using the conjugate gradient method instead of the steepest descent technique to obtain a good initial approximation for Newton's method or the secant method [11].

The local convergence properties are not very clear so far. Although with a restarting procedure the n -step q -superlinear convergence property can be proved, it is almost impossible to tell the difference between it and linear convergence when n is large. Our numerical experience shows that the conjugate gradient method converges only linearly in general and it can be very slow.

(4) **Conditioning** We have already discuss the conditioning of the ARE (1.1) in Section 3.3. We know that a Sylvester equation has to be solved at each Newton iteration or each secant iteration. When the condition number $cond(\phi(X))$ is large, one can hardly solve these Sylvester equations. But when we use the conjugate gradient method to solve the ARE (1.1), the condition number will no longer be the major factor. This is because we do not solve Sylvester equations any more. Moreover, since the ARE (1.1) has been transformed to be an optimization problem, keeping the Hessian matrix to be positive definite becomes the conditioning of the conjugate gradient method for solving the ARE (1.1). In practice, since we use a line search method instead of the Hessian matrix, the function value will be decreasing after each line search and the method will always work. Numerical examples tell us that the conjugate gradient method is not suitable for solving regular well-conditioned problems because Newton's method and the secant method are usually more efficient, but the conjugate gradient method may be suitable for solving ill-conditioned problems.

(5) Large Sparse Systems

When solving large sparse systems, matrix factorizations (LU-decomposition and QR-factorization) are very expensive, also storage requirements are potentially full matrices ($O(m^2 + n^2)$) which may not be available in practice. The conjugate gradient method has the

advantage of only doing matrix additions and multiplications, where the cost is considerably less than doing the matrix factorizations and there is no full matrix storage required. This is probably the only realistic method for solving large sparse problems.

(6) Other Implementations

In practice, if the line search direction $D^{(k)}$ is not a descent direction, i.e., $\sum_{i=1}^m \sum_{j=1}^n [H_{ij}^{(k)} \cdot D_{ij}^{(k)}] > 0$, we reset the direction to be the steepest descent direction $D^{(k)} = -H^{(k)} = -\nabla G(X^{(k)})$. This is called a restarting technique.

As we have mentioned, instead of performing a line search, the Hessian matrix can be used to obtain $\alpha^{(k)}$ at each iteration. In general, $O(m^2n^2)$ storage requirements are needed. A drawback of using the Hessian matrix is that the the global convergence property may not hold because the Hessian matrix can be singular or not positive definite. Since the conjugate gradient method may only be suitable for solving large problems, if we want use the Hessian matrix or some of its approximations, we must avoid using (m^2n^2) storage locations. This aspect needs further studies. An advantage of having the Hessian matrix is that the preconditioning technique can be applied so that the speed of convergence can be improved and the method can be much more efficient.

The process of reordering is not difficult for the conjugate gradient method because the matrices $A_{11} + A_{12}X^{(k)}$ and $A_{22} - X^{(k)}A_{12}$ are easy to obtain. One Newton iteration can be performed on them so that we can do automatic reordering as before. But for large sparse systems, this may be impossible due to storage limitations.

CHAPTER 5

NUMERICAL EXAMPLES

We have already discussed several numerical methods for solving AREs, and comparisons among Newton's method, the secant method, the Schur method and the conjugate gradient method in terms of convergence, operation counts and storage requirements have been given in chapter 4. In this chapter, we will give some numerical examples and results (all of our computations are done on an IBM 3081 machine using VS FORTRAN on the MTS operating system at Simon Fraser University). Our stopping criterion is either $\|F(X_k)\|_F < \epsilon$ or $\frac{\|X_{k-1} - X_k\|_F}{\|X_k\|_F} < \epsilon$.

Examples 1-4 arise from solving control problems and all of them require solving a well-conditioned nonsymmetric ARE (1.1). We will use Newton's method, the generalized secant method, and the conjugate gradient method to solve these problems. For all the methods in these four examples, the initial guesses are $X_0 = I$.

