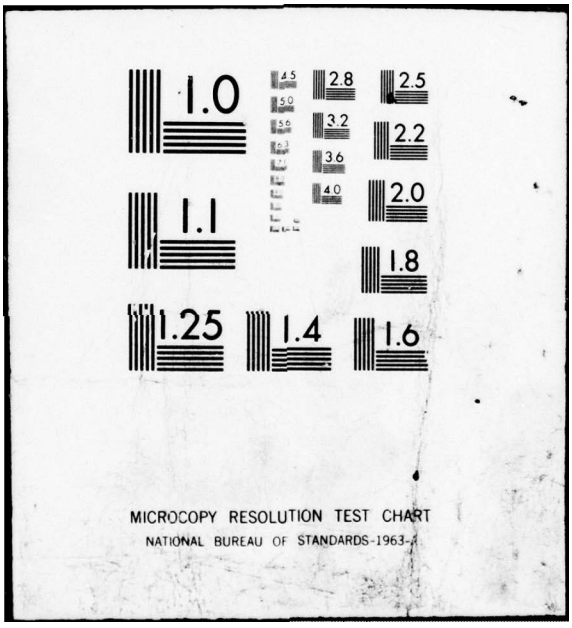
AD-A065 285 STANFORD UNIV CALIF DEPT OF COMPUTER SCIENCE F/G 12/1
SPARSE AND PARALLEL MATRIX COMPUTATIONS.(U)
DEC 78 F T LUK
UNCLASSIFIED STAN-CS-78-685 DAHC04-75-G-0185
NL

1 OF 2
AD A065 285


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

LEVEL II

13 NW

AD A065285

SPARSE AND PARALLEL MATRIX COMPUTATIONS

by

Franklin Tai-cheung Luk

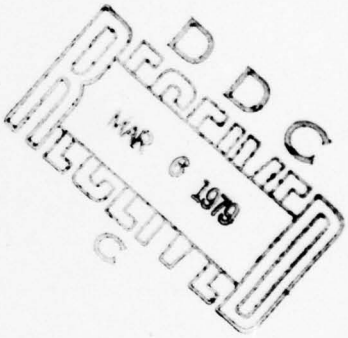
STAN-CS-78-685
DECEMBER 1978

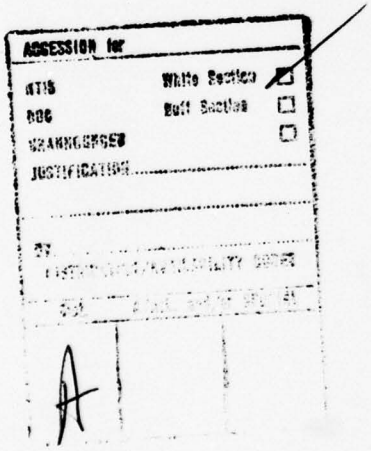
COMPUTER SCIENCE DEPARTMENT
School of Humanities and Sciences
STANFORD UNIVERSITY

SPARSE AND PARALLEL MATRIX COMPUTATIONS

by

Franklin Tai-cheung Luk





Research supported in part under U. S. Army Research Office grant DAHC04-75-G-0185.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER STAN-CS-78-685	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) SPARSE AND PARALLEL MATRIX COMPUTATIONS		5. TYPE OF REPORT & PERIOD COVERED Technical, December 1978
		6. PERFORMING ORG. REPORT NUMBER STAN-CS-78-685
7. AUTHOR(s) Franklin Tai-cheung Luk		8. CONTRACT OR GRANT NUMBER(s) DAHC04-75-G-0185
9. PERFORMING ORGANIZATION NAME AND ADDRESS Computer Science Department Stanford University Stanford, California 94305		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS U.S. Army Research Office Box 12211 Triangle Park, N.C. 27709		12. REPORT DATE December 1978 13. NUMBER OF PAGES 168 p.
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Releasable without limitations on dissemination.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number)		

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This thesis deals with four important matrix problems: (1) the application of many variants of the conjugate gradient method for solving matrix equations, (2) the solution of lower and upper bounds quadratic programs associated with M-matrices, (3) the construction of a Block Lanczos method for computing the greatest singular values of a matrix, and (4) the computation of the singular value decomposition of a matrix on the ILLIAC-IV computer.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

ABSTRACT.

This thesis deals with four important matrix problems: (1) the application of many variants of the conjugate gradient method for solving matrix equations, (2) the solution of lower and upper bounds quadratic programs associated with M-matrices, (3) the construction of a Block Lanczos method for computing the greatest singular values of a matrix, and (4) the computation of the singular value decomposition of a matrix on the ILLIAC-IV computer.

ACKNOWLEDGMENTS

I am deeply indebted to Professor Gene Golub for teaching me about numerical linear algebra and for providing guidance and support throughout my years of graduate study.

I appreciate the suggestions of Professors Joseph Oliger and James Wilkinson which have greatly improved my presentation.

Three researchers have aided me in essential ways. The many discussions with Professor Marcello Pagano led to the work on quadratic programming. Dr. Louis Hageman and Professor David Young shared with me their knowledge of iterative methods.

I would like to thank all the Serra House residents, particularly Mr. William Coughran and Mrs. Janet Wright, for their help and friendship.

The expert typing and editing skills of Miss Rosemarie Stampfel have made my task of writing much easier.

With sincere gratitude I wish to thank my parents, for starting me on my studies in this country; Gene and Rosemarie, for making life so pleasant; and my wife Vivian, for her love.

TABLE OF CONTENTS

TABLE OF CONTENTS Continued

TABLE OF ALGORITHMS

TABLE OF ABBREVIATIONS

Abbreviation	Name	First appears on page
CG method	Conjugate Gradient method	1
J-CG method	CG-accelerated single block Jacobi method	29
DJ-CG method	CG-accelerated double block Jacobi method	29
CJ-CG method	Compressed J-CG method	33
RS-CG method	CG method applied to reduced system of linear equations	36
SVD	Singular Value Decomposition	134
MINFIT	Algorithm for Least Squares problems	148

I. INTRODUCTION

This thesis consists of five chapters. In the next four chapters, we discuss the solution of four important matrix problems under the assumption that they are all of large order. For the first three problems we construct algorithms which exploit the sparsity of the associated matrices. We use a computer with parallel processing abilities to solve the fourth problem.

Chapter 2 of this thesis deals with the many variants of the conjugate gradient (CG) method for solving matrix equations. First, we describe a new preconditioned CG method and derive its many attractive properties. Second, we introduce the CG-accelerated single and double methods and give conditions for their mathematical equivalence. Third, we apply the equivalence result to systems of linear equations possessing 'Property A'. Fourth, this original result is used to show that a new method based on conjugate gradients and cyclic reduction is equivalent to a popular method of Reid. Fifth, we demonstrate that our new method is more efficient than Reid's method in both work and storage.

In Chapter 3 we study the problem of quadratic programming with M-matrices. We describe (1) an effective algorithm for the case where the variables are subject to a lower bound constraint, and (2) an analogous algorithm for the case where the variables are subject to lower and upper bounds constraints. We demonstrate the special monotone behavior of the iterate and gradient vectors.

The result on the gradient vector is new. It leads us to consider a simple updating procedure which preserves the monotonicity of both vectors. The procedure uses the fact that an M-matrix has a non-negative inverse. Two new algorithms are then constructed by incorporating this updating procedure into the two given algorithms. We give numerical examples which show that the new methods can be twice as fast as the original ones.

A Block Lanczos method is introduced in Chapter 4 for computing a few greatest singular values and associated vectors of a matrix. We present a theoretical development of the method and give a theorem on its rate of convergence. The practical implementation aspects are then discussed and particular attention is paid to the choice of block size. We believe that all our results are original. We are in fact unaware of any other algorithms which solve this problem.

In the fifth chapter, we study the computation of the singular value decomposition of a matrix on the ILLIAC IV computer. We describe the architecture of the machine and explain why the standard Golub-Reinsch algorithm is not applicable to this problem. We then present a one-sided orthogonalization method which makes very efficient use of the parallel computing abilities of the ILLIAC machine. Our method is shown to be Jacobi-like and numerically stable. Finally, a comparison of our method on the ILLIAC IV computer with the Golub-Reinsch algorithm on a conventional machine demonstrates the great potential of parallel computers in the important area of matrix computations.

II. CONJUGATE GRADIENT ACCELERATION OF ITERATIVE METHODS

2.1. Introduction

In this chapter we are concerned with the application of the conjugate gradient method (CG method) as an iterative solution procedure (cf. Reid [8]) for large and sparse sets of linear equations.

Let us consider the system

$$A\underset{\sim}{x} = \underset{\sim}{b} , \tag{1.1}$$

where A is a given $n \times n$ nonsingular matrix and $\underset{\sim}{b}$ a given column vector. We assume that the matrix A is large and sparse. Instead of (1.1) we may solve the modified system

$$C^{-1}A\underset{\sim}{x} = C^{-1}\underset{\sim}{b} , \tag{1.2}$$

where C is some $n \times n$ nonsingular matrix such that it is a simple computing task to solve the system

$$C\underset{\sim}{z} = \underset{\sim}{d} .$$

We usually choose C so that the new system (1.2) is "better-conditioned" with respect to inversion than the original system (1.1) (cf. [1]), i.e.

$$\kappa(C^{-1}A) < \kappa(A) ,$$

where $\kappa(M) = \|M\| \cdot \|M^{-1}\|$ for any nonsingular matrix M. We refer to $\kappa(M)$ as the condition number of M with respect to inversion (and with respect to the matrix norm $\|\cdot\|$). The matrix C is frequently

an approximation of A so that the condition number of $C^{-1}A$ is small, e.g., C can be a product of sparse triangular matrices (see [7]).

We wish to apply the conjugate gradient method to the solution of the preconditioned system (1.2). We refer to the new method as the preconditioned conjugate gradient method (preconditioned CG method). This problem has been studied by many researchers (see [1] for a bibliography), who considered the case where A and C are both symmetric and positive definite matrices. We make a slightly weaker assumption here.

ASSUMPTION 1. There exists some nonsingular matrix S of order n such that the matrix $SC^{-1}AS^{-1}$ is symmetric and positive definite.

We call S a "symmetrization" matrix. The next lemma shows that Assumption 1 is meaningful.

LEMMA 1.1. Suppose that both matrices A and C are symmetric and positive definite. If we choose the matrix S such that $S^tS = C$, then Assumption 1 is valid.

Proof. Let $B \equiv SC^{-1}AS^{-1}$. The matrix B is symmetric because

$$B = S(S^tS)^{-1}AS^{-1} = (S^{-1})^tAS^{-1} ,$$

and is positive definite because

$$\underset{\sim}{y}^tB\underset{\sim}{y} = (S^{-1}\underset{\sim}{y})^t A(S^{-1}\underset{\sim}{y}) > 0 \qquad \text{for } \underset{\sim}{y} \neq \underset{\sim}{0} . \qquad \square$$

This apparently new concept of a "symmetrization" matrix makes it a straightforward exercise (in Section 2.3) to derive the properties of the preconditioned CG method from those of the CG method. We see in the same section that the matrix S need not be formed explicitly and that only the product S^tS is required for the preconditioned CG method. For many problems, several choices of the "symmetrization" matrix are possible (see, e.g., Section 2.4). An interesting problem is therefore to determine the effect of the "symmetrization" matrix on the convergence rate of the preconditioned CG method.

Letting

$$G \equiv I - C^{-1}A \tag{1.3}$$

and

$$\underset{\sim}{k} \equiv C^{-1}\underset{\sim}{b} , \tag{1.4}$$

we can rewrite the system (1.2) as

$$(I - G)\underset{\sim}{x} = \underset{\sim}{k} , \tag{1.5}$$

from which we derive the iterative scheme

$$\underset{\sim}{x}^{(i+1)} = G\underset{\sim}{x}^{(i)} + \underset{\sim}{k} \quad \text{for} \quad i = 0,1,\ldots, \tag{1.6}$$

where $\underset{\sim}{x}^{(0)}$ is some initial approximation to $\underset{\sim}{x}$. We refer to (1.6) as the <u>single method</u> for the system (1.5). Let us write the matrix A as

$$A = D + L + U ,$$

where D is a (block) diagonal matrix, L is a strictly lower triangular matrix and U is a strictly upper triangular matrix. We then call (1.6) the (block) Jacobi method if $C = D$, the (block) Gauss-Seidel method if $C = D + L$, or the symmetric (block) successive over-relaxation method (SSOR method) if

$$C = (D + \omega L)\, D^{-1}(D + \omega U) ,$$

where ω is a scalar parameter. Other choices of C lead to other well-known single methods.

We can combine the iterates from the single method to define a more general iterative procedure

$$\underset{\sim}{w}^{(i)} = \sum_{j=0}^{i} \nu_j^{(i)}\, \underset{\sim}{x}^{(j)} \quad \text{for} \quad i = 0,1,\ldots, \tag{1.7}$$

where the $\nu_j^{(i)}$'s are constants such that

$$\sum_{j=0}^{i} \nu_j^{(i)} = 1 \qquad \text{for} \quad i = 0,1,\ldots . \tag{1.8}$$

We call (1.7) a <u>semi-iterative method</u> with respect to the method (1.6) (cf. [11]). A particular instance of (1.7) is the second-order procedure

$$\underset{\sim}{x}^{(i+1)} = \alpha_{i+1}[\beta_i(G\underset{\sim}{x}^{(i)} + \underset{\sim}{k}) + (1 - \beta_i)\underset{\sim}{x}^{(i)}] + (1 - \alpha_{i+1})\underset{\sim}{x}^{(i-1)} , \tag{1.9}$$

$$\text{for} \quad i = 0,1,\ldots ,$$

where $\alpha_1 = 1$. Many methods can be expressed in the form of (1.9), e.g. the Chebyshev semi-iterative method and the Richardson second-order method (see [4]). In the next section, we show that we can also express the conjugate gradient method in this three-term recurrence form. We refer to (1.9) as the <u>CG-accelerated single method</u> if the iteration parameters $\{\alpha_i\}$ and $\{\beta_i\}$ are chosen based on the CG method.

Combining two iterations of the single method (1.6), we obtain the double method

$$\underset{\sim}{y}^{(i+1)} = G^2\underset{\sim}{y}^{(i)} + G\underset{\sim}{k} + \underset{\sim}{k} \quad \text{for} \quad i = 0,1,\ldots, \tag{1.10}$$

where $\underset{\sim}{y}^{(0)} = \underset{\sim}{x}^{(0)}$. The iterative procedure (1.10) can be accelerated by the second-order procedure

$$\underset{\sim}{y}^{(i+1)} = \gamma_{i+1}[\rho_i(G^2\underset{\sim}{y}^{(i)} + G\underset{\sim}{k} + \underset{\sim}{k}) + (1-\rho_i)\underset{\sim}{y}^{(i)}] + (1-\gamma_{i+1})\underset{\sim}{y}^{(i-1)} \quad \text{for} \quad i = 0,1,\ldots, \tag{1.11}$$

where $\gamma_1 = 1$. We want to use the CG method to compute the iteration parameters $\{\gamma_i\}$ and $\{\rho_i\}$, in which case the procedure (1.11) is referred to as the CG-accelerated double method. We need the following assumption.

ASSUMPTION 2. There exists some nonsingular matrix T of order n such that the matrix $T(I - G^2)T^{-1}$ is symmetric and positive definite.

It can be easily checked that $T = S$ is one possible choice.

We have five goals to accomplish in this chapter: (1) to describe the preconditioned CG method and to derive its many nice properties, (2) to determine conditions under which the CG-accelerated single method and the CG-accelerated double method are "virtually-equivalent," i.e.

$$\underset{\sim}{x}^{(2i)} = \underset{\sim}{y}^{(i)} \quad \text{for} \quad i = 0,1,\ldots, \tag{1.12}$$

(3) to apply the equivalence result of (2) to the case of the block Jacobi method when the coefficient matrix A is symmetric and positive definite, and can be written in partitioned form as

$$A = \left(\begin{array}{c|c} M_1 & F \\ \hline F^t & M_2 \end{array}\right) ,$$

where M_1 and M_2 are square matrices such that the systems

$$M_1 \underset{\sim}{z}_1 = \underset{\sim}{d}_1 \qquad \text{and} \qquad M_2 \underset{\sim}{z}_2 = \underset{\sim}{d}_2$$

are easy to solve, (4) to introduce a new method based on conjugate gradients and cyclic reduction for the class of matrix equations considered in (3), and to use the result of (3) to show that our new method is mathematically equivalent to a generalization of a method due to Reid [9], and (5) to demonstrate that our new method is more efficient than the generalized Reid's method in both work and storage, and thus is an effective solution procedure for an important class of matrix equations. We believe that all five results are original.

2.2. Conjugate Gradient Method

Let us consider the system

$$A\underset{\sim}{x} = \underset{\sim}{b} . \qquad (1.1)$$

We assume that the matrix A is symmetric and positive definite, and that the matrix C has been chosen as the identity matrix. Our goal is to use the conjugate gradient method to compute the iteration

parameters $\{\alpha_i\}$ and $\{\beta_i\}$ in the second-order procedure

$$\underset{\sim}{x}^{(i+1)} = \alpha_{i+1}[\beta_i(G\underset{\sim}{x}^{(i)} + \underset{\sim}{k}) + (1-\beta_i)\underset{\sim}{x}^{(i)}] + (1-\alpha_{i+1})\underset{\sim}{x}^{(i-1)} , \qquad \text{for } i = 0,1,\ldots , \tag{1.9}$$

where $\alpha_1 = 1$. We define the residual vector $\underset{\sim}{r}^{(i)}$ by

$$\underset{\sim}{r}^{(i)} \equiv \underset{\sim}{b} - A\underset{\sim}{x}^{(i)} \qquad \text{for } i = 0,1,\ldots . \tag{2.1}$$

As we have

$$G = I - A \qquad \text{and} \qquad \underset{\sim}{k} = \underset{\sim}{b} ,$$

it follows that

$$G\underset{\sim}{x}^{(i)} + \underset{\sim}{k} = \underset{\sim}{x}^{(i)} + \underset{\sim}{r}^{(i)} ,$$

and the procedure (1.9) becomes

$$\underset{\sim}{x}^{(i+1)} = \alpha_{i+1}(\beta_i\underset{\sim}{r}^{(i)} + \underset{\sim}{x}^{(i)}) + (1 - \alpha_{i+1})\underset{\sim}{x}^{(i-1)} \qquad \text{for } i = 0,1,\ldots , \tag{2.2}$$

where $\alpha_1 = 1$. Using the notation

$$(\underset{\sim}{p},\underset{\sim}{q}) \equiv \underset{\sim}{p}^t\underset{\sim}{q} ,$$

we can express the conjugate gradient method for solving the system (1.1) as follows (cf. Rutishauser [10]).

ALGORITHM 2.1 (CG method)

(1) Let $\underset{\sim}{x}^{(0)}$ be an initial approximation to $\underset{\sim}{x}$. Compute

$$\underset{\sim}{r}^{(0)} := \underset{\sim}{b} - A\underset{\sim}{x}^{(0)} ,$$

$$\beta_0 := \frac{(\underset{\sim}{r}^{(0)}, \underset{\sim}{r}^{(0)})}{(\underset{\sim}{r}^{(0)}, A\underset{\sim}{r}^{(0)})}$$

and

$$\underset{\sim}{x}^{(1)} := \beta_0 \underset{\sim}{r}^{(0)} + \underset{\sim}{x}^{(0)} .$$

Let $\alpha_1 := 1$ and $i := 0$.

(2) Repeat until $\underset{\sim}{r}^{(i)} = \underset{\sim}{0}$:

(a) Let

$$i := i + 1 .$$

(b) Compute

$$\underset{\sim}{r}^{(i)} := \underset{\sim}{b} - A\underset{\sim}{x}^{(i)} ,$$

$$\beta_i := \frac{(\underset{\sim}{r}^{(i)}, \underset{\sim}{r}^{(i)})}{(\underset{\sim}{r}^{(i)}, A\underset{\sim}{r}^{(i)})}$$

and

$$\alpha_{i+1} := \left[1 - \frac{\beta_i}{\beta_{i-1}} \frac{(\underset{\sim}{r}^{(i)}, \underset{\sim}{r}^{(i)})}{(\underset{\sim}{r}^{(i-1)}, \underset{\sim}{r}^{(i-1)})} \frac{1}{\alpha_i} \right]^{-1} .$$

(c) Compute

$$\underset{\sim}{x}^{(i+1)} := \alpha_{i+1}[\beta_i \underset{\sim}{r}^{(i)} + \underset{\sim}{x}^{(i)}] + (1 - \alpha_{i+1})\underset{\sim}{x}^{(i-1)} . \quad \square$$

The residual vectors satisfy the well-known orthogonality property (see [6])

$$(\underset{\sim}{r}^{(i)}, \underset{\sim}{r}^{(j)}) = 0 \qquad \text{for } i \neq j . \tag{2.3}$$

Thus, in the absence of round-off errors the solution vector is obtained in at most n iterations of the CG-method. But the computed residuals are not orthogonal in practice. Our approach is to permit the gradual loss of orthogonality and with it the finite termination property of the method. We are concerned primarily with the iterative aspects of the CG algorithm. In fact, when used for solving large sparse sets of linear equations arising from the discretization of elliptic partial differential equations, the CG method often converges within a number of iterations small compared with n (see, e.g., [2]). Nonetheless, this orthogonality property (2.3) is of theoretical importance for it characterizes the CG method among all second-order procedures of the form (2.2).