Example 1 : Kokotovic [24]

Consider the continuous model

$$\begin{bmatrix} \dot{\mathbf{x}} \\ \dot{\mathbf{z}} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{21} \\ A_{12} & A_{22} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{z} \end{bmatrix}$$

where $A_{11} \in \mathbb{R}^{2 \times 2}$, $A_{22} \in \mathbb{R}^{3 \times 3}$, $\mathbf{x} \in \mathbb{R}^2$, $\mathbf{z} \in \mathbb{R}^3$ and

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} -0.11 & 0.02 & 0.03 & 0.0 & 0.02 \\ 0.0 & -0.17 & 0.0 & 0.0 & 0.17 \\ 0.0 & 2.0 & -4.0 & 0.0 & 0.0 \\ -4.0 & 0.0 & 0.0 & -2.0 & 0.0 \\ 0.0 & 0.0 & 0.0 & 4.75 & -5.0 \end{bmatrix}$$

All of our methods give the same solution

$$X = \begin{bmatrix} 0.04840394 & 0.5203825 \\ -2.154609 & 0.04306733 \\ -2.106174 & 0.05827145 \end{bmatrix}$$

Example 2 : Phillips [28]

Consider the discrete model

$$\begin{bmatrix} x(t+1) \\ z(t+1) \end{bmatrix} = \begin{bmatrix} A_{11} & A_{21} \\ A_{12} & A_{22} \end{bmatrix} \begin{bmatrix} x(t) \\ z(t) \end{bmatrix} \quad (5.2)$$

where $A_{11} \in \mathbb{R}^{2 \times 2}$, $A_{22} \in \mathbb{R}^{3 \times 3}$, $x \in \mathbb{R}^2$, $z \in \mathbb{R}^3$ and

$$\begin{bmatrix} A_{11} & A_{21} \\ A_{12} & A_{22} \end{bmatrix} = \begin{bmatrix} 0.9014 & 0.1179 & 0.0525 & 0.0164 & 0.02014 \\ -0.0196 & 0.8743 & 0.0 & 0.025 & 0.02934 \\ -0.0071 & 0.7342 & 0.20175 & 0.013 & 0.21067 \\ -0.75 & -0.0557 & -0.032 & 0.019357 & -0.014076 \\ -0.306 & -0.01694 & -0.011 & 0.14278 & 0.013217 \end{bmatrix}$$

The solution of the ARE (1.1) is

$$X = \begin{bmatrix} -0.0931666 & 1.140467 \\ -1.083789 & 0.1514285 \\ -0.5308208 & 0.1025373 \end{bmatrix}$$

Example 3 : Phillips [28]

Let us also consider the discrete model (5.2) where $A_{11} \in \mathbb{R}^{4 \times 4}$, $A_{22} \in \mathbb{R}^{4 \times 4}$, $x(k) \in \mathbb{R}^4$, $z(k) \in \mathbb{R}^4$ and

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} 0.835 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 \\ 0.096 & 0.861 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.029 \\ -0.002 & -0.005 & 0.882 & -0.253 & 0.041 & -0.003 & -0.025 & -0.001 \\ 0.007 & 0.014 & -0.029 & 0.928 & 0.0 & 0.006 & 0.059 & 0.002 \\ -0.03 & -0.061 & 2.028 & -2.303 & 0.088 & -0.021 & -0.224 & -0.008 \\ 0.048 & 0.758 & 0.0 & 0.0 & 0.0 & 0.165 & 0.0 & 0.023 \\ -0.012 & -0.027 & 1.209 & -1.4 & 0.161 & -0.013 & 0.156 & 0.006 \\ 0.815 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 0.011 \end{bmatrix}$$

The solution of this ARE is

$$X = \begin{bmatrix} -0.01451242 & -0.03516997 & 2.079860 & 1.813171 \\ -0.09707658 & 1.089080 & 0.0 & 0.0 \\ 0.02188941 & 0.01141115 & 2.255242 & -1.524221 \\ 0.9890777 & 0.0 & 0.0 & 0.0 \end{bmatrix}$$

Example 4 : Phillips [29]