THEOREM 2.1. The second-order iterative procedure

$$\underset{\sim}{x}^{(i+1)} = \alpha_{i+1}(\beta_i \underset{\sim}{r}^{(i)} + \underset{\sim}{x}^{(i)}) + (1 - \alpha_{i+1})\underset{\sim}{x}^{(i-1)} \quad \text{for} \quad i = 0,1,\ldots ,$$

where $\alpha_1 = 1$ and

$$\underset{\sim}{r}^{(i)} = \underset{\sim}{b} - A\underset{\sim}{x}^{(i)} \qquad \text{for} \quad i = 0,1,\ldots,$$

reduces to the conjugate gradient method (Algorithm 2.1) if and only if

$$(\underset{\sim}{r}^{(i)}, \underset{\sim}{r}^{(j)}) = 0 \qquad \text{for} \quad i \neq j .$$

Proof. We only have to prove the sufficiency part. Let $i = 0$. From (2.2), we obtain

$$\underset{\sim}{x}^{(1)} = \beta_0 \underset{\sim}{r}^{(0)} + \underset{\sim}{x}^{(0)} .$$

Thus,

$$\underset{\sim}{b} - A\underset{\sim}{x}^{(1)} = -\beta_0 A\underset{\sim}{r}^{(0)} + \underset{\sim}{b} - A\underset{\sim}{x}^{(0)} ,$$

which becomes

$$\underset{\sim}{r}^{(1)} = \underset{\sim}{r}^{(0)} - \beta_0 A\underset{\sim}{r}^{(0)} . \tag{2.4}$$

As $(\underset{\sim}{r}^{(0)}, \underset{\sim}{r}^{(1)}) = 0$, we find

$$0 = (\underset{\sim}{r}^{(0)}, \underset{\sim}{r}^{(0)}) - \beta_0 (\underset{\sim}{r}^{(0)}, A\underset{\sim}{r}^{(0)}) .$$

It follows that

$$\beta_0 = \frac{(\underset{\sim}{r}^{(0)}, \underset{\sim}{r}^{(0)})}{(\underset{\sim}{r}^{(0)}, A\underset{\sim}{r}^{(0)})} .$$

Now let $i \geq 1$. From (2.2), we get

$$\underset{\sim}{r}^{(i+1)} = \alpha_{i+1}(-\beta_i A\underset{\sim}{r}^{(i)} + \underset{\sim}{r}^{(i)}) + (1 - \alpha_{i+1})\underset{\sim}{r}^{(i-1)} . \tag{2.5}$$

Therefore,

$$0 = \alpha_{i+1}[-\beta_i (\underset{\sim}{r}^{(i)}, Ar^{(i)}) + (\underset{\sim}{r}^{(i)}, \underset{\sim}{r}^{(i)})]$$

because

$$(\underset{\sim}{r}^{(i)}, \underset{\sim}{r}^{(i+1)}) = (\underset{\sim}{r}^{(i)}, \underset{\sim}{r}^{(i-1)}) = 0 .$$

Consequently,

$$\beta_i = \frac{(\underset{\sim}{r}^{(i)}, \underset{\sim}{r}^{(i)})}{(\underset{\sim}{r}^{(i)}, A\underset{\sim}{r}^{(i)})} .$$

From (2.5), as $(\underset{\sim}{r}^{(i-1)}, \underset{\sim}{r}^{(i+1)}) = (\underset{\sim}{r}^{(i-1)}, \underset{\sim}{r}^{(i)}) = 0$, we get

$$0 = -\alpha_{i+1}\beta_i (\underset{\sim}{r}^{(i-1)}, A\underset{\sim}{r}^{(i)}) + (1 - \alpha_{i+1})(\underset{\sim}{r}^{(i-1)}, \underset{\sim}{r}^{(i-1)}) . \tag{2.6}$$

Since $\alpha_1 = 1$, we can replace i by i-1 in (2.5) to obtain

$$\underset{\sim}{r}^{(i)} = \alpha_i(-\beta_{i-1}A\underset{\sim}{r}^{(i-1)} + \underset{\sim}{r}^{(i-1)}) + (1 - \alpha_i)\underset{\sim}{r}^{(i-2)} ,$$

or, equivalently,

$$A\underset{\sim}{r}^{(i-1)} = -\frac{1}{\beta_{i-1}}\left\{\frac{1}{\alpha_i}[\underset{\sim}{r}^{(i)} - (1 - \alpha_i)\underset{\sim}{r}^{(i-2)}] - \underset{\sim}{r}^{(i-1)}\right\} .$$

Therefore,

$$(\underset{\sim}{r}^{(i-1)}, A\underset{\sim}{r}^{(i)}) = (A\underset{\sim}{r}^{(i-1)}, \underset{\sim}{r}^{(i)})$$

$$= -\frac{1}{\alpha_i\beta_{i-1}}(\underset{\sim}{r}^{(i)}, \underset{\sim}{r}^{(i)})$$

because $(\underset{\sim}{r}^{(i-2)}, \underset{\sim}{r}^{(i)}) = (\underset{\sim}{r}^{(i-1)}, \underset{\sim}{r}^{(i)}) = 0$. We can now simplify (2.6) to

$$0 = \frac{\alpha_{i+1}\beta_i}{\alpha_i\beta_{i-1}}(\underset{\sim}{r}^{(i)}, \underset{\sim}{r}^{(i)}) + (1 - \alpha_{i+1})(\underset{\sim}{r}^{(i-1)}, \underset{\sim}{r}^{(i-1)}) ,$$

or equivalently,

$$0 = \left[\frac{\beta_i}{\beta_{i-1}} \frac{(\underset{\sim}{r}^{(i)}, \underset{\sim}{r}^{(i)})}{(\underset{\sim}{r}^{(i-1)}, \underset{\sim}{r}^{(i-1)})} \frac{1}{\alpha_i} - 1\right]\alpha_{i+1} + 1 .$$

We conclude that

$$\alpha_{i+1} = \left[1 - \frac{\beta_i}{\beta_{i-1}} \frac{(\underset{\sim}{r}^{(i)}, \underset{\sim}{r}^{(i)})}{(\underset{\sim}{r}^{(i-1)}, \underset{\sim}{r}^{(i-1)})} \frac{1}{\alpha_i}\right]^{-1} . \qquad \square$$

From the recurrence relations (2.4) and (2.5), we can verify by induction that

$$\underset{\sim}{r}^{(i+1)} = \underset{\sim}{r}^{(0)} - A[\sum_{j=0}^{i} \sigma_j^{(i+1)} A^j]\underset{\sim}{r}^{(0)} \quad \text{for} \quad i = 0,1,\ldots, \qquad (2.7)$$

where the $\sigma_j^{(i+1)}$'s are constants. As

$$\underset{\sim}{r}^{(i+1)} - \underset{\sim}{r}^{(0)} = -A(\underset{\sim}{x}^{(i+1)} - \underset{\sim}{x}^{(0)}) ,$$

it follows that

$$\underset{\sim}{x}^{(i+1)} = \underset{\sim}{x}^{(0)} + [\sum_{j=0}^{i} \sigma_j^{(i+1)} A^j]\underset{\sim}{r}^{(0)} \quad \text{for } i = 0,1,\ldots . \quad (2.8)$$

The conjugate gradient method is therefore a special instance of the polynomial acceleration procedure

$$\underset{\sim}{x}^{(i+1)} = \underset{\sim}{x}^{(0)} + P_i(A)\underset{\sim}{r}^{(0)} \quad \text{for } i = 0,1,\ldots , \quad (2.9)$$

where $P_i(A)$ is a matrix polynomial in A of degree i. Let us define an error function

$$f_M(\underset{\sim}{y}) \equiv (\underset{\sim}{y} - \underset{\sim}{x}, M(\underset{\sim}{y} - \underset{\sim}{x})) , \quad (2.10)$$

where M is some $n \times n$ positive definite matrix. The conjugate gradient method possesses the following optimality property (see [3, p. 397]).

THEOREM 2.2. Among all polynomial acceleration procedures of the form (2.9), the conjugate gradient method generates an iterate $\underset{\sim}{x}^{(i)}$ that minimizes the error function $f_A(\underset{\sim}{x}^{(i)})$ of (2.10), for $i = 1,2,\ldots$.

A few interesting results follow from Theorem 2.2. Let us assume that the matrix A has only $p < n$ distinct eigenvalues. Then there exists a matrix polynomial $Q_p(A)$ of degree p so that

$$Q_p(A) = 0 .$$

Therefore,

$$f_A(\underset{\sim}{x}^{(p)}) = 0$$

and the CG method converges in at most p steps. The same result also holds if the matrix A has a large number of distinct eigenvalues but the initial error vector

$$\underset{\sim}{e}^{(0)} = \underset{\sim}{x}^{(0)} - \underset{\sim}{x}$$

lies in a subspace generated by the eigenvectors associated with only p of these eigenvalues. The CG method is thus optimal for the particular eigenvector mix of the initial error $\underset{\sim}{e}^{(0)}$. As the iteration proceeds, the extreme eigenvalues are approximated especially well and the CG procedure would then behave as if the corresponding eigenvectors were not present. Hence we often observe a superlinear convergence rate for the CG method (see, e.g., [2]). It follows that for rapid convergence the eigenvalues should be sparse at the extremes and dense in the center. Convergence would be slow if the eigenvalues were packed in the extremes.

Another consequence of Theorem 2.2 is an error estimate of the CG method. Assume that the eigenvalues of matrix A are included in the interval $[\alpha,\beta]$, where $\alpha > 0$. Then by choosing an appropriate Chebyshev polynomial acceleration, we get (see [3, p. 428])

$$f_A(\underset{\sim}{x}^{(i)}) \leq \frac{1}{T_i^2\left(\frac{\beta + \alpha}{\beta - \alpha}\right)} \; f_A(\underset{\sim}{x}^{(0)}) \,, \qquad (2.11)$$

where T_i is the i-th degree Chebyshev polynomial of the first kind.

Since

$$2 \cdot T_i\left(\frac{\beta + \alpha}{\beta - \alpha}\right) = \left(\frac{\sqrt{\kappa} + 1}{\sqrt{\kappa} - 1}\right)^i + \left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^i$$

$$\geq \left(\frac{\sqrt{\kappa} + 1}{\sqrt{\kappa} - 1}\right)^i ,$$

where $\kappa = \beta/\alpha$, the inequality (2.11) becomes

$$f_A(\underset{\sim}{x}^{(i)}) \leq 4\left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^{2i} f_A(\underset{\sim}{x}^{(0)}) . \qquad (2.12)$$

For fixed i, the error bound of (2.12) decreases as κ decreases, and is thus tightest when α and β are the extreme eigenvalues of A, in which case the number κ is the condition number of A with respect to the spectral norm. The bound is, however, pessimistic asymptotically.

The above convergence properties of the CG method are important when we consider the choice of a preconditioning matrix in the next section.

2.3. Preconditioned Conjugate Gradient Method

We wish to apply the conjugate gradient method to solve the preconditioned system

$$C^{-1}A\underset{\sim}{x} = C^{-1}\underset{\sim}{b} . \qquad (1.2)$$

We construct the matrix equation

$$SC^{-1}AS^{-1}\underset{\sim}{u} = SC^{-1}\underset{\sim}{b} , \qquad (3.1)$$

where $\underset{\sim}{u} = S\underset{\sim}{x}$ and S is the "symmetrization" matrix of Assumption 1. Now we apply the conjugate gradient method to the system (3.1).

ALGORITHM 3.1.

(1) Let $\underset{\sim}{u}^{(0)}$ be an initial approximation to $\underset{\sim}{u}$. Compute

$$\bar{\underset{\sim}{r}}^{(0)} := SC^{-1}\underset{\sim}{b} - SC^{-1}AS^{-1}\underset{\sim}{u}^{(0)} ,$$

$$\beta_0 := \frac{(\bar{\underset{\sim}{r}}^{(0)}, \bar{\underset{\sim}{r}}^{(0)})}{(\bar{\underset{\sim}{r}}^{(0)}, A\bar{\underset{\sim}{r}}^{(0)})}$$

and

$$\underset{\sim}{u}^{(1)} := \beta_0\bar{\underset{\sim}{r}}^{(0)} + \underset{\sim}{u}^{(0)} .$$

Let $\alpha_1 := 1$ and $i := 0$.

(2) Repeat until $\bar{\underset{\sim}{r}}^{(i)} = \underset{\sim}{0}$:

(a) Let

$$i := i + 1 .$$

(b) Compute

$$\bar{\underset{\sim}{r}}^{(i)} := SC^{-1}\underset{\sim}{b} - SC^{-1}AS^{-1}\underset{\sim}{u}^{(i)} ,$$

$$\beta_i := \frac{(\bar{\underset{\sim}{r}}^{(i)}, \bar{\underset{\sim}{r}}^{(i)})}{(\bar{\underset{\sim}{r}}^{(i)}, SC^{-1}AS^{-1}\bar{\underset{\sim}{r}}^{(i)})}$$

and

$$\alpha_{i+1} := \left[1 - \frac{\beta_i}{\beta_{i-1}} \frac{(\bar{\underset{\sim}{r}}^{(i)}, \bar{\underset{\sim}{r}}^{(i)})}{(\bar{\underset{\sim}{r}}^{(i-1)}, \bar{\underset{\sim}{r}}^{(i-1)})} \frac{1}{\alpha_i} \right]^{-1} .$$

(c) Compute

$$\underset{\sim}{u}^{(i+1)} := \alpha_{i+1}(\beta_i\bar{\underset{\sim}{r}}^{(i)} + \underset{\sim}{u}^{(i)}) + (1 - \alpha_{i+1})\underset{\sim}{u}^{(i-1)} . \quad \square$$

Let us define the iterate vector $\underset{\sim}{x}^{(i)}$ by

$$\underset{\sim}{x}^{(i)} \equiv S^{-1}\underset{\sim}{u}^{(i)} \qquad \text{for } i = 0, 1, \ldots , \tag{3.2}$$

and the pseudo-residual vector $\underset{\sim}{z}^{(i)}$ by

$$\underset{\sim}{z}^{(i)} \equiv C^{-1}\underset{\sim}{b} - C^{-1}A\underset{\sim}{x}^{(i)} \qquad \text{for} \quad i = 0, 1, \ldots . \tag{3.3}$$

Substituting $S\underset{\sim}{x}^{(i)}$ for $\underset{\sim}{u}^{(i)}$ and $S\underset{\sim}{z}^{(i)}$ for $\underset{\sim}{\bar{r}}^{(i)}$ in Algorithm 3.1, we get an iterative procedure for

$$C^{-1}A\underset{\sim}{x} = C^{-1}\underset{\sim}{b} \ . \tag{1.2}$$

ALGORITHM 3.2. (Preconditioned CG method or CG-accelerated single method)

(1) Let $\underset{\sim}{x}^{(0)}$ be an initial approximation to $\underset{\sim}{x}$. Solve the system

$$C\underset{\sim}{z}^{(0)} = \underset{\sim}{b} - A\underset{\sim}{x}^{(0)} \ .$$

Compute

$$\beta_0 := \frac{(S\underset{\sim}{z}^{(0)},\ S\underset{\sim}{z}^{(0)})}{(S\underset{\sim}{z}^{(0)},\ SC^{-1}A\underset{\sim}{z}^{(0)})}$$

and

$$\underset{\sim}{x}^{(1)} := \beta_0\underset{\sim}{z}^{(0)} + \underset{\sim}{x}^{(0)} \ .$$

Let $\alpha_1 := 1$ and $i := 0$.

(2) Repeat until $\underset{\sim}{z}^{(i)} = \underset{\sim}{0}$:

(a) Let $i := i + 1$.

(b) Solve the system

$$C\underset{\sim}{z}^{(i)} = \underset{\sim}{b} - A\underset{\sim}{x}^{(i)} \ .$$

(c) Compute

$$\beta_i := \frac{(S\underset{\sim}{z}^{(i)},\ S\underset{\sim}{z}^{(i)})}{(S\underset{\sim}{z}^{(i)},\ SC^{-1}A\underset{\sim}{z}^{(i)})} \ ,$$

$$\alpha_{i+1} := \left[1 - \frac{\beta_i}{\beta_{i-1}} \frac{(S\underset{\sim}{z}^{(i)},\ S\underset{\sim}{z}^{(i)})}{(S\underset{\sim}{z}^{(i-1)},\ S\underset{\sim}{z}^{(i-1)})} \frac{1}{\alpha_i}\right]^{-1}$$

and

$$\underset{\sim}{x}^{(i+1)} := \alpha_{i+1}(\beta_i \underset{\sim}{z}^{(i)} + \underset{\sim}{x}^{(i)}) + (1 - \alpha_{i+1})\underset{\sim}{x}^{(i-1)} \quad . \quad \square$$

Observe that we need not form the matrix S explicitly. It suffices to have the product S^tS. Also, note that the preconditioned CG method reduces to the generalized CG method of Concus et al. [2] if A and C are symmetric and positive definite matrices, and if $S^tS = C$.

From (2.3) and (3.3), we get that the pseudo-residual vectors satisfy the conjugacy property

$$(S\underset{\sim}{z}^{(i)}, S\underset{\sim}{z}^{(j)}) = 0 \qquad \text{for } i \neq j \, . \tag{3.4}$$

Since the matrix S^tS is symmetric and positive definite, the preconditioned CG method therefore terminates in at most n steps in exact arithmetic. In practice, the calculated pseudo-residuals do not satisfy (3.4) due to round-off errors. Our approach is to permit the gradual loss of conjugacy and to consider primarily the iterative aspect of the preconditioned CG method.

The next theorem, a direct consequence of Theorem 2.1, states how the conjugacy property of (3.4) characterizes the preconditioned CG method.

THEOREM 3.1. The second-order iterative procedure

$$\underset{\sim}{x}^{(i+1)} = \alpha_{i+1}(\beta_i \underset{\sim}{z}^{(i)} + \underset{\sim}{x}^{(i)}) + (1 - \alpha_{i+1})\underset{\sim}{x}^{(i-1)} \quad \text{for} \quad i = 0,1,\ldots ,$$

where $\alpha_1 = 1$ and

$$\underset{\sim}{z}^{(i)} = C^{-1}(\underset{\sim}{b} - A\underset{\sim}{x}^{(i)}) \qquad \text{for} \quad i = 0,1,\ldots ,$$

reduces to the preconditioned conjugate gradient method (Algorithm 3.2) if and only if

$$(S\underset{\sim}{z}^{(i)}, S\underset{\sim}{z}^{(j)}) = 0 \qquad \text{for} \quad i \neq j. \qquad \square$$

From the relation (2.8) we get

$$\underset{\sim}{u}^{(i+1)} = \underset{\sim}{u}^{(0)} + [\sum_{j=0}^{i} \tau_j^{(i+1)} H^j]\underset{\sim}{r}^{(0)} \quad \text{for} \quad i = 0,1,\ldots ,$$

where $H \equiv SC^{-1}AS^{-1}$ and the $\tau_j^{(i+1)}$'s are constants. Thus,

$$S\underset{\sim}{x}^{(i+1)} = S\underset{\sim}{x}^{(0)} + S[\sum_{j=0}^{i} \tau_j^{(i+1)} K^j]S^{-1}S\underset{\sim}{z}^{(0)} \quad \text{for} \quad i = 0,1,\ldots ,$$

where $K \equiv C^{-1}A$, so that

$$\underset{\sim}{x}^{(i+1)} = \underset{\sim}{x}^{(0)} + [\sum_{j=0}^{i} \tau_j^{(i+1)} K^j]\underset{\sim}{z}^{(0)} \qquad \text{for} \quad i = 0,1,\ldots . \qquad (3.5)$$

The preconditioned CG method is therefore a special instance of the polynomial acceleration procedure

$$\underset{\sim}{x}^{(i+1)} = \underset{\sim}{x}^{(0)} + Q_i(K)\,\underset{\sim}{z}^{(0)} , \qquad (3.6)$$

where $Q_i(K)$ is a matrix polynomial in K ($\equiv C^{-1}A$) of degree i. Recall the error function

$$f_M(\underset{\sim}{y}) \equiv (\underset{\sim}{y} - \underset{\sim}{x},\ M(\underset{\sim}{y} - \underset{\sim}{x}))\ , \tag{2.10}$$

where M is a positive definite matrix. As

$$(\underset{\sim}{u}^{(i)} - \underset{\sim}{u},\ SC^{-1}AS^{-1}(\underset{\sim}{u}^{(i)} - \underset{\sim}{u})) = (\underset{\sim}{x}^{(i)} - \underset{\sim}{x},\ S^tSC^{-1}A(\underset{\sim}{x}^{(i)} - \underset{\sim}{x})),$$

the following optimality property of the precondition CG method can be derived from Theorem 2.2.

THEOREM 3.2. Among all polynomial acceleration procedures of the form (3.6), the preconditioned CG method generates an iterate $\underset{\sim}{x}^{(i)}$ that minimizes the error function $f_B(\underset{\sim}{x}^{(i)})$ of (2.10), where $B \equiv S^tSC^{-1}A$, for $i = 1,2,\ldots$.

We can draw from Theorem 3.2 conclusions similar to those from Theorem 2.2. The assumption that the matrix $SC^{-1}AS^{-1}$ is symmetic and positive definite implies that the eigenvalues of $C^{-1}A$ are all real and positive. Had the matrix $C^{-1}A$ only $p < n$ distinct eigenvalues, the preconditioned CG method would converge in at most p steps. The eigenvalue distribution of $C^{-1}A$ determines the rate at which the method converges. Convergence would be fast if the eigenvalues were sparse in the extremes and dense in the interior. We often observe that the method converges superlinearly.

Using arguments similar to those in [3, p. 428] and the last section, we obtain an error estimate for the preconditioned CG method:

$$f_B(\underset{\sim}{x}^{(i)}) \leq 4\left(\frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1}\right)^{2i} f_B(\underset{\sim}{x}^{(0)}) , \qquad (3.7)$$

where $B \equiv S^t S C^{-1} A$ and κ is the spectral condition number of $C^{-1}A$.

A good preconditioning matrix C should therefore have the following features:

(a) it is a simple computing task to solve matrix equations with coefficient matrix C,

(b) the matrix $C^{-1}A$ has a favorable eigenvalue distribution, i.e. either that the eigenvalues are sparse in the extremes and dense in the interior or that only a few of them are distinct, and

(c) the spectral condition number of $C^{-1}A$ is much smaller than that of A.

2.4. Equivalence Results

Consider the double method

$$\underset{\sim}{y}^{(i+1)} = G^2 \underset{\sim}{y}^{(i)} + G\underset{\sim}{k} + \underset{\sim}{k} \qquad \text{for } i = 0,1,\ldots , \qquad (1.10)$$

where $\underset{\sim}{y}^{(0)} = \underset{\sim}{x}^{(0)}$. Recall that

$$G \equiv I - C^{-1}A \qquad (1.3)$$

and

$$\underset{\sim}{k} \equiv C^{-1}\underset{\sim}{b} . \qquad (1.4)$$

The procedure (1.10) can be regarded as the single method for the system

$$(I - G^2)\underset{\sim}{x} = G\underset{\sim}{k} + \underset{\sim}{k} . \tag{4.1}$$

Suppose that we have chosen a nonsingular matrix T such that the matrix $T(1 - G^2)T^{-1}$ is symmetric and positive definite. Let us apply the conjugate gradient method to the system

$$[T(I - G^2)T^{-1}]T\underset{\sim}{x} = T(G\underset{\sim}{k} + \underset{\sim}{k}) .$$

ALGORITHM 4.1. (CG-accelerated double method)

(1) Let

$$\underset{\sim}{y}^{(0)} := \underset{\sim}{x}^{(0)} .$$

Compute

$$\underset{\sim}{w}^{(0)} := G\underset{\sim}{k} + \underset{\sim}{k} - (I - G^2)\underset{\sim}{y}^{(0)} ,$$

$$\rho_0 := \frac{(T\underset{\sim}{w}^{(0)}, T\underset{\sim}{w}^{(0)})}{(T\underset{\sim}{w}^{(0)}, T(I - G^2)\underset{\sim}{w}^{(0)})}$$

and

$$\underset{\sim}{y}^{(1)} := \rho_0 \underset{\sim}{w}^{(0)} + \underset{\sim}{y}^{(0)} .$$

Let $\gamma_1 := 1$ and $i := 0$.

(2) Repeat until $\underset{\sim}{w}^{(i)} = 0$:

(a) Let

$$i := i + 1.$$

(b) Compute

$$\underset{\sim}{w}^{(i)} := G\underset{\sim}{k} + \underset{\sim}{k} - (I - G^2)\underset{\sim}{y}^{(i)} ,$$

(c) Compute

$$\rho_i := \frac{(T\underset{\sim}{w}^{(i)},\ T\underset{\sim}{w}^{(i)})}{(T\underset{\sim}{w}^{(i)},\ T(I - G^2)\ \underset{\sim}{w}^{(i)})} \ ,$$

$$\gamma_{i+1} := \left[1 - \frac{\rho_i}{\rho_{i-1}} \frac{(T\underset{\sim}{w}^{(i)},\ T\underset{\sim}{w}^{(i)})}{(T\underset{\sim}{w}^{(i-1)},\ T\underset{\sim}{w}^{(i-1)})} \ \frac{1}{\gamma_i} \right]^{-1}$$

and

$$\underset{\sim}{y}^{(i+1)} := \gamma_{i+1}[\rho_i \underset{\sim}{w}^{(i)} + \underset{\sim}{y}^{(i)}] + (1 - \gamma_{i+1})\underset{\sim}{y}^{(i-1)} \ . \quad \square$$

We now assume that the condition

$$\beta_i = 1 \qquad \text{for} \quad i = 0,1,\ldots \ , \tag{4.2}$$

holds for the CG-accelerated single method (Algorithm 3.2). The three-term recurrence defining the method thereby simplifies to

$$\underset{\sim}{x}^{(i+1)} = \alpha_{i+1}(G\underset{\sim}{x}^{(i)} + \underset{\sim}{k}) + (1 - \alpha_{i+1})\underset{\sim}{x}^{(i-1)} \quad \text{for} \quad i = 0,1,\ldots \ , \tag{4.3}$$

where $\alpha_1 = 1$. Replacing i by $2j + 1$ in (4.3), we get

$$\underset{\sim}{x}^{(2j+2)} = \alpha_{2j+2}(G\underset{\sim}{x}^{(2j+1)} + \underset{\sim}{k}) + (1 - \alpha_{2j+2})\underset{\sim}{x}^{(2j)} \ .$$

Also from (4.3), we obtain

$$G\underset{\sim}{x}^{(2j+1)} = \alpha_{2j+1}(G^2\underset{\sim}{x}^{(2j)} + G\underset{\sim}{k}) + (1 - \alpha_{2j+1})\ G\underset{\sim}{x}^{(2j-1)}$$

and

$$G\underset{\sim}{x}^{(2j-1)} = \frac{1}{\alpha_{2j}}\ \underset{\sim}{x}^{(2j)} - \left(\frac{1 - \alpha_{2j}}{\alpha_{2j}} \right) \underset{\sim}{x}^{(2j-2)} - \underset{\sim}{k} \ .$$

It follows that

$$\underset{\sim}{x}^{(2j+2)} = \alpha_{2j+2}\alpha_{2j+1}(G^2\underset{\sim}{x}^{(2j)} + G\underset{\sim}{k} + \underset{\sim}{k})$$

$$+ \left[1 + \frac{\alpha_{2j+2}}{\alpha_{2j}}(1 - \alpha_{2j+1}) - \alpha_{2j+2}\right]\underset{\sim}{x}^{(2j)}$$

$$- \frac{\alpha_{2j+2}}{\alpha_{2j}}(1 - \alpha_{2j+1})(1 - \alpha_{2j})\,\underset{\sim}{x}^{(2j-2)} \quad . \qquad (4.4)$$

We use (4.4) to define one more iterative scheme.

ALGORITHM 4.2. (Derived CG-accelerated double method)

(1) Let

$$\underset{\sim}{v}^{(0)} := \underset{\sim}{x}^{(0)} \quad .$$

Compute

$$\hat{\underset{\sim}{w}}^{(0)} := G\underset{\sim}{k} + \underset{\sim}{k} - (I - G^2)\,\underset{\sim}{v}^{(0)} \quad ,$$

$$\hat{\rho}_0 := \alpha_2\alpha_1$$

and

$$\underset{\sim}{v}^{(1)} := \hat{\rho}_0\,\hat{\underset{\sim}{w}}^{(0)} + \underset{\sim}{v}^{(0)} \quad .$$

Let

$$i := 0 \quad .$$

(2) Repeat until $\hat{\underset{\sim}{w}}^{(i)} = \underset{\sim}{0}$:

(a) Let

$$i := i + 1 \quad .$$

(b) Compute

$$\hat{\underset{\sim}{w}}^{(i)} := G\underset{\sim}{k} + \underset{\sim}{k} - (I - G^2)\,\underset{\sim}{v}^{(i)} \quad .$$

(c) Compute

$$\hat{\gamma}_{i+1} := 1 + \frac{\alpha_{2i+2}}{\alpha_{2i}}(1 - \alpha_{2i+1})(1 - \alpha_{2i}) ,$$

$$\hat{\rho}_i := \frac{\alpha_{2i+2}\,\alpha_{2i+1}}{\hat{\gamma}_{i+1}}$$

and

$$\underset{\sim}{y}^{(i+1)} := \hat{\gamma}_{i+1}[\hat{\rho}_i\,\hat{\underset{\sim}{w}}^{(i)} + \underset{\sim}{y}^{(i)}] + (1 - \hat{\gamma}_{i+1})\,\underset{\sim}{y}^{(i-1)} . \quad \square$$

We observe that Algorithms 4.1 and 4.2 are identical in form. Our goal is to prove that they are mathematically equivalent, i.e.

$$\underset{\sim}{x}^{(i)} = \underset{\sim}{y}^{(i)} \qquad \text{for} \quad i = 0,1,\ldots .$$

Because of Theorem 3.1, we only have to show that

$$(T\hat{\underset{\sim}{w}}^{(i)},\ T\hat{\underset{\sim}{w}}^{(j)}) = 0 \quad \text{for} \quad i \neq j ,$$

where

$$\hat{\underset{\sim}{w}}^{(i)} = G\underset{\sim}{k} + \underset{\sim}{k} - (I - G^2)\,\underset{\sim}{y}^{(i)} \quad \text{for} \quad i = 0,1,\ldots .$$

Let us define the spectral radius of an $n \times n$ matrix M by the number

$$\rho(M) \equiv \max_{1 \leq i \leq n} |\lambda_i| ,$$

where $\lambda_1, \ldots, \lambda_n$ are the eigenvalues of M. We make the assumption:

ASSUMPTION 3. $\qquad \rho(G) < 1 .$

The matrix $(I + G)$ is therefore nonsingular because all its eigenvalues are bounded away from zero. We now choose the "symmetrization" matrix T of Assumption 2 as

$$T = S(I + G)^{-1} , \qquad (4.5)$$

where S is the "symmetrization" matrix of Assumption 1. It remains to verify that the matrix $B \equiv T(I - G^2)T^{-1}$ is symmetric and positive definite. As

$$T(I - G^2)T^{-1} = I - (SGS^{-1})^2 ,$$

the matrix B is symmetric. Hence the eigenvalues of B are real. By Assumption 3, the eigenvalues of (SGS^{-1}) are all less than unity in modulus. Therefore, all the eigenvalues of B are positive.

Since the derived CG-accelerated double method has been constructed directly from the CG-accelerated single method, we have that

$$\underset{\sim}{y}^{(i)} = \underset{\sim}{x}^{(2i)} \qquad \text{for} \quad i = 0,1,\ldots .$$

Now,

$$\hat{\underset{\sim}{w}}^{(i)} = G\underset{\sim}{k} + \underset{\sim}{k} - (I - G^2)\,\underset{\sim}{x}^{(2i)}$$

$$= (I + G)\,\underset{\sim}{z}^{(2i)} \qquad \text{for} \quad i = 0,1,\ldots ,$$

where

$$\underset{\sim}{z}^{(2i)} \equiv \underset{\sim}{k} - C^{-1}A\underset{\sim}{x}^{(2i)} \qquad \text{for} \quad i = 0,1,\ldots . \qquad (3.3)$$

Thus,

$$T\hat{\underset{\sim}{w}}^{(i)} = S\underset{\sim}{z}^{(2i)} .$$

But from Theorem 3.1, we see that

$$(S\underset{\sim}{z}^{(2i)},\ S\underset{\sim}{z}^{(2j)}) = 0 \qquad \text{for} \quad i \neq j .$$

We have thereby proved the desired equivalence result.

THEOREM 4.1. The CG-accelerated single method and the CG-accelerated double method are "virtually-equivalent" if the iteration parameters $\{\beta_i\}$ of the single method satisfy

$$\beta_i = 1 \qquad \text{for} \quad i = 0,1,\ldots ,$$

and if the "symmetrization" matrices S and T of Assumptions 1 and 2 are chosen so that

$$T = S(I + G)^{-1} .$$

Note that another possibility for the "symmetrization" matrix T is

$$T = S ,$$

but then the above "virtual-equivalence" result no longer holds.

2.5. Block Jacobi Methods

In this and the ensuing section, we consider symmetric and positive definite matrices of the form

$$A = \left(\begin{array}{c|c} M_1 & F \\ \hline F^t & M_2 \end{array} \right), \tag{5.1}$$

where M_1 and M_2 are square matrices such that the systems

$$M_1 \underset{\sim}{z}_1 = \underset{\sim}{d}_1 \qquad \text{and} \qquad M_2 \underset{\sim}{z}_2 = \underset{\sim}{d}_2$$

are easy to solve. Many matrices can be described by (5.1), e.g. those matrices that possess "Property A" [11].

The block Jacobi method corresponds to the preconditioning matrix

$$C = \left(\begin{array}{c|c} M_1 & 0 \\ \hline 0 & M_2 \end{array}\right) . \tag{5.2}$$

As both matrices M_1 and M_2 are symmetric and positive definite, it follows that C is, too. We have

$$G = \left(\begin{array}{c|c} 0 & -M_1^{-1}F \\ \hline -M_2^{-1}F^t & 0 \end{array}\right) \tag{5.3}$$

and

$$\underset{\sim}{k} = \begin{pmatrix} M_1^{-1}\underset{\sim}{b}_1 \\ M_2^{-1}\underset{\sim}{b}_2 \end{pmatrix} . \tag{5.4}$$

Our goal is to show that the CG-accelerated single block Jacobi method (J-CG method) and the CG-accelerated double block Jacobi method (DJ-CG method) are "virtually-equivalent."

There exist matrices S_1 and S_2 such that

$$S_1^t S_1 = M_1 \qquad \text{and} \qquad S_2^t S_2 = M_2 . \tag{5.5}$$

We choose the "symmetrization" matrix S as

$$S = \left(\begin{array}{c|c} S_1 & 0 \\ \hline 0 & S_2 \end{array}\right) \tag{5.6}$$

All is well because $S^t S = C$ (see Lemma 1.1). Also, let

$$B \equiv SC^{-1}AS^{-1}.$$

Since

$$B = \left(\begin{array}{c|c} I & (S_1^{-1})^t F S_2^{-1} \\ \hline (S_2^{-1})^t F^t S_1^{-1} & I \end{array} \right),$$

its eigenvalues are either unity or of the form $1 \pm \sigma^2$, where σ is some nonzero real number. As the matrix B is positive definite, it follows that $\sigma^2 < 1$. Now,

$$G = I - SBS^{-1} ,$$

and so the eigenvalues of G are either zero or of the form $\pm \sigma^2$. Hence

$$\rho(G) < 1 ,$$

i.e. Assumption 3 is valid. We can therefore choose the matrix $S(I + G)^{-1}$ as the "symmetrization" matrix for the double method.

Finally, we must show that

$$\beta_i = 1 \qquad \text{for} \quad i = 0,1,\ldots , \tag{4.2}$$

for the J-CG method. We write (1.1) as

$$\left(\begin{array}{c|c} M_1 & F \\ \hline F^t & M_2 \end{array} \right) \left(\begin{array}{c} \underset{\sim}{x}_1 \\ \underset{\sim}{x}_2 \end{array} \right) = \left(\begin{array}{c} \underset{\sim}{b}_1 \\ \underset{\sim}{b}_2 \end{array} \right) . \tag{5.7}$$

Following Concus et al. [2], we choose the initial vector

$$\underset{\sim}{x}^{(0)} = \begin{cases} \underset{\sim}{x}_1^{(0)} & \text{is some approximation to } \underset{\sim}{x}_1 \\ \underset{\sim}{x}_2^{(0)} = M_2^{-1}(\underset{\sim}{b}_2 - F^t \underset{\sim}{x}_1^{(0)}) , \end{cases} \tag{5.8}$$

so that

$$\underset{\sim}{z}_2^{(0)} = \underset{\sim}{0} .$$

As

$$\beta_i = \frac{(\underset{\sim}{z}^{(i)}, C\underset{\sim}{z}^{(i)})}{(\underset{\sim}{z}^{(i)}, A\underset{\sim}{z}^{(i)})} \qquad \text{for } i = 0,1,\ldots, \text{ (Algorithm 3.2)}$$

we see that

$$\beta_0 = 1 .$$

From the recurrence relation

$$\underset{\sim}{x}^{(i+1)} = \alpha_{i+1}[\beta_i(G\underset{\sim}{x}^{(i)} + \underset{\sim}{k}) + (1-\beta_i)\underset{\sim}{x}^{(i)}] + (1-\alpha_{i+1})\,\underset{\sim}{x}^{(i-1)} ,$$

$$\text{for } i = 0,1,\ldots , \qquad (1.9)$$

where $\alpha_1 = 1$, we get (cf. recurrence relation (2.5))

$$\underset{\sim}{z}^{(i+1)} = \alpha_{i+1}[\beta_i(I - C^{-1}A)\underset{\sim}{z}^{(i)} + (1-\beta_i)\underset{\sim}{z}^{(i)}] + (1-\alpha_{i+1})\underset{\sim}{z}^{(i-1)},$$

or equivalently,

$$\begin{pmatrix} \underset{\sim}{z}_1^{(i+1)} \\ \underset{\sim}{z}_2^{(i+1)} \end{pmatrix} = \alpha_{i+1}\left[\beta_i \left(\begin{array}{c|c} 0 & -M_1^{-1}F \\ \hline -M_2^{-1}F^t & 0 \end{array}\right)\begin{pmatrix} \underset{\sim}{z}_1^{(i)} \\ \underset{\sim}{z}_2^{(i)} \end{pmatrix} + (1-\beta_i)\begin{pmatrix} \underset{\sim}{z}_1^{(i)} \\ \underset{\sim}{z}_2^{(i)} \end{pmatrix}\right]$$

$$+ (1+\alpha_{i+1})\begin{pmatrix} \underset{\sim}{z}_1^{(i-1)} \\ \underset{\sim}{z}_2^{(i-1)} \end{pmatrix} \qquad \text{for } i = 0,1,\ldots . \qquad (5.9)$$

It follows that

$$\underset{\sim}{z}_1^{(1)} = \underset{\sim}{0} .$$

Thus,

$$\beta_1 = 1 .$$

Using (5.9), we readily prove by induction that

$$\begin{cases} \underset{\sim}{z}_2^{(2i)} = \underset{\sim}{0} \quad \text{and} \quad \beta_{2i} = 1 , \\ \underset{\sim}{z}_1^{(2i+1)} = \underset{\sim}{0} \quad \text{and} \quad \beta_{2i+1} = 1 , \end{cases} \qquad \text{for} \quad i = 0,1,\ldots . \tag{5.10}$$

THEOREM 5.1. The J-CG method with "symmetrization" matrix S defined by (5.6) and the DJ-CG method with "symmetrization" matrix $S(I + G)^{-1}$ are "virtually-equivalent" if the initial vector is chosen by (5.8).

2.6. A Compressed Method and Cyclic Reduction

In this section, we first give two special methods for solving the system

$$\left(\begin{array}{c|c} M_1 & F \\ \hline F^t & M_2 \end{array}\right)\left(\begin{array}{c} \underset{\sim}{x}_1 \\ \underset{\sim}{x}_2 \end{array}\right) = \left(\begin{array}{c} \underset{\sim}{b}_1 \\ \underset{\sim}{b}_2 \end{array}\right) . \tag{5.7}$$

and then use the result in the last section to show that the two methods are equivalent.

The first method is based on a procedure introduced by Reid [9]. We consider the J-CG method with the special initial vector

$$\underset{\sim}{x}^{(0)} = \begin{cases} \underset{\sim}{x}_1^{(0)} \\ M_2^{-1}(\underset{\sim}{b}_2 - F^t \underset{\sim}{x}_1^{(0)}) \end{cases} . \tag{5.8}$$

We have seen that

$$\beta_i = 1 \qquad \text{for} \quad i = 0,1,\ldots . \tag{4.2}$$

Thus,

$$\underset{\sim}{x}^{(2i+2)} = \alpha_{2i+2}\alpha_{2i+1}(G^2\underset{\sim}{x}^{(2i)} + G\underset{\sim}{k} + \underset{\sim}{k}) + \left[1 + \frac{\alpha_{2i+2}}{\alpha_{2i}}(1-\alpha_{2i+1}) - \alpha_{2i+2}\right]\underset{\sim}{x}^{(2i)}$$

$$- \frac{\alpha_{2i+2}}{\alpha_{2i}}(1 - \alpha_{2i+1})(1 - \alpha_{2i})\underset{\sim}{x}^{(2i-2)}$$

$$\text{for } i = 0,1,\ldots . \qquad (4.4)$$

Since

$$G^2\underset{\sim}{x}^{(2i)} + G\underset{\sim}{k} + \underset{\sim}{k} - \underset{\sim}{x}^{(2i)} = (I + G)\,\underset{\sim}{z}^{(2i)} ,$$

we can simplify (4.4) to

$$\underset{\sim}{x}^{(2i+2)} - \underset{\sim}{x}^{(2i)}$$

$$= \alpha_{2i+2}\alpha_{2i+1}\left[(I + G)\,\underset{\sim}{z}^{(2i)} + \left(1 - \frac{1}{\alpha_{2i+1}}\right)\left(1 - \frac{1}{\alpha_{2i}}\right)(\underset{\sim}{x}^{(2i)} - \underset{\sim}{x}^{(2i-2)})\right],$$

$$\text{for } i = 0,1,\ldots . \qquad (6.1)$$

To save work, we update only $\underset{\sim}{x}_1^{(2i)}$ in (6.1). If we desire $\underset{\sim}{x}_2^{(2i)}$, we solve the system

$$M_2\underset{\sim}{x}_2^{(2i)} = \underset{\sim}{b}_2 - F^t\underset{\sim}{x}_1^{(2i)} ,$$

as $\underset{\sim}{z}_1^{(2i)} = \underset{\sim}{0}$. We update the pseudo-residual vector $\underset{\sim}{z}^{(i)}$ recursively by the formula of (5.9)

$$\begin{cases} \underset{\sim}{z}_2^{(2i+1)} = \alpha_{2i+1}M_2^{-1}F^t\underset{\sim}{z}_1^{(2i)} + (1 - \alpha_{2i+1})\,\underset{\sim}{z}_2^{(2i-1)} , \\ \underset{\sim}{z}_1^{(2i+2)} = \alpha_{2i+2}M_1^{-1}F\,\underset{\sim}{z}_2^{(2i+1)} + (1 - \alpha_{2i+2})\,\underset{\sim}{z}_1^{(2i)} , \end{cases} \quad \text{for } i = 0,1,\ldots ,$$

where $\alpha_1 = 1$. We can now present what we call the compressed J-CG method (CJ-CG method).

ALGORITHM 6.1. (CJ-CG method)

(1) Let $\underset{\sim}{x}_1^{(0)}$ be an initial approximation to $\underset{\sim}{x}_1$. Solve the system

$$M_2\underset{\sim}{x}_2^{(0)} = \underset{\sim}{b}_2 - F^t\underset{\sim}{x}_1^{(0)} .$$

Compute

$$\hat{z}_1^{(0)} := b_1 - M_1 x_1^{(0)} - F x_2^{(0)}$$

and solve the system

$$M_1 z_1^{(0)} = \hat{z}_1^{(0)} .$$

Let

$$i := -1$$

and let $x_1^{(-2)}$ be an vector not equal to $x_1^{(0)}$.

(2) Repeat until $(x_1^{(2i+2)} - x_1^{(2i)}) = 0$:

(a) Let

$$i := i + 1 .$$

(b) Compute

$$\alpha_{2i+1} := \begin{cases} 1 , & i = 0 , \\ \left[1 - \dfrac{(z_1^{(2i)}, \hat{z}_1^{(2i)})}{(z_2^{(2i-1)}, \hat{z}_2^{(2i-1)})} \right]^{-1} , & i \geq 1 , \end{cases}$$

and

$$\hat{z}_2^{(2i+1)} := \alpha_{2i+1} F^t z_1^{(2i)} + (1 - \alpha_{2i+1}) \hat{z}_2^{(2i-1)} .$$

Solve the system

$$M_2 z_2^{(2i+1)} = \hat{z}_2^{(2i+1)} .$$

(c) Compute

$$\alpha_{2i+2} := \left[1 - \frac{(z_2^{(2i+1)}, \hat{z}_2^{(2i+1)})}{(z_1^{(2i)}, \hat{z}_1^{(2i)})} \frac{1}{\alpha_{2i+1}} \right]^{-1}$$

and

$$\hat{z}_1^{(2i+2)} := \alpha_{2i+2} F z_2^{(2i+1)} + (1 - \alpha_{2i+2}) \hat{z}_1^{(2i)} .$$

Solve the system

$$M_1 z_1^{(2i+2)} = \hat{z}_1^{(2i+2)} .$$

(d) Compute

$$(\underset{\sim}{x}_1^{(2i+2)} - \underset{\sim}{x}_1^{(2i)})$$

$$:= \alpha_{2i+2}\alpha_{2i+1}\left[\underset{\sim}{z}_1^{(2i)} + \left(1 - \frac{1}{\alpha_{2i+1}}\right)\left(1 - \frac{1}{\alpha_{2i}}\right)(\underset{\sim}{x}_1^{(2i)} - \underset{\sim}{x}_1^{(2i-2)})\right]$$

and

$$\underset{\sim}{x}_1^{(2i+2)} := \underset{\sim}{x}_1^{(2i)} + (\underset{\sim}{x}_1^{(2i+2)} - \underset{\sim}{x}_1^{(2i)}) .$$

(3) Let $\underset{\sim}{x}_1^{(2k)}$ be the accepted approximation to $\underset{\sim}{x}_1$. Solve the system

$$M_2\underset{\sim}{x}_2^{(2k)} = \underset{\sim}{b}_2 - F^t\underset{\sim}{x}_1^{(2k)} . \qquad \square$$

Let us turn our attention to the double method. It can be viewed as the single method for the matrix equation

$$(I - G^2)\underset{\sim}{x} = G\underset{\sim}{k} + \underset{\sim}{k} . \tag{4.1}$$

As

$$I - G^2 = \left(\begin{array}{c|c} I - M_1^{-1}FM_2^{-1}F^t & 0 \\ \hline 0 & I - M_2^{-1}F^tM_1^{-1}F \end{array}\right) ,$$

the system (4.1) is the uncoupled equations

$$\begin{cases} (I - M_1^{-1}F\,M_2^{-1}F^t)\underset{\sim}{x}_1 = M_1^{-1}\underset{\sim}{b}_1 - M_1^{-1}F\,M_2^{-1}\underset{\sim}{b}_2 & (6.2a) \\ (I - M_2^{-1}F^tM_1^{-1}F)\,\underset{\sim}{x}_2 = M_2^{-1}\underset{\sim}{b}_2 - M_2^{-1}F^tM_1^{-1}\underset{\sim}{b}_1 . & (6.2b) \end{cases}$$

We refer to the equations of (6.2) as the cyclic reduction (cf. [11]) of the matrix equation

$$\left(\begin{array}{c|c} M_1 & F \\ \hline F^t & M_2 \end{array}\right)\begin{pmatrix} \underset{\sim}{x}_1 \\ \underset{\sim}{x}_2 \end{pmatrix} = \begin{pmatrix} \underset{\sim}{b}_1 \\ \underset{\sim}{b}_2 \end{pmatrix} \quad . \tag{5.7}$$

In the last section, we have chosen the matrix $S(I + G)^{-1}$ as the "symmetrization" matrix for $(I - G^2)$. As

$$[S(I + G)^{-1}]\ (I - G^2)[S(I + G)^{-1}]^{-1} = S(I - G^2)S^{-1}$$

and

$$S = \left(\begin{array}{c|c} S_1 & 0 \\ \hline 0 & S_2 \end{array}\right) \quad , \tag{5.6}$$

we see that the matrix

$$S_1(I - M_1^{-1}FM_2^{-1}F^t)S_1^{-1}$$

is symmetric and positive definite.

An effective procedure for solving the matrix equation (5.7) consists of

(a) applying the CG-accelerated single method to solve the reduced system (6.2a) with "symmetrization" matrix S_1, and

(b) solving the system

$$M_2\underset{\sim}{x}_2^{(k)} = \underset{\sim}{b}_2 - F^t\underset{\sim}{x}_1^{(k)} \ ,$$

where $\underset{\sim}{x}_1^{(k)}$ is the accepted approximation to $\underset{\sim}{x}_1$ in (a).

We call this procedure the RS-CG method.