Consider the continuous model (5.1) where $A_{11} \in \mathbb{R}^{4 \times 4}$, $A_{22} \in \mathbb{R}^{4 \times 4}$, $x \in \mathbb{R}^4$, $z \in \mathbb{R}^4$ and

$$\begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} = \begin{bmatrix} -5.0 & 0.0 & 0.0 & 0.0 & 4.75 & 0.0 & 0.0 & 0.0 \\ 0.0 & -2.0 & 0.0 & 0.0 & 0.0 & -2.0 & 0.0 & 0.0 \\ -0.08 & -0.11 & -3.99 & -0.93 & 0.0 & -0.07 & 10.0 & -9.1 \\ 0.0 & 0.0 & 1.32 & -1.39 & 0.0 & 0.0 & 0.0 & -0.28 \\ 0.0 & 0.0 & 0.0 & 0.0 & -0.2 & 0.0 & 0.0 & 0.0 \\ 0.17 & 0.0 & 0.0 & 0.0 & 0.0 & -0.17 & 0.0 & 0.0 \\ 0.0 & 0.0 & 0.2 & 0.0 & 0.0 & 0.0 & -0.5 & 0.0 \\ 0.01 & 0.01 & -0.06 & 0.12 & 0.0 & 0.01 & 0.0 & -0.11 \end{bmatrix}$$

The solution is

$$X = \begin{bmatrix} 0.0 & 0.0 & 0.0 & 0.0 \\ -0.03519669 & 0.0 & 0.0 & 0.0 \\ 0.000638565 & 0.001647461 & -0.05102651 & 0.002348209 \\ -0.001882734 & -0.004962614 & -0.01556921 & -0.009525143 \end{bmatrix}$$

We compare our numerical results in table 5.1, where $\varepsilon = 10^{-6}$.

Table 5.1 : Numerical Results for Examples 1-4										
Example	κ	Newton			Secant			C-G		
		N	Time	r	N	Time	r	N	Time	r
1	0.086	3	0.0050	0.16E-8	4	0.0047	0.52E-9	20	0.016	0.47E-6
2	0.222	3	0.0067	0.20E-7	4	0.0061	0.17E-7	10	0.0078	0.50E-6
3	1.255	3	0.016	0.45E-7	4	0.016	0.26E-6	13	0.022	0.45E-5
4	3.525	4	0.015	0.12E-8	6	0.018	0.25E-9	34	0.058	0.80E-6

Remark 5.1:

- (1) κ is the condition number given by theorem (3.9);
- (2) N indicates the number of iterations for 6-digit accuracy;
- (3) r is the residual of $F(X)$, i.e.

$$r = \|F(X_N)\|_F = \|A_{21} + A_{22}X_N - X_N A_{11} - X_N A_{12}X_N\|_F ;$$

- (4) Time is the CPU time in seconds.

The numerical results in examples 1-4 show that Newton's method and the secant method are comparable while the conjugate gradient method is less efficient for solving well-conditioned problems.

Examples 5-6 arise from solving continuous optimal control problems. As we know from chapter 2, a symmetric ARE (1.2) has to be solved. We apply the Schur method, Newton's method and the conjugate gradient method to solve these two problems (since we know that the secant method is not as good as Newton's method for solving symmetric AREs, we will not use it). For Newton's method and the conjugate gradient method, because we seek the symmetric positive semi-definite solution, we choose a symmetric positive semi-definite initial guesses,

$X_0 = 0$, for both problems.

Example 5 : Laub [25]

Let

$$\hat{A}_{11} = \begin{bmatrix} -2 & 1 & 0 & & 0 & 1 \\ 1 & -2 & 1 & 0 & & 0 \\ 0 & 1 & \ddots & & & \\ & 0 & \ddots & \ddots & & \\ & & & \ddots & \ddots & 0 \\ 0 & & & & \ddots & 1 \\ 1 & 0 & & & 0 & 1 & -2 \end{bmatrix}$$

and $\hat{A}_{21} = \hat{A}_{12} = I$.

Table 5.2 gives the numerical results for $n = 50$. In this example, the Schur method works very well; Newton's method converges to the solution which is not the desired one and one cannot expect this solution to be more accurate; the conjugate gradient method works fairly well, and it converges to the correct solution.

Example 6 : Laub [25]

Let

$$(\hat{A}_{11})_{ij} = \begin{cases} 1, & i = j + 1 \\ 0, & \text{otherwise} \end{cases}$$

and $\hat{A}_{12} = \text{diag}(1, 0, \dots, 0)$, $\hat{A}_{21} = \text{diag}(0, \dots, 0, 1)$.