ALGORITHM 6.2 (RS-CG method)

(1) Let

$$\underset{\sim}{y}_1^{(0)} := \underset{\sim}{x}_1^{(0)} .$$

Solve the system

$$M_1 \underset{\sim}{g}^{(0)} = \underset{\sim}{b}_1 - F M_2^{-1} \underset{\sim}{b}_2 - (M_1 - F M_2^{-1} F^t) \underset{\sim}{y}_1^{(0)} .$$

Compute

$$\rho_0 := \frac{(\underset{\sim}{g}^{(0)}, M_1 \underset{\sim}{g}^{(0)})}{(\underset{\sim}{g}^{(0)}, (M_1 - F M_2^{-1} F^t) \underset{\sim}{g}^{(0)})}$$

and

$$\underset{\sim}{y}_1^{(1)} := \rho_0 \underset{\sim}{g}^{(0)} + \underset{\sim}{y}_1^{(0)} .$$

Let $\gamma_1 := 1$ and $i := 0$.

(2) Repeat until $\underset{\sim}{g}^{(0)} = \underset{\sim}{0}$:

(a) Let

$$i := i + 1.$$

(b) Solve the system

$$M_1 \underset{\sim}{g}_1^{(i)} = \underset{\sim}{b}_1 - F M_2^{-1} \underset{\sim}{b}_2 - (M_1 - F M_2^{-1} F^t) \underset{\sim}{y}_1^{(i)} .$$

(c) Compute

$$\rho_i := \frac{(\underset{\sim}{g}^{(i)}, M_1 \underset{\sim}{g}^{(i)})}{(\underset{\sim}{g}^{(i)}, (M_1 - F M_2^{-1} F^t) \underset{\sim}{q}^{(i)})} ,$$

$$\gamma_{i+1} := \left[1 - \frac{\rho_i}{\rho_{i-1}} \frac{(\underset{\sim}{g}^{(i)}, M_1 \underset{\sim}{g}^{(i)})}{(\underset{\sim}{g}^{(i-1)}, M_1 \underset{\sim}{g}^{(i-1)})} \frac{1}{\gamma_i} \right]^{-1}$$

and

$$\underset{\sim}{y}_1^{(i+1)} := \gamma_{i+1}[\rho_i \underset{\sim}{g}^{(i)} + \underset{\sim}{y}_1^{(i)}] + (1 - \gamma_{i+1})\underset{\sim}{y}_1^{(i-1)} .$$

(3) Let $\underset{\sim}{y}_1^{(k)}$ be the accepted approximation to $\underset{\sim}{x}_1$. Solve the system

$$M_2 \underset{\sim}{y}_2^{(k)} = \underset{\sim}{b}_2 - F^t \underset{\sim}{y}_1^{(k)} . \qquad \square$$

We proceed to show how we can derive the RS-CG method from the DJ-CG method. Theorem 5.1 states that the J-CG and DJ-CG methods are "virtually-equivalent," i.e.

$$\underset{\sim}{x}^{(2i)} = \underset{\sim}{y}^{(i)} \qquad \text{for} \quad i = 0,1,\ldots .$$

Also we know that

$$\underset{\sim}{z}_2^{(2i)} = \underset{\sim}{0} \qquad \text{for} \quad i = 0,1,\ldots . \tag{5.10}$$

Now, let us use the DJ-CG method to update $\underset{\sim}{y}_1^{(i)}$:

$$\underset{\sim}{y}_1^{(i+1)} = \gamma_{i+1}(\rho_i \underset{\sim}{w}_1^{(i)} + \underset{\sim}{y}_1^{(i)}) + (1 - \gamma_{i+1})\underset{\sim}{y}_1^{(i-1)} \quad \text{for} \quad i = 0,1,\ldots, \tag{6.3}$$

where $\gamma_1 = 1$. As

$$\begin{aligned} \underset{\sim}{w}^{(i)} &= (I + G)[\underset{\sim}{k} - (I - G)\underset{\sim}{y}^{(i)}] \\ &= (I + G)[\underset{\sim}{k} - (I - G)\underset{\sim}{x}^{(2i)}] \\ &= (I + G)\underset{\sim}{z}^{(2i)} , \end{aligned}$$

we get

$$\underset{\sim}{w}_1^{(i)} = \underset{\sim}{z}_1^{(2i)} \qquad \text{for} \quad i = 0,1,\ldots . \tag{6.4}$$

Let

$$H \equiv I - M_1^{-1}FM_2^{-1}F^t \tag{6.5}$$

and

$$\underset{\sim}{h} \equiv M_1^{-1}\underset{\sim}{b}_1 - M_1^{-1}FM_2^{-1}\underset{\sim}{b}_2 . \tag{6.6}$$

Then

$$\begin{aligned}\underset{\sim}{z}_1^{(2i)} &= M_1^{-1}\underset{\sim}{b}_1 - \underset{\sim}{x}_1^{(2i)} - M_1^{-1}F\underset{\sim}{x}_2^{(2i)} \\ &= M_1^{-1}\underset{\sim}{b}_1 - \underset{\sim}{x}_1^{(2i)} - M_1^{-1}FM_2^{-1}(\underset{\sim}{b}_2 - F^t\underset{\sim}{x}_1^{(2i)}) \\ &= \underset{\sim}{h} - (I - H)\underset{\sim}{x}_1^{(2i)} .\end{aligned}$$

We define the residual vector $\underset{\sim}{g}^{(i)}$ for the reduced system (6.2a) by

$$\underset{\sim}{g}^{(i)} \equiv \underset{\sim}{h} - (I - H)\underset{\sim}{y}_1^{(i)} \qquad \text{for} \quad i = 0,1,\ldots . \tag{6.7}$$

Thus,

$$\underset{\sim}{g}^{(i)} = \underset{\sim}{z}_1^{(2i)} = \underset{\sim}{w}_1^{(i)} \qquad \text{for} \quad i = 0,1,\ldots , \tag{6.8}$$

so that the recurrence relation (6.3) becomes

$$\underset{\sim}{x}_1^{(i+1)} = \gamma_{i+1}(\rho_i\underset{\sim}{g}^{(i)} + \underset{\sim}{y}_1^{(i)}) + (1 - \gamma_{i+1})\underset{\sim}{x}^{(i-1)}$$

$$\text{for} \quad i = 0,1,\ldots , \tag{6.9}$$

where $\gamma_1 = 1$. Now,

$$\begin{aligned}(T\underset{\sim}{w}^{(i)}, T\underset{\sim}{w}^{(i)}) &= (S\underset{\sim}{z}^{(2i)}, S\underset{\sim}{z}^{(2i)}) \\ &= (S_1\underset{\sim}{z}_1^{(2i)}, S_1\underset{\sim}{z}_1^{(2i)}) \\ &= (\underset{\sim}{g}^{(i)}, M_1\underset{\sim}{g}^{(i)})\end{aligned}$$

and

$$(T\underset{\sim}{w}^{(i)},\ T(I-G^2)\underset{\sim}{w}^{(i)}) = (S\underset{\sim}{z}^{(2i)},\ S(I-G^2)\underset{\sim}{z}^{(2i)})$$
$$= (\underset{\sim}{g}^{(i)},\ M_1(I-H)\underset{\sim}{g}^{(i)}) .$$

Therefore, the iteration parameters $\{\rho_i\}$ and $\{\gamma_i\}$ in the DJ-CG method can be given by

$$\rho_i = \frac{(\underset{\sim}{g}^{(i)},\ M_1\underset{\sim}{g}^{(i)})}{(\underset{\sim}{g}^{(i)},\ M_1(I-H)\underset{\sim}{g}^{(i)})} , \qquad i \geq 0 , \tag{6.10}$$

and

$$\gamma_{i+1} = \begin{cases} 1 , & i = 0 , \\ \left[1 - \dfrac{\rho_i}{\rho_{i-1}} \dfrac{(\underset{\sim}{g}^{(i)},\ M_1\underset{\sim}{g}^{(i)})}{(\underset{\sim}{g}^{(i-1)},\ M_1\underset{\sim}{g}^{(i-1)})} \dfrac{1}{\gamma_i}\right]^{-1} , & i \geq 1. \end{cases} \tag{6.11}$$

The iterative scheme (6.9) with its iteration parameters given by (6.10) and (6.11) is a second-order procedure for solving the reduced system (6.2a). Since

$$(S_1\underset{\sim}{g}^{(i)},\ S_1\underset{\sim}{g}^{(j)}) = (S_1\underset{\sim}{z}_1^{(2i)},\ S_1\underset{\sim}{z}_1^{(2j)})$$
$$= (S\underset{\sim}{z}^{(2i)},\ S\underset{\sim}{z}^{(2j)})$$
$$= 0 \qquad \text{for } i \neq j,$$

we conclude that the procedure is identical to the CG-accelerated single method (Theorem 2.1).

We have therefore derived the CJ-CG method from the J-CG method, and the RS-CG method from the DJ-CG method. As the J-CG and the DJ-CG methods are "virtually-equivalent" by Theorem 5.1, we have proved our desired result:

THEOREM 6.1. The CJ-CG and the RS-CG methods are "virtually-equivalent."

2.7. Computer Implementation

There are two popular versions of the conjugate gradient method. One is the original two-term version due to Hestenes and Stiefel [6], the other is a three-term recurrence version due to Rutishauser [10]. We have given our algorithms in the second version for expository purposes. In [8], Reid compared the two versions and found that the Hestenes and Stiefel version is more efficient in both storage and computational work.

Let us present the Hestenes and Stiefel version of the conjugate gradient method for the system

$$A\underset{\sim}{x} = \underset{\sim}{b} , \tag{1.1}$$

where the matrix A is symmetric and positive definite.

ALGORITHM 7.1. (CG method)

(1) Let $\underset{\sim}{x}^{(0)}$ be an initial approximation to $\underset{\sim}{x}$. Compute

$$\underset{\sim}{r}^{(0)} := \underset{\sim}{b} - A\underset{\sim}{x}^{(0)} .$$

Let

$$\underset{\sim}{p}^{(0)} := \underset{\sim}{r}^{(0)} .$$

Compute

$$\hat{\underset{\sim}{p}}^{(0)} := A\underset{\sim}{p}^{(0)} ,$$

$$\sigma_0 := \frac{(\underset{\sim}{r}^{(0)}, \underset{\sim}{r}^{(0)})}{(\underset{\sim}{p}^{(0)}, \hat{\underset{\sim}{p}}^{(0)})}$$

and

$$\underset{\sim}{x}^{(1)} := \underset{\sim}{x}^{(0)} + \sigma_0 \underset{\sim}{p}^{(0)} .$$

Let

$$i := 0 .$$

(2) Repeat until $\underset{\sim}{r}^{(i)} = \underset{\sim}{0}$ or $\underset{\sim}{p}^{(i)} = \underset{\sim}{0}$:

(a) Let

$$i := i + 1.$$

(b) Compute

$$\underset{\sim}{r}^{(i)} := \underset{\sim}{r}^{(i-1)} - \sigma_{i-1} \hat{\underset{\sim}{p}}^{(i-1)}$$

(c) Compute

$$\tau_i := \frac{(\underset{\sim}{r}^{(i)}, \underset{\sim}{r}^{(i)})}{(\underset{\sim}{r}^{(i-1)}, \underset{\sim}{r}^{(i-1)})}$$

and

$$\underset{\sim}{p}^{(i)} := \underset{\sim}{r}^{(i)} + \tau_i \underset{\sim}{p}^{(i-1)} .$$

(d) Compute

$$\hat{\underset{\sim}{p}}^{(i)} := A\underset{\sim}{p}^{(i)} ,$$

$$\sigma_i := \frac{(\underset{\sim}{r}^{(i)}, \underset{\sim}{r}^{(i)})}{(\underset{\sim}{p}^{(i)}, \hat{\underset{\sim}{p}}^{(i)})}$$

and

$$\underset{\sim}{x}^{(i+1)} := \underset{\sim}{x}^{(i)} + \sigma_i \underset{\sim}{p}^{(i)} . \qquad \square$$

Algorithm 7.1 requires one matrix-vector product $A\underset{\sim}{p}$ and $5n$ multiplications per iteration. We have to store four vectors $\underset{\sim}{x}$, $\underset{\sim}{r}$, $\underset{\sim}{p}$ and $A\underset{\sim}{p}$, in addition to the matrix A. The three-term version of the CG method (Algorithm 2.1) requires n more multiplications per iteration and one more vector of storage.

Proceeding in a similar manner as in Section 2.3, we derive from Algorithm 7.1 the two-term version of the preconditioned CG method for the system

$$C^{-1}A\underset{\sim}{x} = C^{-1}\underset{\sim}{b} . \tag{1.2}$$

ALGORITHM 7.2 (Preconditioned CG method)

(1) Let $\underset{\sim}{x}^{(0)}$ be an initial approximation to $\underset{\sim}{x}$. Solve the system

$$C\underset{\sim}{z}^{(0)} = \underset{\sim}{b} - A\underset{\sim}{x}^{(0)} .$$

Let

$$\underset{\sim}{p}^{(0)} := \underset{\sim}{z}^{(0)}$$

and solve the system

$$C\bar{\underset{\sim}{p}}^{(0)} = A\underset{\sim}{p}^{(0)} .$$

Compute

$$\sigma_0 := \frac{(\underset{\sim}{z}^{(0)}, S^t S\underset{\sim}{z}^{(0)})}{(\underset{\sim}{p}^{(0)}, S^t S\bar{\underset{\sim}{p}}^{(0)})}$$

and

$$\underset{\sim}{x}^{(1)} := \underset{\sim}{x}^{(0)} + \sigma_0\underset{\sim}{p}^{(0)} .$$

Let

$$i := 0 .$$

(2) Repeat until $\underset{\sim}{z}^{(i)} = \underset{\sim}{0}$ or $\underset{\sim}{p}^{(i)} = \underset{\sim}{0}$:

(a) Let

$$i := i + 1 .$$

(b) Compute

$$\underset{\sim}{z}^{(i)} := \underset{\sim}{z}^{(i-1)} - \sigma_{i-1}\bar{\underset{\sim}{p}}^{(i-1)}$$

(c) Compute

$$\tau_i := \frac{(\underset{\sim}{z}^{(i)}, S^tS\underset{\sim}{z}^{(i)})}{(\underset{\sim}{z}^{(i-1)}, S^tS\underset{\sim}{z}^{(i-1)})} .$$

(d) Compute

$$\underset{\sim}{p}^{(i)} := \underset{\sim}{z}^{(i)} + \tau_i\underset{\sim}{p}^{(i-1)}$$

and solve the system

$$C\bar{\underset{\sim}{p}}^{(i)} = A\underset{\sim}{p}^{(i)} .$$

(e) Compute

$$\sigma_i := \frac{(\underset{\sim}{z}^{(i)}, S^tS\underset{\sim}{z}^{(i)})}{(\underset{\sim}{p}^{(i)}, S^tS\bar{\underset{\sim}{p}}^{(i)})}$$

and

$$\underset{\sim}{x}^{(i+1)} := \underset{\sim}{x}^{(i)} + \sigma_i\underset{\sim}{p}^{(i)} . \qquad \square$$

An iteration of Algorithm 7.2 requires three matrix-vector products $A\underset{\sim}{p}$, $S^tS\underset{\sim}{z}$ and $S^tS\bar{\underset{\sim}{p}}$, one matrix equation solution with coefficient matrix C, and $5n$ multiplications. Storage is required for the three matrices A, C and S^tS, and for the five vectors $\underset{\sim}{x}$, $\underset{\sim}{z}$, $\underset{\sim}{p}$, $\bar{\underset{\sim}{p}}$ and $S^tS\underset{\sim}{z}$. We can store the vector $S^tS\bar{\underset{\sim}{p}}$ in $S^tS\underset{\sim}{z}$.

In the special instance where $S^tS = C$, Algorithm 7.2 reduces to the generalized CG method of Concus et al. [2]:

ALGORITHM 7.3 (Generalized CG method)

(1) Let $\underset{\sim}{x}^{(0)}$ be an initial approximation to $\underset{\sim}{x}$. Compute

$$\underset{\sim}{r}^{(0)} := \underset{\sim}{b} - A\underset{\sim}{x}^{(0)}$$

and solve the system

$$C\underset{\sim}{z}^{(0)} = \underset{\sim}{r}^{(0)} .$$

Let

$$\underset{\sim}{p}^{(0)} := \underset{\sim}{z}^{(0)} .$$

Compute

$$\sigma_0 := \frac{(\underset{\sim}{z}^{(0)}, \underset{\sim}{r}^{(0)})}{(\underset{\sim}{p}^{(0)}, A\underset{\sim}{p}^{(0)})}$$

and

$$\underset{\sim}{x}^{(1)} := \underset{\sim}{x}^{(0)} + \sigma_0\underset{\sim}{p}^{(0)} .$$

Let

$$i := 0 .$$

(2) Repeat until $\underset{\sim}{z}^{(i)} = \underset{\sim}{0}$ or $\underset{\sim}{p}^{(i)} = \underset{\sim}{0}$:

(a) Let

$$i := i + 1 .$$

(b) Compute

$$\underset{\sim}{r}^{(i)} := \underset{\sim}{r}^{(i-1)} - \sigma_{i-1}A\underset{\sim}{p}^{(i-1)}$$

and solve the system

$$C\underset{\sim}{z}^{(i)} = \underset{\sim}{r}^{(i)} .$$

(c) Compute

$$\tau_i := \frac{(\underset{\sim}{z}^{(i)}, \underset{\sim}{r}^{(i)})}{(\underset{\sim}{z}^{(i-1)}, \underset{\sim}{r}^{(i-1)})} .$$

(d) Compute

$$\underset{\sim}{p}^{(i)} := \underset{\sim}{z}^{(i)} + \tau_i \underset{\sim}{p}^{(i-1)} .$$

(e) Compute

$$\sigma_i := \frac{(\underset{\sim}{z}^{(i)}, \underset{\sim}{r}^{(i)})}{(\underset{\sim}{p}^{(i)}, A\underset{\sim}{p}^{(i)})}$$

and

$$\underset{\sim}{x}^{(i+1)} := \underset{\sim}{x}^{(i)} + \sigma_i \underset{\sim}{p}^{(i)} . \qquad \square$$

Algorithm 7.3 requires one matrix-vector product $A\underset{\sim}{p}$, one matrix equation solution with coefficient matrix C and $5n$ multiplications for each iteration. We have to store the two matrices A and C, and the four vectors $\underset{\sim}{x},, \underset{\sim}{r}$, $\underset{\sim}{p}$ and $A\underset{\sim}{p}$. We can store the vector $\underset{\sim}{z}$ in $A\underset{\sim}{p}$.

We now consider solution procedures for the system

$$\left(\begin{array}{c|c} M_1 & F \\ \hline F^t & M_2 \end{array}\right) \begin{pmatrix} \underset{\sim}{x}_1 \\ \underset{\sim}{x}_2 \end{pmatrix} = \begin{pmatrix} \underset{\sim}{b}_1 \\ \underset{\sim}{b}_2 \end{pmatrix} . \qquad (5.7)$$

The J-CG method is Algorithm 7.3 with the preconditioning matrix

$$C = \left(\begin{array}{c|c} M_1 & 0 \\ \hline 0 & M_2 \end{array}\right) . \qquad (5.2)$$

As the RS-CG method is the CG method applied to the reduced system (6.2a), we can convert it to the two-term form without much difficulty:

Algorithm 7.4. (RS-CG method)

(1) Let $\underset{\sim}{x}_1^{(0)}$ be an initial approximation to $\underset{\sim}{x}_1$. Compute

$$\hat{\underset{\sim}{g}}^{(0)} := \underset{\sim}{b}_1 - FM_2^{-1}\underset{\sim}{b}_2 - (M_1 - FM_2^{-1}F^t)\underset{\sim}{x}_1^{(0)} .$$

Solve the system

$$M_1 \underset{\sim}{g}^{(0)} = \hat{\underset{\sim}{g}}^{(0)} .$$

Let

$$\underset{\sim}{p}_1^{(0)} := \underset{\sim}{g}^{(0)}$$

and

$$\hat{\underset{\sim}{p}}_1^{(0)} := \hat{\underset{\sim}{g}}^{(0)} .$$

Compute

$$\sigma_0 := \frac{(\underset{\sim}{g}^{(0)}, \hat{\underset{\sim}{g}}^{(0)})}{(\underset{\sim}{p}_1^{(0)}, \hat{\underset{\sim}{p}}_1^{(0)} - FM_2^{-1}F^t\underset{\sim}{p}_1^{(0)})}$$

and

$$\underset{\sim}{x}_1^{(1)} := \underset{\sim}{x}_1^{(0)} + \sigma_0 \underset{\sim}{p}_1^{(0)} .$$

Let

$$i := 0 .$$

(2) Repeat until either $p_1^{(i)} = 0$ or $g^{(i)} = 0$:

(a) Let

$$i := i + 1.$$

(b) Compute

$$\hat{g}^{(i)} := \hat{g}^{(i-1)} - \sigma_{i-1}\hat{p}_1^{(i-1)} .$$

(c) Solve the system

$$M_1 g^{(i)} = \hat{g}^{(i)} .$$

(d) Compute

$$\tau_i := \frac{(g^{(i)}, \hat{g}^{(i)})}{(g^{(i-1)}, \hat{g}^{(i-1)})} ,$$

$$p_1^{(i)} := g^{(i)} + \tau_i p_1^{(i-1)}$$

and

$$\hat{p}_1^{(i)} := \hat{g}^{(i)} + \tau_i \hat{p}_1^{(i-1)} .$$

(e) Compute

$$\sigma_i := \frac{(g^{(i)}, \hat{q}^{(i)})}{(p_1^{(i)}, \hat{p}_1^{(i)} - FM_2^{-1}F^t p_1^{(i)})}$$

and

$$y_1^{(i+1)} := y_1^{(i)} + \sigma_i p_1^{(i)} .$$

(3) Let $y_1^{(k)}$ be the accepted approximation to x_1. Solve the system

$$M_2 y_2^{(k)} = b_2 - F^t y_1^{(k)} . \qquad \square$$

An iteration of Algorithm 7.4 requires one matrix-vector product with F and another product with F^t, one matrix equation solution with coefficient matrix M_1 and another solution with M_2, and $6n_1$ multiplications, where n_1 is the order of the vector $\underset{\sim}{x}_1$. We must store the matrix A, the four n_1-vectors $\underset{\sim}{y}_1$, $\hat{\underset{\sim}{q}}$, $\underset{\sim}{p}_1$ and $\hat{\underset{\sim}{p}}_1$, and the two $(n-n_1)$-vectors necessary for computing $FM_2^{-1}F^t\underset{\sim}{p}_1$. The vector $\underset{\sim}{q}$ can be stored with one of the two $(n-n_1)$-vectors.

It is not straightforward to convert the CJ-CG method to the two-term form. Since

$$\underset{\sim}{z}_2^{(2i)} = \underset{\sim}{0} , \tag{5.10}$$

and the method is "virtually-equivalent" to the RS-CG method, we conclude that any two-term version of the CJ-CG method would be essentially Algorithm 7.4. Every two steps of the CJ-CG method (Algorithm 6.1) requires the same number of matrix-vector products and matrix equations solutions as the RS-CG method, plus $3n + 2n_1$ multiplications. Storage is required for the two n-vectors $\underset{\sim}{z}$ and $\hat{\underset{\sim}{z}}$, and for the two n_1-vectors $\underset{\sim}{x}_1$ and $\delta\underset{\sim}{x}_1$, in addition to the matrix A.

In the following table, we summarize the work and storage requirements of the four CG-based methods for solving the matrix equation (5.7). We give the necessary work for one step of each method. As one step of the RS-CG method is equivalent to two steps of the other three algorithms, we halve the work requirements for that method.

Method	Matrix-vector products				Solve		Mult.	Storage
	F	F^t	M_1	M_2	M_1^{-1}	M_2^{-1}		
CG	1	1	1	1	0	0	$5n$	$4n$
J-CG	1	1	1	1	1	1	$5n$	$4n$
CJ-CG	0.5	0.5	0	0	0.5	0.5	$1.5n + n_1$	$3n$
RS-CG	0.5	0.5	0	0	0.5	0.5	$3n_1$	$2n + 2n_1$

It is obvious that the CJ-CG or the RS-CG method is about twice as fast as the J-CG method. In the next section we give three examples showing how a (block) diagonal matrix preconditioning (≡ J-CG method) can greatly accelerate the convergence of the CG method. As $\underset{\sim}{x}_1$ can be chosen so that $n_1 \leq n/2$, the RS-CG method is always more efficient than the CJ-CG method in that about n_1 fewer multiplications are required per iteration.