The numerical results are in table 5.2 for $n = 21$. In this example, the Schur method and Newton's method fail, but the conjugate gradient method still works.

Table 5.2 : Numerical Results for Examples 5-6								
Example	Schur		Newton			C-G		
	Time	r	N	Time	r	N	Time	r
5	18.2	0.58E-10	5	35.4	0.81E-1	49	81.4	0.99E-6
6	1.62	0.45E+3	fails			365	35.3	0.18E-5

The results in table 5.2 show that: the Schur method can have difficulty because of ill-conditioning; Newton's method may not converge to the required solution when the initial guess is not close enough to the solution; the conjugate gradient method converges to a solution after many iterations. In addition, both the Schur method and Newton's method need a lot of storage locations (full matrix).

These results also show that, for Newton's method, one may not obtain a more accurate solution (see example 5) and the method may even fail to converge (example 6, Newton's method overflows) without going through some kind of descent techniques (such as a steepest descent technique) to obtain a good initial approximation. For the conjugate gradient method, the convergence may be very slow, but because of the sparsity, the function evaluations can be done in a cheaper way (not necessary to perform the matrix multiplications in $O(n^3)$ flops), such that the cost of each iteration is very cheap (for example, in our results, the cost of each conjugate gradient iteration is only about 10% to 15% of the cost of each Newton iteration).

Examples 7-8 are coming from computing invariant subspaces.

Example 7 : Golub [19]

Consider the problem of computing an invariant subspace $\begin{bmatrix} I_2 \\ X \end{bmatrix} \in \mathbb{R}^{5 \times 2}$ of the Hessenberg

matrix

$$A = \begin{bmatrix} 2 & 3 & 4 & 5 & 6 \\ 4 & 4 & 5 & 6 & 7 \\ 0 & 3 & 6 & 7 & 8 \\ 0 & 0 & 2 & 8 & 9 \\ 0 & 0 & 0 & 1 & 10 \end{bmatrix}.$$

With the initial guess $X_0 = 0$, Newton's method and the secant method converge to the solution of the ARE (1.1)

$$X = \begin{bmatrix} -0.747269 & -1.023547 \\ 0.345389 & 0.348213 \\ -0.044609 & -0.039395 \end{bmatrix},$$

while the conjugate gradient method converges to the solution

$$X = \begin{bmatrix} 0.845147 & -0.212120 \\ -0.699520 & -0.184228 \\ 0.127461 & 0.048286 \end{bmatrix}.$$

The results are in table 5.3.

Table 5.3 : Numerical Results for Example 7								
Secant			Newton			C-G		
N	Time	r	N	Time	r	N	Time	r
8	0.012	0.19E-13	7	0.016	0.47E-11	108	0.080	0.85E-6

Example 8 : Varah [35]

Also consider the $l \times l$ upper Hessenberg matrix

$$A = (a_{ij}) = \begin{cases} l + 1 - \max(i, j), & j \geq i - 1 \\ 0, & \text{otherwise} \end{cases}$$

We compute an eigenvector $\begin{bmatrix} 1 \\ X \end{bmatrix}$. Again, we use the secant method, Newton's method and the conjugate gradient method. All of our methods converge to the same solution for both $X_0 = [0, 0, \dots, 0]^T$ and $X_0 = [1, 0, \dots, 0]^T$. For example, when $l = 8$, the solution is

$$X = [0.947712, 0.532145, 0.206999, 0.057052, 0.010775, 0.001263, 0.000070]^T .$$

The results are given in table 5.4 for $l = 8, 12$, and 20 with the initial guess $X_0 = [1, 0, \dots, 0]^T$.

l	Secant			Newton			C-G		
	N	Time	r	N	Time	r	N	Time	r
8	8	0.035	0.39E-9	6	0.048	0.21E-10	21	0.020	0.43E-6
12	16	0.170	0.71E-6	6	0.117	0.30E-8	23	0.035	0.86E-6
20	10	0.430	0.92E-12	7	0.554	0.65E-8	86	0.287	0.92E-6

The numerical results of example 8 verify our claim in chapter 4, i.e., when $m \gg n$ (here $n = 1$ and $m = l - 1$), the conjugate gradient method can be more efficient.