2.8. Applications of the RS-CG Method

The conjugate gradient method is already well established as an effective iterative solution procedure for large and sparse matrix equations. We wish to show here that the CG method, coupled with cyclic reduction, can be particularly useful for solving certain classes of matrix equations which arise in elliptic partial differential equations problems.

Our presentation here is very similar to that of Hageman [5]. We consider the second-order self-adjoint partial differential equation

$$-(a(x,y)u_x)_x - (a(x,y)u_y)_y + \sigma(x,y)u = f(x,y), \quad (x,y) \in R, \quad (8.1)$$

defined in a bounded rectangular region R and subject to the boundary conditions

$$\frac{\partial u(x,y)}{\partial n} = 0 \quad \text{or} \quad u(x,y) = \text{constant}, \quad (x,y) \in \partial R . \tag{8.2}$$

We assume that the given functions a and σ are continuous in $\bar{R} \equiv R \cup \partial R$ with

$$a(x,y) > 0 \quad \text{and} \quad (x,y) \geq 0 , \qquad (x,y) \in \bar{R} . \tag{8.3}$$

A spatial mesh is then imposed on $\bar{R}$. Discretizing (8.1) and (8.2) with a five-point difference approximation [11, p. 183], we obtain the matrix equation

$$A\underset{\sim}{x} = \underset{\sim}{b} , \tag{1.1}$$

where the $n \times n$ matrix A is symmetric and positive definite.

We divide the mesh points into two sets, one set consisting of square (or black) mesh points and the other consisting of circle (or red) mesh points. The division is done by first making the lower left corner point a circle point and then proceeding by making square points of the four nearest neighbors of the circle points and making circle points of the four nearest neighbors of the square points. We index the mesh points by indexing first all the square points consecutively by rows and then all the circle points consecutively by rows. This ordering is called a point red/black ordering. We give an example in Figure 1.

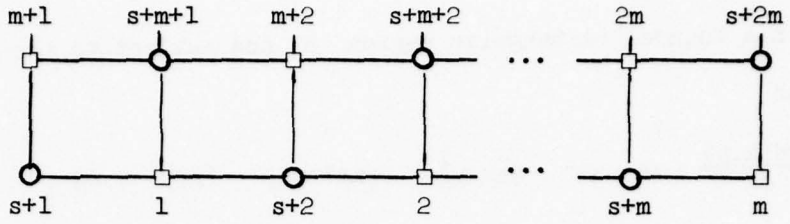


FIGURE 1.

With the point red/black ordering, the matrix equation (1.1) can be written in partitioned form as

$$\left(\begin{array}{c|c} D_1 & H \\ \hline H^t & D_2 \end{array}\right)\begin{pmatrix} \underset{\sim}{x}_1 \\ \underset{\sim}{x}_2 \end{pmatrix} = \begin{pmatrix} \underset{\sim}{b}_1 \\ \underset{\sim}{b}_2 \end{pmatrix}, \tag{8.4}$$

where D_1 and D_2 are diagonal matrices. We have shown in Section 2.6 that the system (8.4) can be cyclically reduced to the lower order matrix equations

$$\begin{cases} (I - D_1^{-1} H D_2^{-1} H^t)\underset{\sim}{x}_1 = D_1^{-1}\underset{\sim}{b}_1 - D_1^{-1} H D_2^{-1}\underset{\sim}{b}_2 , & (8.5a) \\ (I - D_2^{-1} H^t D_1^{-1} H)\underset{\sim}{x}_2 = D_2^{-1}\underset{\sim}{b}_2 - D_2^{-1} H^t D_1^{-1}\underset{\sim}{b}_1 . & (8.5b) \end{cases}$$

The two sets of equations are disjoint, one set involving only the square points and the other involving only the circle points. Let us consider the system (8.5a). Because of our use of the five-point difference approximation, the matrix $(I - D_1^{-1} H D_2^{-1} H^t)$ corresponds to a nine-point difference approximation. Figure 2 illustrates the typical coupling of a (shaded) square mesh point to the eight (darkened) neighboring square mesh points.

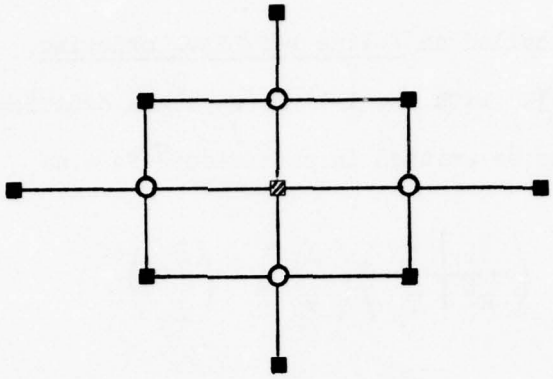


FIGURE 2

We proceed to describe another way to order the mesh points. Assume that there are $p\ell$ horizontal mesh lines. We split the p blocks of successive ℓ horizontal mesh lines into square blocks and circle blocks; we make the first block a circle block, the second one a square block, the third a circle block, and so on. We index the blocks from 1 to p by indexing first all the square blocks consecutively and then all the circle blocks consecutively, as shown in Figure 3.

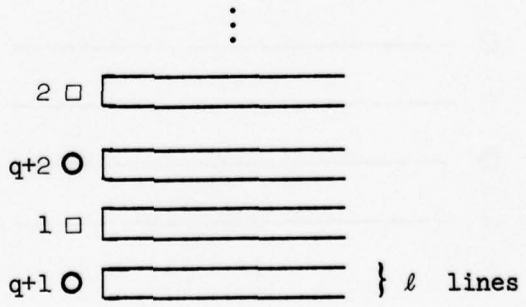


FIGURE 3.

Within each block, the mesh points are indexed consecutively by rows. This ordering is called an <u>ℓ-line red/black ordering</u>.

Let $\ell = 1$. With the 1-line red/black ordering, the matrix equation (1.1) can be written in partitioned form as

$$\left(\begin{array}{c|c} T_1 & H \\ \hline H^t & T_2 \end{array}\right)\begin{pmatrix} \underset{\sim}{x}_1 \\ \underset{\sim}{x}_2 \end{pmatrix} = \begin{pmatrix} \underset{\sim}{b}_1 \\ \underset{\sim}{b}_2 \end{pmatrix}, \tag{8.6}$$

where T_1 and T_2 are tridiagonal matrices. The cyclically reduced systems obtainable from (8.6) are

$$(I - T_1^{-1}H\, T_2^{-1}H^t)\underset{\sim}{x}_1 = T_1^{-1}\underset{\sim}{b}_1 - T_1^{-1}HT_2^{-1}\underset{\sim}{b}_2 , \tag{8.7a}$$

$$(I - T_2^{-1}H^tT_1^{-1}H)\ \underset{\sim}{x}_2 = T_2^{-1}\underset{\sim}{b}_2 - T_2^{-1}H^tT_1^{-1}\underset{\sim}{b}_1 . \tag{8.7b}$$

The system (8.7a) involves only the square mesh lines. Indeed, the matrix $(I - T_1^{-1}HT_2^{-1}H^t)$ corresponds to a three-line difference approximation, as illustrated in Figure 4.

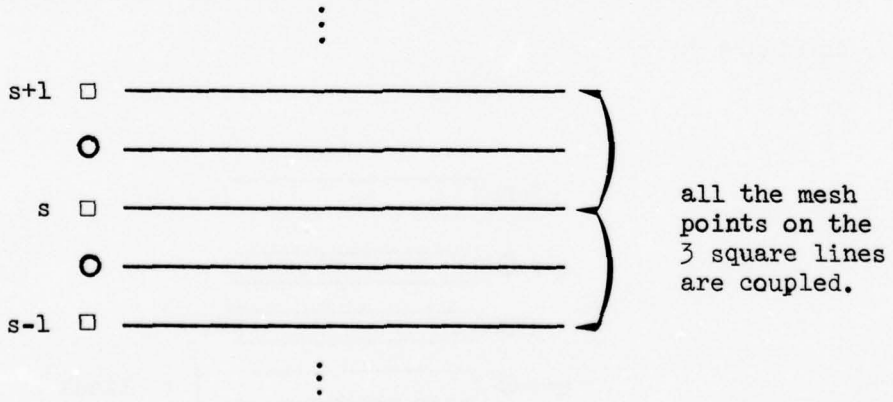
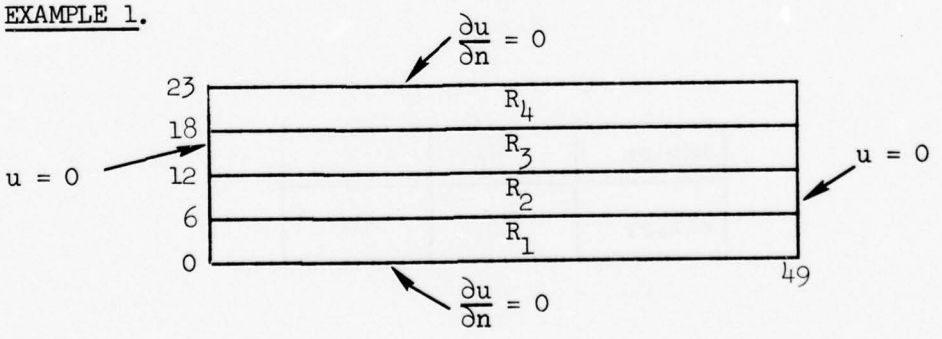


FIGURE 4.

We wish to compare the CG, J-CG and RS-CG methods for the numerical solution of the differential equation (8.1). The CJ-CG method is not included because it is both "virtually-equivalent" to and less efficient than the RS-CG method. We have written three computer programs in the ALGOL W language implementing the three methods. Our programs were run on an IBM 370/168 computer at the Stanford Linear Accelerator Center. The time we give is machine execution time in seconds.

We choose the functions a, σ and f, and the boundary conditions so that the exact solution to the discretized problem is known. Also, we choose the initial vector $\underset{\sim}{x}^{(0)}$ so that each component of the vector is a random number from a uniform distribution in the open interval $(-10^3, 10^3)$. Each method has a different initial vector. The iterative procedure terminates when the maximum norm of the error vector is less than 10^{-3}.

EXAMPLE 1.



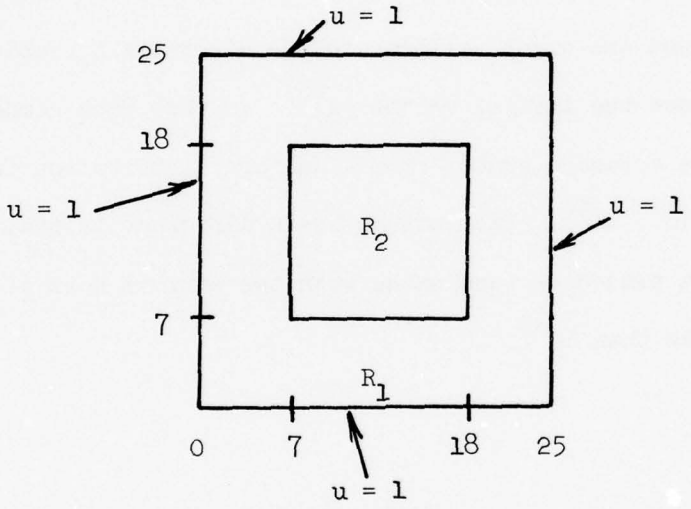
Region	R_1	R_2	R_3	R_4
a(x,y)	5	4	9	1

We choose $\sigma(x,y) \equiv 0$ and $f(x,y) \equiv 0$ so that the null vector $\underset{\sim}{0}$ is the solution to the discretized problem. We use the point red/black ordering.

n = 1152

Method	Iter	Time
CG	262	42.61
J-CG	136	25.61
RS-CG	66	9.19

EXAMPLE 2.



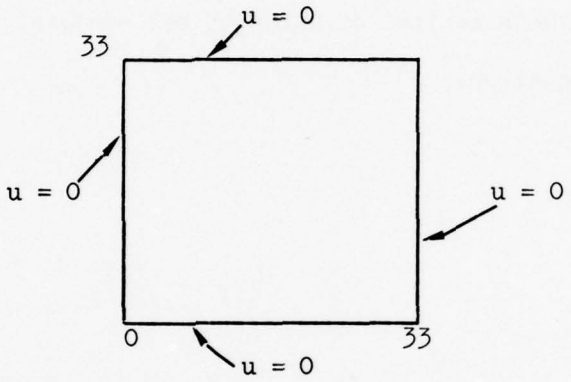
Region	R_1	R_2
a(x,y)	2	500

We choose $\sigma(x,y) \equiv 0.05$ and $f(x,y) \equiv 0.05$ so that the vector $\underset{\sim}{e} \equiv (1,1,\ldots,1)^t$ is the solution to the discretized problem. We use the point red/black ordering.

n = 576

Method	Iter	Time
CG	400	33.34
J-CG	78	7.52
RS-CG	40	2.80

EXAMPLE 3.



We choose $a(x,y) \equiv 1$, $\sigma(x,y) \equiv 0$ and $f(x,y) \equiv 0$ so that the null vector solves the discretized problem. We use a 1-line red/black ordering. The tridiagonal systems of linear equations are solved by Gaussian elimination; pivoting is not necessary because the systems are positive definite.

n = 1024

Method	Iter	Time
CG	86	12.89
J-CG	62	12.05
RS-CG	34	4.55

Our examples show how a (block) diagonal matrix preconditioning can greatly improve the convergence rate of the CG method, especially for the case where the diagonal elements of the coefficient matrix are of different orders of magnitude (see Example 2). The RS-CG method, as expected, requires about half as many iterations as the J-CG method. The saving in machine execution time is, however, even more substantial. It is thus fair to conclude that the RS-CG method is an effective procedure for the numerical solution of self-adjoint elliptic partial differential equations.

REFERENCES

[1] Axelsson, O., "Solution of linear systems of equations: iterative methods," Chapter 1 of Sparse Matrix Techniques, Copenhagen 1976, Springer-Verlag, Berlin (1977).

[2] Concus, P., Golub, G.H., and O'Leary, D.P., "A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations," in Sparse Matrix Computations (J. R. Bunch and D. J. Rose, Ed.), Academic Press, New York (1976), 309-332.

[3] Faddeev, D.K., and Faddeeva, V.N., Computational Methods of Linear Algebra, Freeman, San Francisco (1963).

[4] Golub, G.H., and Varga, R.S., "Chebyshev semi-iterative methods, successive over-relaxation iterative methods and second order Richardson iterative methods," Numer. Math. 3 (1961), 147-168.

[5] Hageman, L.A., "Block iterative methods for two-cyclic matrix equations with special application to the numerical solution of the second-order self-adjoint elliptic partial differential equations in two dimensions," Report WAPD-TM-327, Bettis Atomic Power Laboratory (1962).

[6] Hestenes, M., and Stiefel, E., "Methods of conjugate gradients for solving linear equations," J. Research NBS 49 (1952), 409-436.

[7] Meijerink, J.A., and van der Vorst, H.A., "An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix," Math. Comp. 31 (1977), 148-162.

[8] Reid, J.K., "On the method of conjugate gradients for the solution of large sparse systems of linear equations," in Large Sparse Sets of Linear Equations, (J. K. Reid, Ed.), Academic Press, New York (1971), 231-254.

[9] Reid, J.K., "The use of conjugate gradients for systems of linear equations possessing 'Property A'," SIAM J. Numer. Anal. 9 (1972), 325-332.

[10] Rutishauser, H., "Theory of gradient methods," Chapter 2 of Refined Iterative Methods for Computation of the Solution and the Eigenvalues of Self-adjoint Boundary Value Problems, M. Engeli, Th. Ginsburg, H. Rutishauser and E. Stiefel, Birkhäuser, Basel (1959).

[11] Varga, R.S., Matrix Iterative Analysis, Prentice-Hall, New York (1962).

III. QUADRATIC PROGRAMMING WITH M-MATRICES

3.1. Introduction

In this chapter, we address the lower and upper bounds quadratic program

$$\min_{\underset{\sim}{x}} \tfrac{1}{2} \underset{\sim}{x}^t A \underset{\sim}{x} - \underset{\sim}{x}^t \underset{\sim}{b} \qquad \text{subject to} \quad \underset{\sim}{c} \leq \underset{\sim}{x} \leq \underset{\sim}{d} , \tag{1.1}$$

where A is an $n \times n$ M-matrix and $\underset{\sim}{b}$, $\underset{\sim}{c}$ and $\underset{\sim}{d}$ are n-vectors. An important special case of (1.1) is the linear complementarity problem, in which $\underset{\sim}{c} = \underset{\sim}{0}$ and $\underset{\sim}{d} = \underset{\sim}{\infty}$:

$$\min_{\underset{\sim}{x}} \tfrac{1}{2} \underset{\sim}{x}^t A \underset{\sim}{x} - \underset{\sim}{x}^t \underset{\sim}{b} \qquad \text{subject to} \quad \underset{\sim}{x} \geq \underset{\sim}{0} . \tag{1.2}$$

We assume that the matrix A is large and sparse. The problems (1.1) and (1.2) find applications in the numerical solution of free boundary problems for elliptic partial differential equations. Such problems include various types of Dirichlet problems with obstacles ([7] and [10]), and models of the journal bearing [5] and of the application of torsion to a bar [1].

We define an M-matrix as follows.

DEFINITION 1.1 [11, p. 85]. A real square matrix $A = (a_{ij})$ with $a_{ij} \leq 0$ for all $i \neq j$ is an M-matrix if A is nonsingular, and $A^{-1} \geq 0$.

We lose no generality by restricting our attention to symmetric M-matrices, for we can replace the matrix A by its symmetric part $(A + A^t)/2$ in the quadratic form of (1.1) or (1.2) without changing the value of the quadratic form. The next lemma states that a symmetric M-matrix is positive definite. Thus, the problem (1.1) or (1.2) always has a unique solution.

DEFINITION 1.2 [11, p. 85]. A real square matrix $A = (a_{ij})$ with $a_{ij} \leq 0$ for all $i \neq j$ is a Stieltjes matrix if A is symmetric and positive definite.

LEMMA 1.1 [11, p. 85 and 87]. A symmetric M-matrix is a Stieltjes matrix and vice versa.

There are many good algorithms for solving problems (1.1) and (1.2) when the matrix A is positive definite (see [4]). However, it is possible to exploit the special properties of an M-matrix to obtain more efficient special algorithms. Chandrasekaran [2] proposed an algorithm for the linear complementarity problem (1.2), and Pang [8] developed an analogous algorithm for the lower and upper bounds problem (1.1). Cottle and Goheen [4] performed an extensive comparison of the latter method with four other well known algorithms. Their results indicate that Pang's method is the most efficient. They also described a preprocessing scheme that could be used with the method. Their scheme would identify some of the variables which will be at their bounds in

the optimal solution. Those variables could then be eliminated from further consideration.

In this chapter, we study the algorithms of Chandrasekaran and Pang. We demonstrate the special monotone behavior of the iterate and gradient vectors. The result on the gradient vector is new. It leads us to consider a simple updating procedure which preserves the monotonicity of both vectors. The procedure uses the fact that an M-matrix has a nonnegative inverse. Two new algorithms are then constructed by incorporating this updating procedure into the two described algorithms. Numerical tests show that our new algorithms can be twice as fast as the original methods (see Section 3.5).

We also consider the extension of problem (1.1) to the case when some components of the bounds are infinite. We show how we can compute finite a priori bounds on all the variables and reduce the extended problem to one with finite bounds (see Section 3.4). This result is new.

Let

$$\nu(\underset{\sim}{x}) \equiv \frac{1}{2}\, \underset{\sim}{x}^t A \underset{\sim}{x} - \underset{\sim}{x}^t \underset{\sim}{b} \tag{1.3}$$

and

$$\underset{\sim}{g}(\underset{\sim}{x}) \equiv A\underset{\sim}{x} - \underset{\sim}{b}\ , \tag{1.4}$$

i.e., $\nu(\underset{\sim}{x})$ is the value of the quadratic form at $\underset{\sim}{x}$ and $\underset{\sim}{g}(\underset{\sim}{x})$ is the gradient vector of the quadratic form at $\underset{\sim}{x}$. The solution $\underset{\sim}{x}$ for problem (1.1) must satisfy the optimality conditions:

$$\begin{aligned} &\text{if } x_i = c_i && \text{then } g_i(\underset{\sim}{x}) \geq 0 \\ &\text{if } c_i < x_i < d_i && \text{then } g_i(\underset{\sim}{x}) = 0 \\ &\text{if } x_i = d_i && \text{then } g_i(\underset{\sim}{x}) \leq 0\ , \end{aligned} \tag{1.5}$$

for $i = 1,2,\ldots,n$. The solution $\underset{\sim}{x}$ for problem (1.2) must satisfy the optimality conditions:

$$\begin{aligned} &\text{if } x_i = 0 \qquad \text{then} \qquad g_i(\underset{\sim}{x}) \geq 0 \\ &\text{if } x_i > 0 \qquad \text{then} \qquad g_i(\underset{\sim}{x}) = 0 , \end{aligned} \tag{1.6}$$

for $i = 1,2,\ldots, n$.

The following lemma and notations are used in this chapter.

DEFINITION 1.3 [11, p. 30]. A principal submatrix of an $n \times n$ matrix A is any matrix obtained by crossing out any j rows and the corresponding j columns of A, where $1 \leq j < n$.

LEMMA 1.2 [11, p. 86]. Any principal submatrix of an M-matrix is an M-matrix.

NOTATIONS.

(1) $N \equiv \{1,2,\ldots,n\}$.

(2) $\bar{J} \equiv N \setminus J$ for any subset J of N, i.e. $\bar{J}$ is the complement of J in N.

(3) $\underset{\sim}{x}_J \equiv (x_{j_1}, x_{j_2}, \ldots, x_{j_p})$, where

$J = \{j_1, j_2, \ldots, j_p\} \subseteq N$ and $j_1 < j_2 < \cdots < j_p$.

(4) $A_{JK} \equiv (c_{rs})$, where

$c_{rs} = a_{j_r k_s}$, $J = \{j_1, j_2, \ldots, j_p\} \subseteq N$,

$K = \{k_1, k_2, \ldots, k_q\} \subseteq N$, $j_1 < j_2 < \cdots < j_p$ and $k_1 < k_2 < \cdots < k_q$.

3.2. Linear Complementarity Problem

In this section, we consider the linear complementarity problem

$$\min_{\underset{\sim}{x}} \frac{1}{2} \underset{\sim}{x}^t A \underset{\sim}{x} - \underset{\sim}{x}^t \underset{\sim}{b} \qquad (1.2)$$

$$\text{subject to} \quad \underset{\sim}{x} \geq \underset{\sim}{0} ,$$

where A is a symmetric M-matrix. We assume that $\underset{\sim}{b}$ contains both positive and negative components to eliminate two trival cases.

LEMMA 2.1.

(1) If $\underset{\sim}{b} \leq \underset{\sim}{0}$, then $\underset{\sim}{x} = \underset{\sim}{0}$ solves (1.2).

(2) If $\underset{\sim}{b} \geq \underset{\sim}{0}$, then $\underset{\sim}{x} = A^{-1}\underset{\sim}{b}$ solves (1.2).

Proof.