CHAPTER 6

CONCLUSIONS

In this thesis, we have reviewed the applications and the methods for solving matrix algebraic Riccati equations. These applications are refining invariant subspaces, solving optimal control problems (computing optimal control functions for both continuous and discrete problems, solving singular perturbation control systems), and decoupling boundary value problems for ordinary differential equations. The methods we have reviewed include direct methods (the Schur method and the symplectic method), iterative methods (Kokotovic's linear iterative methods, Stewart's linear iterative method, Newton's method and the generalized secant method). We have also discussed the conditioning of AREs as well as some implementation issues (the iterative refinement method, the steepest descent technique and the eigenvalues reordering). Furthermore, we have proposed and applied the *conjugate gradient* method to solve AREs (only a line search conjugate gradient method is used in this thesis). Theoretical comparisons in chapter 4 and numerical results in chapter 5 indicate that this method is not suitable for solving regular well-conditioned problems (Newton's method, the secant method and the Schur method are usually more efficient). However, for large sparse problems, the conjugate gradient method has advantages. Since no matrix factorization is required, this method can be more efficient for solving large sparse problems and also more stable for solving ill-conditioned problems. The efficiency of the line search conjugate gradient method is critically dependent upon the efficiency of the line search algorithm, since it determines the number of function evaluations. Also, we have shown from both theory and practice that the conjugate gradient method can be more efficient if $m \gg n$.

It is well-known that for the local convergence methods (such as Newton's method and the secant method), it is very important to have a good initial approximation of the solution. Although this work can be done by using the steepest descent technique it can also be done by using the conjugate gradient method, which is at least as good as the steepest descent technique. Actually, the steepest descent technique is a special case of the conjugate gradient method

($\beta_k = 0$).

The conjugate gradient method without line search, particularly, the preconditioned conjugate gradient method, needs to be investigated further. The difficulties are how to find the Hessian matrix without storing full $mn \times mn$ matrices, and how much we can improve the method in terms of cost vs. convergence.

Another question is, whether we can use the conjugate gradient method to solve the generalized (system of) algebraic Riccati equations

$$A_{22}R - LA_{11} = -A_{21} + LA_{12}R$$

$$B_{22}R - LB_{11} = -B_{21} + LB_{12}R$$

and, if possible, how efficient it will be (Newton's method and the secant method can easily be applied to this case [11]).

Finally, improving the bound for the angle and gap between two invariant subspaces is still one of our considerations, particularly for nonsymmetric matrices.

REFERENCES

- [1] M.Al-Baali; "Descent property and global convergence of the Fletcher-Reeves method with inexact line search", *IMA J. Numer. Anal.* 5 (1985), pp. 121-124.
- [2] U.M.Ascher, R.M.M.Mattheij, R.D.Russell; *Numerical Solution of Boundary Value Problems for Ordinary Differential Equations*, Prentice-Hall (1988).
- [3] R.H.Bartels, G.W.Stewart; "Solution of the matrix equation $AX + XB = C$ ", *CACM* 15 (1972), pp. 820-826.
- [4] A.Bunse-Gerstner, V.Mehrmann; "A symplectic QR-like algorithm for the solution of the real algebraic Riccati equation", *IEEE Trans. Auto. Control* AC-31 (1986), pp. 1104-1113.
- [5] A.Bunse-Gerstner, V.Mehrmann, D.Watkins; "An SR algorithm for Hamiltonian matrices based on Gaussian elimination", manuscript (1987).
- [6] F.Chatelin; "Simultaneous Newton's iteration for the eigenproblem", *Computing Suppl.* 5 (1984), pp. 67-74.
- [7] C.Davis, W.Kahan; "The rotation of eigenvectors by a perturbation, III", *SIAM J. Numer. Anal.*, 7 (1970), pp. 1-46.
- [8] J.W.Demmel; "Three methods for refining estimates of invariant subspaces", *Computing* 38 (1987), pp. 43-57.
- [9] J.E.Dennis, R.B.Schnabel; *Numerical Methods for Unconstrained Optimization and Non-linear Equations*, Prentice-Hall (1983).
- [10] L.Dieci; "Some numerical considerations and a new approach for the solution of algebraic Riccati equations", manuscript (1988).
- [11] L.Dieci, Y.M.Lee, R.D.Russell; "Iterative methods for solving algebraic Riccati equations" manuscript (1987).
- [12] L.Dieci, R.D.Russell; "On the computation of invariant subspaces", manuscript (1987).