(1) Assume $\underset{\sim}{x}$ solves (1.2) with $\underset{\sim}{x}_P > \underset{\sim}{0}$ and $\underset{\sim}{x}_{\bar{P}} = \underset{\sim}{0}$ for some nonempty set $P \subseteq N$. The optimality conditions (1.6) give

$$\underset{\sim}{g}_P(\underset{\sim}{x}) \equiv A_{PP}\underset{\sim}{x}_P - \underset{\sim}{b}_P = \underset{\sim}{0} .$$

Hence

$$\underset{\sim}{x}_P = A_{PP}^{-1}\underset{\sim}{b}_P \leq \underset{\sim}{0} \quad \text{because} \quad A_{PP}^{-1} \geq 0 \quad \text{and} \quad \underset{\sim}{b}_P \leq \underset{\sim}{0} .$$

Contradiction.

(2) We have

$$\underset{\sim}{x} = A^{-1}\underset{\sim}{b} \geq \underset{\sim}{0} \quad \text{because} \quad A^{-1} \geq 0 \quad \text{and} \quad \underset{\sim}{b} \geq \underset{\sim}{0} ,$$

and

$$\underset{\sim}{g}(\underset{\sim}{x}) = A\underset{\sim}{x} - \underset{\sim}{b} = \underset{\sim}{0} .$$

Thus, $\underset{\sim}{x}$ solves (1.2). □

We proceed to prove two more lemmas and an important theorem.

LEMMA 2.2. Suppose we have a Stieltjes matrix A and a nonempty subset I of N. The matrix

$$V \equiv A_{II} - A_{I\bar{I}} A^{-1}_{\bar{I}\bar{I}} A_{\bar{I}I}$$

is a Stieltjes matrix.

Proof. The matrix V is symmetric because $A_{I\bar{I}} = A^{t}_{\bar{I}I}$. Let $W \equiv A^{-1}$. Since A is a positive definite matrix, so is W. A fortiori, so are W_{II} and W^{-1}_{II}. But $V = W^{-1}_{II}$ [6, p. 99]. The matrix $A_{\bar{I}\bar{I}}$ is an M-matrix by Lemma 1.2. Hence $A^{-1}_{\bar{I}\bar{I}} \geq 0$. The off-diagonal elements of V are therefore nonpositive because the off-diagonal elements of A_{II} are nonpositive, and $A_{I\bar{I}} \leq 0$, $A^{-1}_{\bar{I}\bar{I}} \geq 0$ and $A_{\bar{I}I} \leq 0$. □

LEMMA 2.3. Suppose we have vectors $\underset{\sim}{x}$ and $\underset{\sim}{y}$ with $\underset{\sim}{x}_J = \underset{\sim}{y}_J$ and $\underset{\sim}{x}_{\bar{J}} \leq \underset{\sim}{y}_{\bar{J}}$ for some nonempty subset J of N. Then $\underset{\sim}{g}_J(\underset{\sim}{x}) \geq \underset{\sim}{g}_J(\underset{\sim}{y})$.

Proof.

$$\begin{aligned}
\underset{\sim}{g}_J(\underset{\sim}{x}) &= A_{JJ}\underset{\sim}{x}_J + A_{J\bar{J}}\underset{\sim}{x}_{\bar{J}} - \underset{\sim}{b}_J \\
&= A_{JJ}\underset{\sim}{y}_J + A_{J\bar{J}}\underset{\sim}{y}_{\bar{J}} - \underset{\sim}{b}_J + A_{J\bar{J}}(\underset{\sim}{x}_{\bar{J}} - \underset{\sim}{y}_{\bar{J}}) \\
&= \underset{\sim}{g}_J(\underset{\sim}{y}) + A_{J\bar{J}}(\underset{\sim}{x}_{\bar{J}} - \underset{\sim}{y}_{\bar{J}}) \\
&\geq \underset{\sim}{g}_J(\underset{\sim}{y})
\end{aligned}$$

because $A_{J\bar{J}} \leq 0$ and $\underset{\sim}{x}_{\bar{J}} - \underset{\sim}{y}_{\bar{J}} \leq \underset{\sim}{0}$. □

THEOREM 2.1. Assume $A_{JJ}^{-1}b_J > 0$ for some nonempty subset J of N. Define a vector x with $x_J = A_{JJ}^{-1}b_J$ and $x_{\bar{J}} = 0$. Let K and L be two sets partitioning $\bar{J}$ so that $g_K(x) < 0$ and $g_L(x) \geq 0$.

If the set K is empty, then

(1) x solves (1.2),

else

(2) let $Q := J \cup K$. Construct y with $y_Q = A_{QQ}^{-1}b_Q$ and $y_L = 0$.

We get

(a) $y_J \geq x_J > 0$, $y_K > 0$,

(b) $g_L(y) \leq g_L(x)$,

and

(c) $\nu(y) < \nu(x)$. (Remember $\nu(z) \equiv \frac{1}{2} z^t A z - z^t b$).

Proof.

(1) The optimality conditions (1.6) are satisfied because $g_J(x) = 0$ and $g_{\bar{J}}(x) \geq 0$.

(2a) From the system

$$A_{JJ}y_J + A_{JK}y_K = b_J ,$$

$$A_{KJ}y_J + A_{KK}y_K = b_K ,$$

we obtain

$$y_K = (A_{KK} - A_{KJ}A_{JJ}^{-1}A_{JK})^{-1} (b_K - A_{KJ}A_{JJ}^{-1}b_J) .$$

But

$$b_K - A_{KJ}A_{JJ}^{-1}b_J = b_K - A_{KJ}x_J$$

$$= -g_K(x) > 0$$

and

$$(A_{KK} - A_{KJ}A_{JJ}^{-1}A_{JK})^{-1} \geq 0 \quad \text{by Lemma 2.2.}$$

Thus,

$$\underset{\sim}{y}_K > \underset{\sim}{0} .$$

Also,

$$\begin{aligned} \underset{\sim}{y}_J &= A_{JJ}^{-1}\underset{\sim}{b}_J - A_{JJ}^{-1}A_{JK}\underset{\sim}{y}_K \\ &\geq A_{JJ}^{-1}\underset{\sim}{b}_J \qquad \text{because } A_{JJ}^{-1} \geq 0,\ A_{JK} \leq 0 \text{ and } \underset{\sim}{y}_K > \underset{\sim}{0} \\ &= \underset{\sim}{x}_J \\ &> \underset{\sim}{0} . \end{aligned}$$

(2b) We have shown in (2a) that $\underset{\sim}{y}_J \geq \underset{\sim}{x}_J$ and $\underset{\sim}{y}_K > \underset{\sim}{0} = \underset{\sim}{x}_K$. Let $\bar{L} = J \cup K$ and $\underset{\sim}{y}_L = \underset{\sim}{x}_L = \underset{\sim}{0}$. The conditions of Lemma 2.3 are satisfied and therefore $g_L(\underset{\sim}{y}) \leq g_L(\underset{\sim}{x})$.

$$\begin{aligned} (2c)\ 2\cdot\nu(\underset{\sim}{y}) &= \underset{\sim}{y}_Q^t A_{QQ}\underset{\sim}{y}_Q - 2\underset{\sim}{y}_Q^t\underset{\sim}{b}_Q \\ &= -\underset{\sim}{b}_Q^t A_{QQ}^{-1}\underset{\sim}{b}_Q < (\underset{\sim}{x}_Q - A_{QQ}^{-1}\underset{\sim}{b}_Q)^t\, A_{QQ}(\underset{\sim}{x}_Q - A_{QQ}^{-1}\underset{\sim}{b}_Q) - \underset{\sim}{b}_Q^t A_{QQ}^{-1}\underset{\sim}{b}_Q \\ &= \underset{\sim}{x}_Q^t A_{QQ}\underset{\sim}{x}_Q - 2\underset{\sim}{x}_Q^t\underset{\sim}{b}_Q \\ &= 2\cdot\nu(\underset{\sim}{x}). \qquad \square \end{aligned}$$

Let Z denote the index set of the constrained (zero) components of the iterate vector and P the index set of the unconstrained (positive) components. We can describe the lower bound algorithm in [2] as follows.

ALGORITHM 2.1 (Chandrasekaran's method). Let

$$P := \{i \in N | b_i > 0\} \quad \text{and} \quad Z := \bar{P} .$$

Define $\underset{\sim}{x}$ so that $\underset{\sim}{x}_P = A_{PP}^{-1}\underset{\sim}{b}_P$ and $\underset{\sim}{x}_Z = \underset{\sim}{0}$. Let J be a nonempty set.

Repeat until either set Z or J is empty:

1. Compute $\underset{\sim}{g}_Z := A_{ZP}\underset{\sim}{x}_P - \underset{\sim}{b}_Z$
2. Let $J := \{j \in Z | g_j < 0\}$.
3. If set J is nonempty, then
   (a) Let $P := P \cup J$ and $Z := Z \setminus J$.
   (b) Reconstruct $\underset{\sim}{x}$ so that $\underset{\sim}{x}_P = A_{PP}^{-1}\underset{\sim}{b}_P$ and $\underset{\sim}{x}_Z = \underset{\sim}{0}$. □

Part (2a) of Theorem 2.1 says that the iterate vector is non-decreasing in value. Thus, a positive component stays positive and no element leaves the index set P. We enlarge set P when we release variables with negative gradients from their constraints. If no such variables exist, then we have computed the solution (Part (1) of Theorem 2.1). Hence Algorithm 2.1 always terminates. It is a descent method by Part (2c) of the same theorem.

Most of the work in an iteration of Algorithm 2.1 is spent in the solution of a matrix equation. We can use an iterative method. The conjugate gradient method (cf. [9]) or its generalized variant (cf. [3]) is particularly effective for solving a large sparse set of linear equations when the coefficient matrix is symmetric and positive definite. Furthermore, we can construct a good starting vector from the solution to the matrix equation in the previous iteration. A straightforward strategy is to place zeros in the new positions. In many cases, the quadratic program (1.2) arises from an approximation to a continuous problem and some sort of interpolation and extrapolation schemes can be successfully exploited.

It is obvious that we do not need to solve the matrix equations to full accuracy in any but the last iteration. This gives another reason for using an iterative method instead of a direct method for solving the matrix equations.

The intent of solving the matrix equation is to go from the minimum of one subspace S_1 to the minimum of a larger subspace S_2 that contains S_1. The computation of the minima increases the size of the index set P rapidly. A disadvantage is that the cost involved can be substantial. Thus, we are interested in the possibility of using some simple computing process that lets us enlarge the set P with very little work. It will take more iterations to determine the final index set P, but the total cost may be lower because of the smaller amount of work per iteration.

Suppose we are at the start of an iteration of Algorithm 2.1. Let $\underset{\sim}{x}$ and $g(\underset{\sim}{x})$ be the iterate and gradient vectors, respectively. Let P be the index set of positive components of $\underset{\sim}{x}$ and J be the index set of zero components with negative gradients, i.e.,

$$P := \{j \in Z_n | x_j > 0\} \quad \text{and} \quad J := \{j \in \bar{P} | g_j < 0\} .$$

Assume set J is nonempty. Let $Q := P \cup J$. We construct the new iterate $\hat{\underset{\sim}{x}}$ so that $\hat{\underset{\sim}{x}}_{\bar{Q}} = \underset{\sim}{x}_{\bar{Q}}$ and $\hat{\underset{\sim}{x}}_Q = \underset{\sim}{y}$, where $\underset{\sim}{y}$ solves the matrix equation

$$A_{QQ}\underset{\sim}{y} = \underset{\sim}{b}_Q . \tag{2.1}$$

Let us consider a cheaper way to construct $\hat{\underset{\sim}{x}}$: update only the j-th components, where $j \in J$. In other words, $\hat{\underset{\sim}{x}}_{\bar{J}} = \underset{\sim}{x}_{\bar{J}}$ and $\hat{\underset{\sim}{x}}_J = \underset{\sim}{z}$, where $\underset{\sim}{z}$ solves the matrix equation

$$B\underset{\sim}{z} = -\underset{\sim}{g}_J \tag{2.2}$$

and B is an M-matrix of order equal to the size of J. Thus $\underset{\sim}{z} > \underset{\sim}{0}$. We can choose B as A_{JJ}, but a better strategy is to choose B so that the matrix equation (2.2) is easily solvable. Regardless of which B we choose, it is very important that the new iterate $\hat{\underset{\sim}{x}}$ satisfies the inequality

$$\hat{\underset{\sim}{x}}_Q \leq A_{QQ}^{-1}\underset{\sim}{b}_Q . \tag{2.3}$$

If (2.3) does not hold, then the gradients of some constrained variables

may have decreased too much in value (cf. Lemma 2.3) and those variables may be erroneously released from their constraints.

We propose to choose B as C_{JJ}, where C is an $n \times n$ M-matrix so that $C \geq A$. The next theorem shows that the inequality (2.3) always holds for these choices of B and C. We now give a simple way to construct the matrix C.

LEMMA 2.4 [11, p. 85]. Let A be an M-matrix, and let C be any matrix obtained from A by setting certain off-diagonal entries of the matrix to zero. Then, C is also an M-matrix.

A possible choice for C is therefore a (block) diagonal matrix with the same (block) diagonal part as A.

ASSUMPTION 2.1. We have chosen an M-matrix C such that $C \geq A$ and that matrix equations with C or a principal submatrix of C as the coefficient matrix are easily solvable.

LEMMA 2.5. Suppose there is a vector $\underset{\sim}{x} \geq \underset{\sim}{0}$ such that $\underset{\sim}{x}_J = \underset{\sim}{0}$ and $\underset{\sim}{g}_J(\underset{\sim}{x}) < \underset{\sim}{0}$ for some nonempty subset J of N. Construct a vector $\underset{\sim}{y}$ so that $\underset{\sim}{y}_{\bar{J}} = \underset{\sim}{x}_{\bar{J}}$ and $\underset{\sim}{y}_J = -C_{JJ}^{-1}\underset{\sim}{g}_J(\underset{\sim}{x})$. Then $\underset{\sim}{g}_J(\underset{\sim}{y}) \leq \underset{\sim}{0}$.

Proof. C_{JJ} is an M-matrix by Lemma 1.2. Hence $C_{JJ}^{-1} \geq 0$. But $A_{JJ} \leq C_{JJ}$. Thus,

$$A_{JJ}C_{JJ}^{-1} \leq C_{JJ}C_{JJ}^{-1} = I ,$$

which implies

$$I - A_{JJ}C_{JJ}^{-1} \geq 0 .$$

Finally,

$$\begin{aligned} g_J(\underset{\sim}{y}) &= A_{JJ}\underset{\sim}{y}_J + A_{J\bar{J}}\underset{\sim}{y}_{\bar{J}} - \underset{\sim}{b}_J \\ &= -A_{JJ}C_{JJ}^{-1}g_J(\underset{\sim}{x}) + A_{J\bar{J}}\underset{\sim}{x}_{\bar{J}} - \underset{\sim}{b}_I \\ &= -A_{JJ}C_{JJ}^{-1}g_J(\underset{\sim}{x}) + g_J(\underset{\sim}{x}) \qquad \text{because} \qquad \underset{\sim}{x}_J = \underset{\sim}{0} \\ &= (I - A_{JJ}C_{JJ}^{-1})\, g_J(\underset{\sim}{x}) \\ &\leq \underset{\sim}{0} \qquad \text{because} \qquad g_J(\underset{\sim}{x}) < \underset{\sim}{0} \,. \end{aligned}$$ □

LEMMA 2.6. Suppose there is a nonempty subset J of N and two vectors $\underset{\sim}{x}$ and $\underset{\sim}{y}$ such that $\underset{\sim}{x}_{\bar{J}} = \underset{\sim}{0}$, $g_J(\underset{\sim}{x}) = \underset{\sim}{0}$, $\underset{\sim}{y}_{\bar{J}} = \underset{\sim}{0}$ and $g_J(\underset{\sim}{y}) \leq \underset{\sim}{0}$. Then $\underset{\sim}{x}_J \geq \underset{\sim}{y}_J$.

Proof.

$$\begin{aligned} A_{JJ}(\underset{\sim}{x}_J - \underset{\sim}{y}_J) &= A_{JJ}\underset{\sim}{x}_J - \underset{\sim}{b}_J - (A_{JJ}\underset{\sim}{y}_J - \underset{\sim}{b}_J) \\ &= g_J(\underset{\sim}{x}) - g_J(\underset{\sim}{y}) \\ &\geq \underset{\sim}{0} \,. \end{aligned}$$

Hence

$$\underset{\sim}{x}_J - \underset{\sim}{y}_J \geq \underset{\sim}{0} \,.$$ □

THEOREM 2.2. Assume we have a vector $\underset{\sim}{x} \geq \underset{\sim}{0}$. Let $P := \{j \in N | x_j > 0\}$ and $J := \{j \in \bar{P} | g_j(\underset{\sim}{x}) < 0\}$. Suppose that both index sets P and J are nonempty, and that $g_P(\underset{\sim}{x}) \leq \underset{\sim}{0}$. Construct a vector $\underset{\sim}{y}$ such that $\underset{\sim}{y}_J = -C_{JJ}^{-1}g_J(\underset{\sim}{x})$ and $\underset{\sim}{y}_{\bar{J}} = \underset{\sim}{x}_{\bar{J}}$. Let $K := \bar{P} \setminus J$. Then

(1) $g_P(\underset{\sim}{y}) \leq g_P(\underset{\sim}{x}) \leq \underset{\sim}{0}$, $g_J(\underset{\sim}{y}) \leq \underset{\sim}{0}$ and $g_K(\underset{\sim}{y}) \leq g_K(\underset{\sim}{x})$.

(2) $\underset{\sim}{y}_Q \leq A_{QQ}^{-1}\underset{\sim}{b}_Q$, where $Q = P \cup J$.

(3) $\nu(\underset{\sim}{y}) < \nu(\underset{\sim}{x})$.

Proof.

(1) By Lemma 2.5, $g_J(\underset{\sim}{y}) \leq \underset{\sim}{0}$. Since $\underset{\sim}{y}_J > \underset{\sim}{0} = \underset{\sim}{x}_J$ and $\underset{\sim}{y}_{\bar{J}} = \underset{\sim}{x}_{\bar{J}}$, we get $g_{\bar{J}}(\underset{\sim}{y}) \leq g_{\bar{J}}(\underset{\sim}{x})$ by Lemma 2.3. But $\bar{J} = P \cup K$.

(2) Define a vector $\underset{\sim}{z}$ so that $\underset{\sim}{z}_Q = A_{QQ}^{-1}\underset{\sim}{b}_Q$ and $z_{\bar{Q}\bar{Q}} = \underset{\sim}{y}_{\bar{Q}} = \underset{\sim}{0}$. Hence $g_Q(\underset{\sim}{z}) = \underset{\sim}{0}$. From Part (1), $g_Q(\underset{\sim}{y}) \leq \underset{\sim}{0}$. Thus, $\underset{\sim}{z}_Q \geq \underset{\sim}{y}_Q$ by Lemma 2.6.

(3) $$\nu(\underset{\sim}{y}) - \nu(\underset{\sim}{x}) = \frac{1}{2}\underset{\sim}{y}_J^t A_{JJ}\underset{\sim}{y}_J + \underset{\sim}{y}_J^t A_{J\bar{J}}\underset{\sim}{x}_{\bar{J}} - \underset{\sim}{y}_J^t\underset{\sim}{b}_J$$

$$= \frac{1}{2}\underset{\sim}{y}_J^t A_{JJ}(-C_{JJ}^{-1}g_J(\underset{\sim}{x})) + \underset{\sim}{y}_J^t(A_{J\bar{J}}\underset{\sim}{x}_{\bar{J}} - \underset{\sim}{b}_J)$$

$$= \underset{\sim}{y}_J^t(I - \frac{1}{2}A_{JJ}C_{JJ}^{-1})\, g_J(\underset{\sim}{x}) .$$

We have shown that $I - A_{JJ}C_{JJ}^{-1} \geq 0$. Hence $(I - (A_{JJ}C_{JJ}^{-1})/2) \geq 0$ and equality is not possible. Since $\underset{\sim}{y}_J > \underset{\sim}{0}$ and $g_J(\underset{\sim}{x}) < \underset{\sim}{0}$, we get $\nu(\underset{\sim}{y}) < \nu(\underset{\sim}{x})$. □

We now have the tool to modify Algorithm 2.1. Instead of going from one constrained minimum to another through solving matrix equations involving the unconstrained variables, we take descent steps through solving very simple matrix equations involving only those just released variables. We call our technique "partial updating". It lowers the

gradients of some constrained variables and we can again release those variables with negative gradients from the lower bound using the same technique. The process is repeated until the gradients of the constrained variables are all nonnegative. Then we compute the constrained minimum by solving the matrix equation

$$A_{PP}\underset{\sim}{y} = \underset{\sim}{b}_P \tag{2.4}$$

and defining the new iterate $\underset{\sim}{x}$ so that $\underset{\sim}{x}_P = \underset{\sim}{y}$ and $\underset{\sim}{x}_{\bar{P}} = \underset{\sim}{0}$. The iterate vector from the "partial updating" is usually a very good initial vector for solving (2.4) using an iterative method.

ALGORITHM 2.2. (Modified Chandrasekaran's method). Let

$$J := \{j \in N | b_j > 0\}, \quad P := J ,$$

$Z := \bar{J}$, $\underset{\sim}{x} = \underset{\sim}{0}$ and $\underset{\sim}{g} := -\underset{\sim}{b}$.

Repeat until either set Z or J is empty:

1. Repeat until either set Z or J is empty:
   (a) Compute $\underset{\sim}{x}_J := -C_{JJ}^{-1}\underset{\sim}{g}_J$.
   (b) Update $\underset{\sim}{g}_Z := \underset{\sim}{g}_Z + A_{ZJ}\underset{\sim}{x}_J$.
   (c) Redefine $J := \{j \in Z | g_j < 0\}$.
   (d) Let $P := P \cup J$ and $Z := Z \setminus J$.
2. Compute $\underset{\sim}{x}_P := A_{PP}^{-1}\underset{\sim}{b}_P$.
3. Compute $\underset{\sim}{g}_Z := A_{ZP}\underset{\sim}{x}_P - \underset{\sim}{b}_Z$.
4. Redefine $J := \{y \in Z | g_j < 0\}$. Let $P := P \cup J$ and $Z := Z \setminus J$. □

We can easily extend Algorithm 2.2 to solve the general lower-bound quadratic program

$$\min_{\underset{\sim}{x}} \frac{1}{2} \underset{\sim}{x}^t A \underset{\sim}{x} - \underset{\sim}{x}^t \underset{\sim}{b} \tag{2.5}$$

$$\text{subject to} \quad \underset{\sim}{x} \geq \underset{\sim}{c} .$$

ALGORITHM 2.3. Let

$$P := \emptyset, \quad Z := N, \quad \underset{\sim}{x} = \underset{\sim}{c}, \quad \text{and} \quad \underset{\sim}{g} := A\underset{\sim}{c} - \underset{\sim}{b} .$$

Set $J := \{j \in Z | g_j < 0\}$.

Repeat until either set Z or J is empty:

1. Repeat until either set Z or J is empty:
   (a) Compute $\underset{\sim}{x}_J := \underset{\sim}{c}_J - C_{JJ}^{-1} \underset{\sim}{g}_J$.
   (b) Update $\underset{\sim}{g}_Z := \underset{\sim}{g}_Z + A_{ZJ} \underset{\sim}{x}_J$.
   (c) Redefine $J := \{j \in Z | g_j < 0\}$.
   (d) Let $P := P \cup J$ and $Z := Z \setminus J$.
2. Compute $\underset{\sim}{x}_P := A_{PP}^{-1}(\underset{\sim}{b}_P - A_{PZ}\underset{\sim}{c}_Z)$.
3. Compute $\underset{\sim}{g}_Z := A_{ZP}\underset{\sim}{x}_P + A_{ZZ}\underset{\sim}{c}_Z - \underset{\sim}{b}_Z$.
4. Redefine $J := \{j \in Z | g_j < 0\}$.
5. Let $P := P \cup J$ and $Z : Z \setminus J$. □

3.3. Lower and Upper Bounds Problem

In this section, we consider the lower and upper bounds quadratic program

$$\min_{\underset{\sim}{x}} \frac{1}{2} \underset{\sim}{x}^t A \underset{\sim}{x} - \underset{\sim}{x}^t \underset{\sim}{b} \tag{1.1}$$

$$\text{subject to} \quad \underset{\sim}{c} \leq \underset{\sim}{x} \leq \underset{\sim}{d} .$$

We introduce an index set U denoting the variables constrained at the upper bound $\underset{\sim}{d}$. The index sets P and Z denote the unconstrained variables and the variables constrained at the lower bound $\underset{\sim}{c}$, respectively.