- [13] J.J.Dongarra, C.B.Moler, J.H.Wilkinson; "Improving the accuracy of computed eigenvalues and eigenvectors", *SIAM J. Numer. Anal.* 20 (1983), pp. 23-45.
- [14] R.Fletcher; *Practical Methods of Optimization*, second edition, John Wiley, (1987).
- [15] R.Fletcher, C.M.Reeves; "Function minimization by conjugate gradients", *Comput. J.*, 7 (1964), pp. 149-154.
- [16] B.S.Garbow, J.M.Boyle, J.J.Dongarra, C.B.Moler; *Matrix Eigensystem Routines: EISPACK Guide Extension*, Springer-Verlag (1977).
- [17] P.E.Gill, W.Murray, M.H.Wright; *Practical Optimization*, Academic Press (1981).
- [18] G.H.Golub, S.Nash, C.F.vanLoan; "A Hessenberg-Schur method for the problem $AX + XB = C$ ", *IEEE Trans. Auto. Control*, AC-24 (1979), pp. 909-913.
- [19] G.H.Golub, C.F.vanLoan; *Matrix Computations*, The Johns Hopkins Univ. Press (1983).
- [20] M.R.Hestenes, E.L.Stiefel; "Methods of conjugate gradients for solving linear systems", *J. Res. Nat. Bur. Standards, Section B*, 49 (1952), pp. 409-436.
- [21] M.R.Hestenes; *Conjugate-Direction Methods in Optimization*, Springer-Verlag, (1980).
- [22] W.Kahan, B.N.Parlett, E.Jiang; "Residual bounds on approximate eigensystems of nonnormal matrices", *SIAM J. Numer. Anal.*, 19 (1982), pp.470-484.
- [23] P.V.Kokotovic; "Applications of singular perturbation techniques to control problems", *SIAM Review*, 26 (1986), pp. 501-550.
- [24] P.V.Kokotovic; "A Riccati equation for block-diagonalization of ill-conditioned systems", *IEEE Trans. Auto. Control* AC-20 (1975), pp. 812-814.
- [25] A.J.Laub; "A Schur method for solving algebraic Riccati equations", *IEEE Trans. Auto. Control* AC-24 (1979), pp. 913-921.
- [26] F.L.Lewis; *Optimal Control*, Wiley, (1986).
- [27] B.Parlett; *The Symmetric Eigenvalue Problem*, Prentice-Hall (1980).
- [28] R.G.Phillips; "Reduced order modelling and control of two-time-scale discrete systems", *Int J. Control*, 31 (1980), pp. 756-780.

- [29] R.G.Phillips; "A two-stage design of linear feedback controls", IEEE Trans. Auto. Control AC-25 (1980), pp. 1220-1223. ip [30] E.Polak; *Computational Methods in Optimization: A Unified Approach*, Academic Press (1971).
- [31] D.J.M.Powell; "Convergence properties of algorithms for nonlinear optimization", SIAM Review 28 (1986), pp. 487-500.
- [32] G.W.Stewart; "HQR3 and EXCHANG: Fortran subroutines for calculating and ordering the eigenvalues of a real upper Hessenberg matrix", ACM Trans. Math. Software 2 (1970), pp. 275-280.
- [33] G.W.Stewart; "Error and perturbation bounds for subspaces associated with certain eigenvalue problems", SIAM Review, 15 (1973), pp. 727-764.
- [34] C.vanLoan; "A symplectic method for approximating all the eigenvalues of a Hamiltonian matrix", Lin. Alg. Appl. 61 (1984), pp. 233-251.
- [35] J.W.Varah; "The calculation of the eigenvectors of a general complex matrix by inverse iteration", Math. Comp. 22 (1968), pp.785-791.