Our lower and upper bounds algorithm starts with the iterate vector $\underset{\sim}{x}$ at the upper bound $\underset{\sim}{d}$ and the index set U equal to N. We examine the gradient vector $\underset{\sim}{g}(\underset{\sim}{x}) \equiv A\underset{\sim}{x} - \underset{\sim}{b}$, and release those variables whose gradients are positive. Let P denote the just released variables and let $U := \bar{P}$. We solve the lower bound quadratic program

$$\min_{\underset{\sim}{y}} \frac{1}{2} \underset{\sim}{y}^t A_{PP} \underset{\sim}{y} - \underset{\sim}{y}^t (\underset{\sim}{b}_P - A_{PU} \underset{\sim}{d}_U) \tag{3.1}$$

$$\text{subject to} \quad \underset{\sim}{y} \geq \underset{\sim}{c}_P .$$

The iterate vector $\underset{\sim}{x}$ is redefined so that $\underset{\sim}{x}_U = \underset{\sim}{d}_U$ and $\underset{\sim}{x}_P = \underset{\sim}{y}$, the solution to (3.1). The indices of the components of $\underset{\sim}{x}$ at the lower bound are moved from set P to set Z. We start a new iteration by releasing those variables at the upper bound whose gradients have now become positive.

THEOREM 3.1. Suppose we have a vector $\underset{\sim}{x}$ and two nonempty index sets U and P partitioning N such that $\underset{\sim}{x}_U = \underset{\sim}{d}_U$ and $\underset{\sim}{x}_P = \underset{\sim}{y}$, where $\underset{\sim}{y}$ solves (3.1). Let $J := \{j \in U | g_j(\underset{\sim}{x}) > 0\}$.

If the index set J is empty, then

1. $\underset{\sim}{x}$ solves the lower and upper bounds quadratic program (1.1), else
2. let $P := \{j \in P | x_j > c_j\}$, $Z := \{j \in P | x_j = c_j\}$, $K := P \cup J$ and $L := U \setminus J$. Define a vector $\underset{\sim}{y}$ so that $\underset{\sim}{y}_L = \underset{\sim}{d}_L$, $\underset{\sim}{y}_Z = \underset{\sim}{c}_Z$ and $\underset{\sim}{y}_K = \underset{\sim}{p}$, where $\underset{\sim}{p}$ solves the lower bound quadratic program

$$\min_{\underset{\sim}{p}} \frac{1}{2} \underset{\sim}{p}^t A_{KK} \underset{\sim}{p} - \underset{\sim}{p}^t (\underset{\sim}{b}_K - A_{KL} \underset{\sim}{d}_L)$$

subject to $\underset{\sim}{p} \geq \underset{\sim}{c}_K$

Then

(a) $\underset{\sim}{y}_J < \underset{\sim}{d}_J$ and $\underset{\sim}{y}_P \leq \underset{\sim}{x}_P$.

(b) $g_L(\underset{\sim}{y}) \geq g_L(\underset{\sim}{x})$ and $g_Z(\underset{\sim}{y}) \geq g_Z(\underset{\sim}{x}) \geq \underset{\sim}{0}$.

(c) $\nu(\underset{\sim}{y}) < \nu(\underset{\sim}{x})$.

Proof.

(1) The optimality conditions (1.5) are satisfied because $g_U(\underset{\sim}{x}) \leq \underset{\sim}{0}$ and $\underset{\sim}{x}_P$ solves (3.1).

(2a) The inequalities hold trivially for those components of $\underset{\sim}{y}_K$ that are at the lower bound. The gradients of the other components equal zero and we can prove the inequalities using a technique similar to that of Part (2a) of Theorem 2.1.

(2b) We have shown in Part (2a) that $\underset{\sim}{y}_K \leq \underset{\sim}{x}_K$. As $\underset{\sim}{y}_{\bar{K}} = \underset{\sim}{x}_{\bar{K}}$ by construction, we get $g_{\bar{K}}(\underset{\sim}{y}) \geq g_{\bar{K}}(\underset{\sim}{x})$ from Lemma 2.3. But $\bar{K} = L \cup Z$.

(2c) Since $g_Z(y) \geq 0$ by Part (2b), the vector $g = y_{K \cup Z}$ solves the lower quadratic program

$$\min_{g} \frac{1}{2} g^t A_{QQ} g - g^t (b_Q - A_{QL} d_L)$$

$$\text{subject to} \quad g \geq c_Q ,$$

where $Q = K \cup Z$. But $\bar{L} = Q$, $y_L = x_L$ and $y \neq x$. Hence $\nu(y) < \nu(x)$. □

ALGORITHM 3.1. (Pang's method). Set $x := d$ and compute $g := Ad - b$. Let $J := \{j \in N | g_j > 0\}$, $P := J$, $U := \bar{J}$ and $Z := \emptyset$.

Repeat until set U or P or J is empty:

1. Reconstruct x so that $x_U = d_U$, $x_Z = c_Z$ and $x_P = y_P$, where y solves the lower bound quadratic program (3.1) (we may use Algorithm 2.3).
2. Set $K := \{k \in P | x_K = c_K\}$. Let $P := P \setminus K$ and $Z := Z \cup K$.
3. Compute $g_U := A_{UU} d_U + A_{UP} x_P + A_{UZ} c_Z - b_U$.
4. Redefine $J := \{j \in U | g_j > 0\}$. Set $P := P \cup J$ and $U := U \setminus J$. □

Part (2a) of Theorem 3.1 says that the iterate vector is non-increasing in value. Hence once a variable leaves its upper bound it never returns, and once a variable enters its lower bound it never exits. Since we release variables from their upper bounds when their gradients become positive, there is a flow of variables from the upper bound to the unconstrained region, and then to the lower bound. If

there is no outflow of variables from the upper bound, then the iterate vector is the desired solution (Part (1) of Theorem 3.1). Thus, Algorithm 3.1 always terminates. It is a descent method by Part (2c) of the same theorem. Algorithm 3.1 is essentially the algorithm proposed by Pang [8], except that he releases variables from their upper bounds when their gradients are nonnegative. The difference is so minor in real arithmetic that we expect both algorithms to produce identical results for almost all problems. Indeed, the two algorithms behaved identically in all our test examples.

We are interested in a "partial updating" technique that is similar to the one in the previous section. Such a technique may save many expensive solutions of lower bound quadratic programs. However, the presence of a lower bound restricts our choice of a "partial updating" matrix to a diagonal matrix. Recall that a positive definite matrix has a positive diagonal.

ASSUMPTION 3.1. We have chosen a diagonal matrix D such that $D \geq A$.

NOTATION. If $\underset{\sim}{v} = (v_1, v_2, \ldots, v_p)^t$, then $\underset{\sim}{v}_+ = (w_1, w_2, \ldots, w_p)^t$, where $w_j = \max(v_j, 0)$ for $j = 1, 2, \ldots, p$.

LEMMA 3.2. Suppose there is a vector $\underset{\sim}{x}$ with $\underset{\sim}{c} \leq \underset{\sim}{x} \leq \underset{\sim}{d}$ such that $\underset{\sim}{x}_J = \underset{\sim}{d}_J$ and $g_J(\underset{\sim}{x}) > 0$ for some nonempty subset J of N. Construct a vector $\underset{\sim}{y}$ so that $\underset{\sim}{y}_{\bar{J}} = \underset{\sim}{x}_{\bar{J}}$ and $\underset{\sim}{y}_J = \underset{\sim}{c}_J + [\underset{\sim}{d}_J - \underset{\sim}{c}_J - D_{JJ}^{-1} \underset{\sim}{g}_J(\underset{\sim}{x})]_+$. Then $g_J(\underset{\sim}{y}) \geq 0$.

Proof. There exists a diagonal matrix $\hat{D} \geq D_{JJ}$ such that

$$\underset{\sim}{y}_J = \underset{\sim}{c}_J + [\underset{\sim}{d}_J - \underset{\sim}{c}_J - \hat{D}^{-1} g_J(\underset{\sim}{x})] .$$

Hence $\hat{D} \geq A_{II}$ and this lemma can be proved in the same way as Lemma 2.5. □

Note that Lemma 3.2 may not hold if we replace D by a non-diagonal M-matrix E with $E \geq A$.

THEOREM 3.2. Suppose we have a vector $\underset{\sim}{x}$ with $\underset{\sim}{c} < \underset{\sim}{x} \leq \underset{\sim}{d}$ and two nonempty index sets U and P partitioning N such that $\underset{\sim}{x}_U = \underset{\sim}{d}_U$, $\underset{\sim}{x}_P < \underset{\sim}{d}_P$ and $g_P(\underset{\sim}{x}) \geq 0$. Let $J := \{j \in U | g_j(\underset{\sim}{x}) > 0\}$ and assume that it is not empty. Construct a vector $\underset{\sim}{y}$ such that $\underset{\sim}{y}_{\bar{J}} = \underset{\sim}{x}_{\bar{J}}$ and $\underset{\sim}{y}_J = \underset{\sim}{c}_J + [\underset{\sim}{d}_J - \underset{\sim}{c}_J - D_{JJ}^{-1} g_J(\underset{\sim}{x})]_+$. Let $K := U \setminus J$. Then

(1) $g_K(\underset{\sim}{y}) \geq g_K(\underset{\sim}{x})$, $g_J(\underset{\sim}{y}) \geq 0$ and $g_P(\underset{\sim}{y}) \geq g_P(\underset{\sim}{x}) \geq \underset{\sim}{0}$.

(2) $\underset{\sim}{y}_P \geq \underset{\sim}{g}$, where $\underset{\sim}{g}$ solves the lower bound quadratic program (3.1).

(3) $\nu(\underset{\sim}{y}) < \nu(\underset{\sim}{x})$.

Proof.

(1) By Lemma 3.2, $g_J(\underset{\sim}{y}) \geq \underset{\sim}{0}$. Since $\underset{\sim}{y}_J < \underset{\sim}{d}_J = \underset{\sim}{x}_J$ and $\underset{\sim}{y}_{\bar{J}} = \underset{\sim}{x}_{\bar{J}}$, we get $g_{\bar{J}}(\underset{\sim}{y}) \geq g_{\bar{J}}(\underset{\sim}{x})$ by Lemma 2.3. But $\bar{J} = K \cup P$.

(2) The inequality is trivial for those components of $\underset{\sim}{g}$ that are at the lower bound. Since the gradients of the other components of $\underset{\sim}{g}$ equal zero and $g_P(\underset{\sim}{y}) \geq \underset{\sim}{0}$ from Part (1), we can complete the proof using a technique similar to the one in Lemma 2.6.

(3) Let $\hat{D} \geq D_{JJ}$ be the diagonal matrix such that

$$\begin{aligned} \underset{\sim}{y}_J &= \underset{\sim}{c}_J + [\underset{\sim}{d}_J - \underset{\sim}{c}_J - \hat{D}^{-1} g_J(\underset{\sim}{x})] \\ &= \underset{\sim}{d}_J - \hat{D}^{-1} g_J(\underset{\sim}{x}). \end{aligned}$$

Let $\underset{\sim}{z} \equiv \hat{D}^{-1} g_J(\underset{\sim}{x})$. Then

$$\underset{\sim}{y}_J = \underset{\sim}{x}_J - \underset{\sim}{z}$$

and

$$\begin{aligned} \nu(\underset{\sim}{y}) - \nu(\underset{\sim}{x}) &= \tfrac{1}{2} \underset{\sim}{z}^t A_{JJ} \underset{\sim}{z} - \underset{\sim}{z}^t A_{JJ} \underset{\sim}{x}_J + \underset{\sim}{z}^t \underset{\sim}{b}_J \\ &= \tfrac{1}{2} \underset{\sim}{z}^t A_{JJ} \hat{D}^{-1} g_J(\underset{\sim}{x}) - \underset{\sim}{z}^t g_J(\underset{\sim}{x}) \\ &= -\underset{\sim}{z}^t (I - \tfrac{1}{2} A_{JJ} \hat{D}^{-1}) \, g_J(\underset{\sim}{x}) \, . \end{aligned}$$

It is easy to show that $I - (A_{JJ}\hat{D}^{-1})/2 \geq 0$, equality excluded.

Since $\underset{\sim}{z} > \underset{\sim}{0}$ and $g_J(\underset{\sim}{x}) > \underset{\sim}{0}$, we get $\nu(\underset{\sim}{y}) - \nu(\underset{\sim}{x}) < 0$. □

ALGORITHM 3.2. (Modified Pang's method). Set $\underset{\sim}{x} := \underset{\sim}{d}$ and compute $\underset{\sim}{g} := A\underset{\sim}{d} - \underset{\sim}{b}$.

Let $J := \{j \in N | g_j > 0\}$, $P := J$, $U := \bar{J}$ and $Z := \emptyset$.

Repeat until set U or P or J is empty:

1. Repeat until set U or J is empty:

   (a) Update $\underset{\sim}{x}_J := \underset{\sim}{c}_J + [\underset{\sim}{d}_J - \underset{\sim}{c}_J - D_{JJ}^{-1} \underset{\sim}{g}_J]_+$.

   (b) Let $K := \{k \in J | x_k = c_k\}$, $Z := Z \cup K$ and $P := P \cup (J \setminus K)$.

   (c) Update $\underset{\sim}{g}_U := \underset{\sim}{g}_U - A_{UJ}(\underset{\sim}{d}_J - \underset{\sim}{x}_J)$.

   (d) Redefine $J := \{j \in U | g_j > 0\}$. Let $U := U \setminus J$.

2. Reconstruct $\underset{\sim}{x}$ so that $\underset{\sim}{x}_U = \underset{\sim}{d}_U$, $\underset{\sim}{x}_Z = \underset{\sim}{c}_Z$ and $\underset{\sim}{x}_P = \underset{\sim}{y}$, where $\underset{\sim}{y}$ solves the lower bound quadratic program (3.1) (we may use Algorithm 2.3).

3. Set $K := \{k \in P | x_k = c_k\}$. Let $P := P \setminus K$ and $Z := Z \cup K$.

4. Compute $\underset{\sim}{g}_U := A_{UU}\underset{\sim}{d}_U + A_{UP}\underset{\sim}{x}_P + A_{UZ}\underset{\sim}{c}_Z - \underset{\sim}{b}_U$.

5. Redefine $J := \{j \in U | g_j > 0\}$. Let $P := P \cup J$ and $U := U \setminus J$. □

3.4. Problem with Non-finite Bounds

In this section, we consider the lower and upper bounds quadratic program

$$\min_{\underset{\sim}{x}} \frac{1}{2}\underset{\sim}{x}^t A \underset{\sim}{x} - \underset{\sim}{x}^t \underset{\sim}{b} \qquad (1.1)$$

$$\text{subject to} \quad \underset{\sim}{c} \leq \underset{\sim}{x} \leq \underset{\sim}{d} ,$$

where some components of the bounds are infinite. We show that we can compute finite bounds on all the variables.

LEMMA 4.1. Consider the quadratic program (1.1) with $\underset{\sim}{d} = \underset{\sim}{\infty}$. Assume $\underset{\sim}{c}_J = -\underset{\sim}{\infty}$ and $\underset{\sim}{c}_{\bar{J}} > -\underset{\sim}{\infty}$ for some nonempty subset J of N. Let $\underset{\sim}{x}$ be the solution vector. Then

$$\underset{\sim}{x}_J \geq A_{JJ}^{-1}(\underset{\sim}{b}_J - A_{J\bar{J}}\underset{\sim}{c}_{\bar{J}}) .$$

Proof. Define a vector $\underset{\sim}{p}$ with $\underset{\sim}{p}_J = A_{JJ}^{-1}(\underset{\sim}{b}_J - A_{J\bar{J}}\underset{\sim}{c}_{\bar{J}})$ and $\underset{\sim}{p}_{\bar{J}} = \underset{\sim}{c}_{\bar{J}}$. By construction, $\underset{\sim}{g}_J(\underset{\sim}{p}) = \underset{\sim}{0}$. Define another vector $\underset{\sim}{q}$ with $\underset{\sim}{q}_J = \underset{\sim}{p}_J$ and $\underset{\sim}{q}_{\bar{J}} = \underset{\sim}{x}_{\bar{J}}$. Since $\underset{\sim}{q}_{\bar{J}} \geq \underset{\sim}{p}_{\bar{J}}$, we get $\underset{\sim}{g}_J(\underset{\sim}{q}) \leq \underset{\sim}{g}_J(\underset{\sim}{p})$ from Lemma 2.3.

Also $g_J(\underset{\sim}{x}) = \underset{\sim}{0}$ by the optimality conditions (1.5). Thus,

$$\begin{aligned} A_{JJ}(\underset{\sim}{x}_J - \underset{\sim}{p}_J) &= A_{JJ}\underset{\sim}{x}_J + A_{J\bar{J}}\underset{\sim}{x}_{\bar{J}} - \underset{\sim}{b}_J - (A_{JJ}\underset{\sim}{p}_J + A_{J\bar{J}}\underset{\sim}{x}_{\bar{J}} - \underset{\sim}{b}_J) \\ &= g_J(\underset{\sim}{x}) - g_J(\underset{\sim}{q}) \\ &\geq 0 . \end{aligned}$$

Hence

$$\underset{\sim}{x}_J - \underset{\sim}{p}_J \geq \underset{\sim}{0} . \qquad \square$$

LEMMA 4.2. Consider the quadratic program (1.1) with $\underset{\sim}{c} = -\underset{\sim}{\infty}$. Assume $\underset{\sim}{d}_J = \underset{\sim}{\infty}$ and $\underset{\sim}{d}_{\bar{J}} < \underset{\sim}{\infty}$ for some nonempty subset J of N. Let $\underset{\sim}{x}$ be the solution vector. Then

$$\underset{\sim}{x}_J \leq A_{JJ}^{-1}(\underset{\sim}{b}_J - A_{J\bar{J}}\underset{\sim}{x}_{\bar{J}}) .$$

Proof. Similar to that of Lemma 4.1. $\square$

We now have the tools to handle the problem when both bounds have infinite components.

LEMMA 4.3. For problem (1.1), assume $\underset{\sim}{d}_J = \underset{\sim}{\infty}$ and $\underset{\sim}{d}_{\bar{J}} < \underset{\sim}{\infty}$ for some nonempty subset J of N. Let $\underset{\sim}{x}$ be the solution to (1.1) and $\underset{\sim}{y}$ be the solution to the lower bound quadratic program

$$\begin{aligned} \min_{\underset{\sim}{y}} \ & \frac{1}{2}\underset{\sim}{y}^t A_{JJ}\underset{\sim}{y} - \underset{\sim}{y}^t(\underset{\sim}{b}_J - A_{J\bar{J}}\underset{\sim}{d}_{\bar{J}}) \\ \text{subject to} \ & \underset{\sim}{y} \geq \underset{\sim}{c}_J . \end{aligned} \tag{4.1}$$

Then

$$\underset{\sim}{x}_J \leq \underset{\sim}{y} .$$

Proof. The inequality is trivial for those components of $\underset{\sim}{x}_J$ that are at the lower bound. Let $K := \{k \in J | x_k > c_k\}$. Then $\underset{\sim}{g}_K(\underset{\sim}{x}) = \underset{\sim}{0}$.

Now define a vector $\underset{\sim}{p}$ with $\underset{\sim}{p}_J = \underset{\sim}{y}$ and $\underset{\sim}{p}_{\bar{J}} = \underset{\sim}{d}_{\bar{J}}$. Hence $\underset{\sim}{g}_J(\underset{\sim}{p}) \geq \underset{\sim}{0}$ by the optimality conditions (1.5). Define another vector $\underset{\sim}{q}$ with $\underset{\sim}{q}_J = \underset{\sim}{y}$ and $\underset{\sim}{q}_{\bar{J}} = \underset{\sim}{x}_{\bar{J}}$. Since $\underset{\sim}{q}_{\bar{J}} \leq \underset{\sim}{p}_{\bar{J}}$, we get $\underset{\sim}{g}_J(\underset{\sim}{q}) \geq \underset{\sim}{g}_J(\underset{\sim}{p})$ by Lemma 2.3. It follows that $\underset{\sim}{g}_K(\underset{\sim}{q}) \geq 0$.

Let $L := J \setminus K$. Then $\underset{\sim}{x}_L = \underset{\sim}{c}_L \leq \underset{\sim}{p}_L$, or equivalently, $\underset{\sim}{x}_L - \underset{\sim}{p}_L \leq \underset{\sim}{0}$. Thus,

$$\begin{aligned} A_{KK}(\underset{\sim}{x}_K - \underset{\sim}{p}_K) &= (A_{KK}\underset{\sim}{x}_K + A_{KL}\underset{\sim}{x}_L + A_{K\bar{J}}\underset{\sim}{x}_{\bar{J}} - \underset{\sim}{b}_K) \\ &\quad - (A_{KK}\underset{\sim}{p}_K + A_{KL}\underset{\sim}{p}_L + A_{K\bar{J}}\underset{\sim}{x}_{\bar{J}} - \underset{\sim}{b}_K) - A_{KL}(\underset{\sim}{x}_L - \underset{\sim}{p}_L) \\ &= \underset{\sim}{g}_K(\underset{\sim}{x}) - \underset{\sim}{g}_K(\underset{\sim}{q}) - A_{KL}(\underset{\sim}{x}_L - \underset{\sim}{p}_L) \\ &\leq 0 \quad \text{because} \quad A_{KL} \leq 0 \ . \end{aligned}$$

Hence

$$\underset{\sim}{x}_K - \underset{\sim}{p}_K \leq \underset{\sim}{0} \ . \quad \square$$

LEMMA 4.4. For problem (1.1), assume $\underset{\sim}{c}_J = -\underset{\sim}{\infty}$ and $\underset{\sim}{c}_{\bar{J}} > -\underset{\sim}{\infty}$ for some nonempty subset J of N. Let $\underset{\sim}{x}$ be the solution to (1.1) and $\underset{\sim}{y}$ be the solution to the upper bound quadratic program

$$\begin{aligned} \min_{\underset{\sim}{y}} \ & \frac{1}{2}\underset{\sim}{y}^t A_{JJ}\underset{\sim}{y} - \underset{\sim}{y}^t(\underset{\sim}{b}_J - A_{J\bar{J}}\underset{\sim}{c}_{\bar{J}}) \\ \text{subject to} \ & \underset{\sim}{y} \leq \underset{\sim}{d}_J \ . \end{aligned} \tag{4.2}$$

Then

$$\underset{\sim}{x}_J \geq \underset{\sim}{y} \ .$$

Proof. Similar to that of Lemma 4.3. $\square$

We can solve the lower bound quadratic program (4.1) by first using Lemma 4.1 to compute finite lower bounds for those variables with

constraints and then applying Algorithm 2.3. The upper bound problem (4.2) can be solved in a similar fashion using Lemma 4.2 and Algorithm 2.3. Note that an upper bound problem is reduced to one with a lower bound if all its variables are negated.

3.5. Numerical Examples

We have chosen four representative problems to study the effectiveness of our "partial updating" technique. We use the conjugate gradient method as our matrix equations solver. Our programs were written in FORTRAN and run on an IBM 370/168 computer at the Stanford Linear Accelerator Center. The codes were compiled with optimization level 2 of the H EXTENDED compiler.

EXAMPLE 1. Let us consider the linear complementarity problem

$$\min_{\underset{\sim}{x}} \frac{1}{2} \underset{\sim}{x}^t A \underset{\sim}{x} - \underset{\sim}{x}^t \underset{\sim}{b} \qquad (1.2)$$

$$\text{subject to} \quad \underset{\sim}{x} \geq \underset{\sim}{0} .$$

The matrix A is chosen as

$$A = \begin{pmatrix} 2 & -1 & & \bigcirc \\ -1 & 2 & -1 & \\ & \ddots & \ddots & \ddots \\ & & -1 & 2 & -1 \\ \bigcirc & & & -1 & 2 \end{pmatrix}_{n \times n}$$

and the vector $\underset{\sim}{b}$ is generated by

$$b_i = 8 - 20r_i \qquad \text{for} \quad i = 1,2,\ldots, n ,$$

where r_i is a random number from a uniform distribution in the open interval $(0,1)$. The "partial updating" matrix C is chosen as

$$C = \begin{pmatrix} 2 & & \mathbf{0} \\ & 2 & \\ & \ddots & \\ \mathbf{0} & & 2 \end{pmatrix}_{n \times n} .$$

We define an iteration to be a sequence of "partial updates" followed by a solution of a matrix equation with coefficient matrix A_{PP}. The scalar $|P|$ gives the number of elements in the index set P at the end of an iteration. The other scalar u gives the number of "partial updates" in an iteration. Time is machine execution time in seconds.

n = 1000

Iteration	Algorithm 2.1	Algorithm 2.2
	|P|	u, |P|
1	382	2,456
2	499	2,524
3	535	1,539
4	539	1,542
5	542	
Time	2.67	2.40

n = 1500

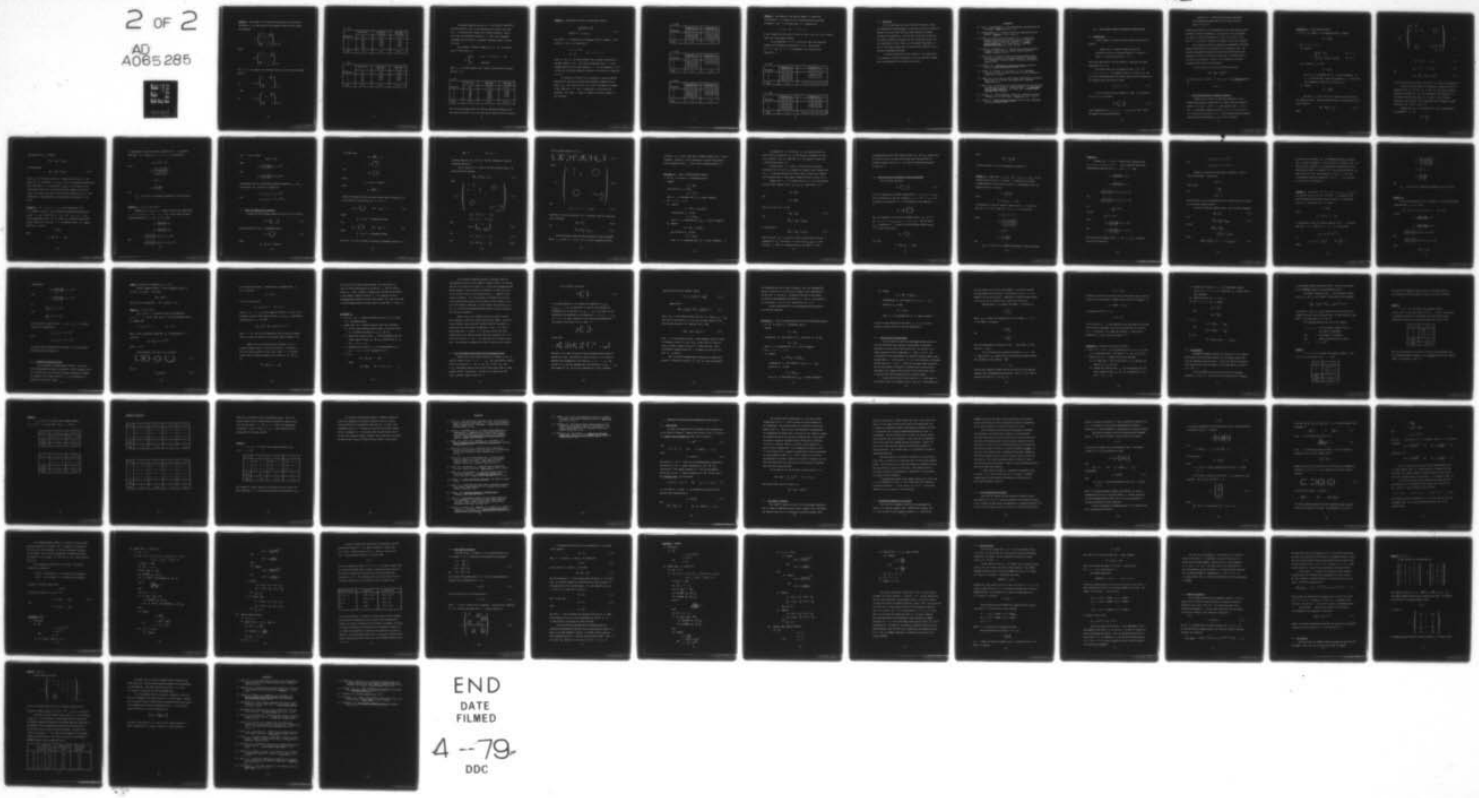
Iteration	Algorithm 2.1	Algorithm 2.2
	|P|	u, |P|
1	585	2,692
2	763	2,806
3	824	3,838
4	837	2,848
5	845	2,851
6	848	1,853
7	850	
8	852	
9	853	
Time	8.43	6.49

n = 2000

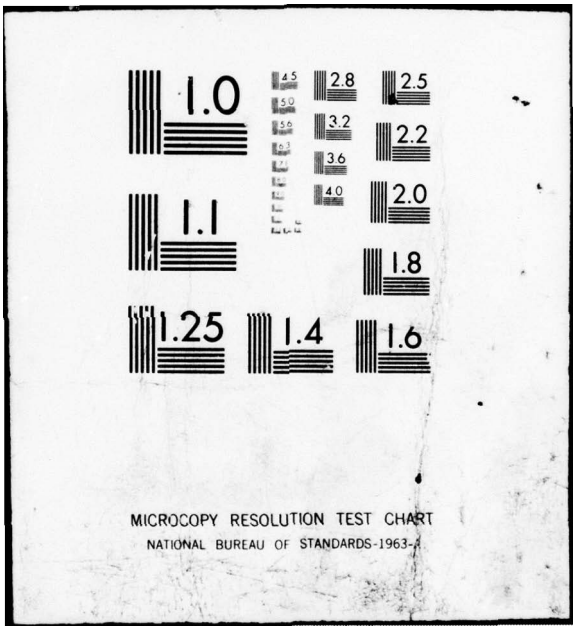
Iteration	Algorithm 2.1	Algorithm 2.2
	|P|	u, |P|
1	782	2,923
2	1020	2,1074
3	1098	3,1118
4	1119	2,1131
5	1128	2,1136
6	1133	1,1138
7	1135	
8	1137	
9	1138	
Time	11.79	9.92

AD-A065 285 STANFORD UNIV CALIF DEPT OF COMPUTER SCIENCE F/G 12/1
SPARSE AND PARALLEL MATRIX COMPUTATIONS.(U)
DEC 78 F T LUK
UNCLASSIFIED STAN-CS-78-685 DAHC04-75-G-0185
NL

2 OF 2
AD A065 285



END
DATE FILMED
4-79
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

EXAMPLE 2. This example is the same as the previous one, except that the matrix A has been chosen as the Laplacian 5-point finite difference operator:

$$A = \begin{pmatrix} B & -I & & \\ -I & B & -I & \\ & \ddots & \ddots & \ddots \\ & & -I & B & -I \\ & & & -I & B \end{pmatrix}_{m^2 \times m^2},$$

where

$$B = \begin{pmatrix} 4 & -1 & & \\ -1 & 4 & -1 & \\ & \ddots & \ddots & \ddots \\ & & -1 & 4 & -1 \\ & & & -1 & 4 \end{pmatrix}_{m \times m}.$$

Let $n = m^2$. We consider two different choices of the "partial updating" matrix:

(1) $$C = D \equiv \begin{pmatrix} 4 & & & \\ & 4 & & \\ & & \ddots & \\ & & & 4 \end{pmatrix}_{n \times n},$$

and

(2) $$C = T \equiv \begin{pmatrix} B & & & \\ & B & & \\ & & \ddots & \\ & & & B \end{pmatrix}_{n \times n}.$$

n = 900

Iteration	Algorithm 2.1	Algorithm 2.2 (C = D)	Algorithm 2.2 (C = T)
	|P|	u, |P|	u, |P|
1	349	2,416	2,416
2	450	2,462	2,462
3	463	1,465	1,465
4	465	1,466	1,466
5	466		
Time	7.54	6.73	6.70

n = 1600

Iteration	Algorithm 2.1	Algorithm 2.2 (C = D)	Algorithm 2.2 (C = T)
	|P|	u, |P|	u, |P|
1	620	2,730	2,730
2	798	2,828	1,828
3	837	1,842	1,842
4	842		
Time	17.08	15.29	15.41

We observe that both choices of C have produced essentially identical results. This is not surprising if we look at the index set J of the variables eligible for "partial updating." Rarely do we find two consecutive indices in J. Thus, the tridiagonal "updating" matrix reduces to a diagonal "updating" matrix in almost all cases.

We construct a different example with $n = 900$ and another choice of the vector $\underset{\sim}{b}$:

$$b_i = \begin{cases} -6r_i & \text{for } i = 301, 302, \ldots, 600 \\ 8r_i & \text{otherwise,} \end{cases}$$

where r_i is a random number from a uniform distribution in the open interval $(0,1)$.

$n = 900$

Iteration	Algorithm 2.1 |P|	Algorithm 2.2 (C = D) u, |P|	Algorithm 2.2 (C = T) u, |P|
1	300	2,307	2,307
2	366	2,380	3,386
3	396	2,401	2,405
4	406	1,406	1,407
5	407	1,407	
Time	24.97	25.08	23.80

This is also an example where the "partial updating" technique is not particularly efficient due to the very special structure of the vector $\underset{\sim}{b}$.

EXAMPLE 3. We address the lower and upper bounds problem

$$\min_{\underset{\sim}{x}} \frac{1}{2} \underset{\sim}{x}^t A \underset{\sim}{x} - \underset{\sim}{x}^t \underset{\sim}{b}$$
$$\text{subject to} \quad \underset{\sim}{0} \leq \underset{\sim}{x} \leq \underset{\sim}{d} . \tag{5.1}$$

The matrix A is chosen as the tridiagonal matrix in Example 1. The vectors $\underset{\sim}{b}$ and $\underset{\sim}{d}$ are generated by

$$b_i = 11 - 20r_i, \qquad \text{for } i = 1,2,\ldots, n,$$
$$d_i = 7s_i,$$

where r_i and s_i are random numbers from a uniform distribution in the open interval $(0,1)$. The "partial updating" matrix C is the diagonal matrix with the same diagonal as A. We use Algorithm 2.1 (2.2) to solve the lower bound quadratic program in an iteration of Algorithm 3.1 (3.2).

An iteration is defined to be a sequence of "partial updates" followed by a solution of a lower bound quadratic program with the matrix A_{PP}. The 3-tuple $(|Z|, |P|, |U|)$ gives the number of elements in the index sets Z, P and U, respectively, at the end of an iteration. The scalar u gives the number of "partial updates" in the iteration.

n = 1000

Iteration	Algorithm 3.1 (|Z|, |P|, |U|)	Algorithm 3.2 u, (|Z|, |P|, |U|)
1	(234,216,550)	3, (307,275,418)
2	(310,283,407)	1, (311,292,397)
3	(313,291,396)	1, (313,291,396)
Time	1.80	1.63

n = 1500

Iteration	Algorithm 3.1 (|Z| |P|, |U|)	Algorithm 3.2 u, (|Z|, |P|, |U|)
1	(366,324,810)	3, (489,396,615)
2	(492,408,600)	1, (495,416,589)
3	(495,415,590)	
4	(495,416,589)	
Time	4.13	2.09

n = 2000

Iteration	Algorithm 3.1 (|Z|, |P|, |U|)	Algorithm 3.2 u, (|Z|, |P|, |U|)
1	(452,452,1096)	3, (590,560,850)
2	(602,579,819)	2, (605,602,793)
3	(605,602,793)	1, (605,603,792)
4	(605,603,792)	
Time	7.51	5.01

EXAMPLE 4. This example is the same as Example 3, except that (1) the matrix A is chosen as the discretized Laplacian operator of Example 2, and (2) the upper bound $\underset{\sim}{d}$ is generated by

$$d_i = 3s_i \quad \text{for} \quad i = 1,2,\ldots,n .$$

We have changed the upper bound so that the three index sets are of roughly equal size in the final solution.

We use Algorithm 2.1 (2.2) to solve the lower bound quadratic program in an iteration of Algorithm 3.1 (3.2). Both choices: (1) $C = D$ and (2) $C = T$ (see Example 2) are considered for Algorithm 2.2.

n = 900

	Algorithm 3.1	Algorithm 3.2
Iteration	$(\|Z\|,\|P\|,\|U\|)$	$u,(\|Z\|,\|P\|,\|U\|)$
1	(274,168,458)	3,(307,223,370)
2	(325,277,298)	2,(325,294,281)
3	(325,294,281)	1,(325,296,279)
4	(325,296,279)	
Time	6.79	5.97 (C = D), 5.94 (C = T)

n = 1600

	Algorithm 3.1	Algorithm 3.2
Iteration	$(\|Z\|,\|P\|,\|U\|)$	$u,(\|Z\|,\|P\|,\|U\|)$
1	(478,277,845)	3,(552,430,618)
2	(582,487,531)	2,(585,524,497)
3	(585,520,495)	1,(585,526,489)
4	(585,526,489)	
Time	17.72	15.35 (C = D), 15.28 (C = T)

3.6. Conclusion

It is evident from the work of Cottle and Goheen [4] that the algorithms of Chandrasekaran [2] and Pang [8] are very effective schemes for solving lower and upper bounds quadratic programs associated with M-matrices. We have seen in the last section how our "partial updating" technique can cut the execution time of the two algorithms by 10-50%. Thus, our new schemes (Algorithms 2.2 and 3.2) are highly competitive for solving this important class of quadratic programming problems.

Our other contribution of this chapter is the introduction of a technique to handle the special case of the quadratic program when some components of the bounds are infinite.

REFERENCES

[1] Ceá, J., and Glowinski, R., "Sur des methodes d'optimisation par relaxation," RAIRO R-3 (1973), 5-32.

[2] Chandrasekaran, R., "A special case of the complementary pivot problem," Opsearch 7 (1970), 263-268.

[3] Concus, P., Golub. G.H., and O'Leary, D.P., "A generalized conjugate gradient method for the numerical solution of elliptic partial differential equations," in Sparse Matrix Computations (J. R. Bunch and D.J. Rose, Ed.) Academic Press, New York (1976), 309-332.

[4] Cottle, R. and Goheen, M., "A special class of large quadratic programs," Report SOL-76-7, Systems Optimization Lab., Stanford University (1976).

[5] Cryer, C.W., "A survey of trial free-boundary methods for the numerical solution of free-boundary problems," MRC Report 1693, Mathematics Research Center, University of Wisconsin, Madison (1976).

[6] Fröberg, C.E., Introduction to Numerical Analysis, Addison-Wesley, Reading, Massachusetts (1965).

[7] Levati, G., Scarpini, F., and Volpi, G., "Sul trattemento numerico di alcuni problemi variazionali di tipo unilaterale," L.A.N. Pub. 82 (1974).

[8] Pang, J.S., "On a class of least-element complementarity problems," Report SOL-76-10, Systems Optimization Lab., Stanford University (1976).

[9] Reid, J.K., "On the method of conjugate gradients for the solution of large sparse systems of linear equations," in Large Sparse Sets of Linear Equations, (J.K. Reid, Ed.), Academic Press, New York (1971), 231-254.

[10] Scarpini, F., "Some algorithms solving the unilateral Dirichlet problems with two constraints," Calcolo 12 (1975), 113-149.

[11] Varga, R.S., Matrix Iterative Analysis, Prentice Hall, Englewood Cliffs, New Jersey (1962).

IV. A BLOCK LANCZOS METHOD FOR COMPUTING SINGULAR VALUES

4.1. Introduction

In this chapter, we construct a Block Lanczos method for the problem:

> Compute the k greatest singular values and associated vectors of a large and sparse $m \times n$ matrix A, where k is much smaller than m or n,

which finds applications in factor analysis, regression and image enhancement (cf. [4]).

We assume without loss of generality that $m \geq n$. For $i = 1,2,\ldots,n$, let σ_i be a singular value of A, and let $\underset{\sim}{u}_i$ and $\underset{\sim}{v}_i$ be the corresponding left and right singular vectors, respectively. The singular values are ordered so that

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \,. \tag{1.1}$$

Let us exploit an idea of Lanczos [8, Chap. 3] and consider the $(m+n) \times (m+n)$ matrix

$$\tilde{A} = \begin{pmatrix} 0 & A \\ A^t & 0 \end{pmatrix}, \tag{1.2}$$

whose eigenvalues are $\pm\,\sigma_1, \pm\,\sigma_2, \ldots, \pm\,\sigma_n$, plus $(m-n)$ zeros. We address the equivalent problem:

Compute the k algebraically greatest eigenvalues and corresponding eigenvectors of the large and sparse matrix $\tilde{A}$ of (1.2).

An efficient scheme for this eigenproblem is the Block Lanczos method developed by several researchers, in particular, Cullum and Donath ([1] and [2]), Golub and Underwood ([7] and [14]), Lewis [9] and Ruhe [11]. We choose to consider the variant of Golub and Underwood.

We are going to present a theoretical development of the Block Lanczos method and give two theorems on its convergence rate. The practical implementation aspects are then discussed and particular attention is paid to the choice of block size. We believe that all our results are original. In fact, we are unaware of any other procedures which solve the same problem.

In this chapter, we use the Euclidean vector norm

$$\|\underset{\sim}{x}\| = \|\underset{\sim}{x}\|_2 = (\underset{\sim}{x}^t \underset{\sim}{x})^{1/2} ,$$

and refer to an $n \times b$ matrix X with $n \geq b$ as an <u>orthonormal</u> matrix if

$$X^t X = I .$$

4.2. <u>Block Lanczos Method for Symmetric Matrices</u>

Suppose that we desire accurate approximations to the k algebraically greatest eigenvalues of a large, sparse and symmetric matrix B of order ℓ, where k is much smaller than ℓ. Let b and s be two given integers such that $b \geq 1$, $s \geq 2$ and $bs \leq \ell$. It is usually the case that $bs \ll \ell$. We can define the Block Lanczos method of Golub and Underwood ([7] and [14]) as follows.

ALGORITHM 2.1. (Block Lanczos method)

1. Let X_1 be a given $\ell \times b$ orthonormal matrix. Compute

$$M_1 := X_1^t B X_1 .$$

2. For $i = 2,3,\ldots,s$, do

(a) Compute

$$Z_i := \begin{cases} BX_1 - X_1 M_1 & \text{for } i = 2, \\ BX_{i-1} - X_{i-1} M_{i-1} - X_{i-2} R_{i-1}^t & \text{for } i \geq 3. \end{cases}$$

(b) Factorize Z_i so that

$$Z_i := X_i R_i,$$

where X_i is orthonormal and R_i is upper triangular. If Z_i is rank deficient, choose the columns of X_i so that they are orthogonal to those of all previous X_j's.

(c) Compute

$$M_i := X_i^t B X_i . \qquad \square$$

The value b is thus the order of the block, and the value s the number of blocks. The Block Lanczos method is characterized by the matrix equation

$$B\bar{X}_s = \bar{X}_s \bar{M}_s + \bar{Z}_{s+1} , \qquad (2.1)$$

where

$$\bar{M}_s = \begin{pmatrix} M_1 & R_2^t & & & & \\ R_2 & M_2 & R_3^t & & \bigcirc & \\ & \cdot & \cdot & \cdot & & \\ & & \cdot & \cdot & \cdot & \\ & & & \cdot & \cdot & \cdot \\ & \bigcirc & & & R_{s-1} \; M_{s-1} & R_s^t \\ & & & & R_s & M_s \end{pmatrix} , \qquad (2.2)$$

$$\bar{X}_s = (X_1, X_2, \ldots , X_s) , \qquad (2.3)$$

$$\bar{Z}_{s+1} = (0, \ldots , 0, Z_{s+1}) \qquad (2.4)$$

and

$$Z_{s+1} = BX_s - X_s M_s - X_{s-1} R_s^t \quad . \qquad (2.5)$$

We can prove that the columns of the matrix $\bar{X}_s$ form an orthonormal set.

The Block Lanczos method thus generates a symmetric block tridiagonal matrix $\bar{M}_s$ of order bs. As the R_i's are upper triangular matrices, the matrix $\bar{M}_s$ is also a band matrix with bandwidth $2b + 1$. We consider only the case where bs is small, so that standard techniques can be applied to the computation of the eigenvalue decomposition of $\bar{M}_s$ (see [12] and [15]).

Suppose that y_i is an eigenvector of $\bar{M}_s$ corresponding to the eigenvalue μ_i. Let

$$x_i = \bar{X}_s y_i \, .$$

From equation (2.1), we obtain

$$B\underset{\sim}{x}_i = \mu_i \underset{\sim}{x}_i + \bar{Z}_{s+1} \underset{\sim}{y}_i \ ,$$

or equivalently,

$$B\underset{\sim}{x}_i = \mu_i \underset{\sim}{x}_i + Z_{s+1} \underset{\sim}{f}_i \ , \tag{2.6}$$

where $\underset{\sim}{f}_i$ is the vector of order b consisting of the last b components of $\underset{\sim}{y}_i$. Therefore, μ_i and $\underset{\sim}{x}_i$ would be an eigenvalue-eigenvector pair of the matrix B if we had that $Z_{s+1}\underset{\sim}{f}_i = \underset{\sim}{0}$. The power of the Block Lanczos method lies in the fact that the elements of the vector $Z_{s+1}\underset{\sim}{f}_i$ are usually very small for the extreme eigenvalues of $\bar{M}_s$. This observation is substantiated by a theorem given by Underwood [14, pp. 37-38].

THEOREM 2.1. Let $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_\ell$ be the eigenvalues of an $\ell \times \ell$ symmetric matrix B with corresponding normalized eigenvectors $g_1, g_2, \cdots, g_\ell$. Assume that $\lambda_b > \lambda_{b+1}$. Let $\mu_1 \geq \mu_2 \geq \cdots \geq \mu_{bs}$ be the eigenvalues of the $bs \times bs$ matrix $\bar{M}_s$ generated by the Block Lanczos method with an initial $\ell \times b$ orthonormal matrix X_1. Suppose that the $b \times b$ matrix

$$W = G_1^t X_1 \ ,$$

where

$$G_1 = (g_1, g_2, \cdots, g_b) \ ,$$

is nonsingular so that its smallest singular value τ is positive. Note that $\tau \leq 1$. Then, for $i = 1, 2, \ldots, b$, we have that

$$\lambda_i \geq \mu_i \geq \lambda_i - \epsilon_i^2 ,$$

where

$$\epsilon_i^2 = \frac{(\lambda_i - \lambda_\ell) \tan^2 \theta}{T_{s-1}^2\left(\frac{1 + \gamma_i}{1 - \gamma_i}\right)} ,$$

$$\theta = \cos^{-1} \tau ,$$

$$\gamma_i = \frac{\lambda_i - \lambda_{b+1}}{\lambda_i - \lambda_\ell}$$

and

T_{s-1} is the (s-1)-st Chebyshev polynomial of the first kind.

EXAMPLE 2.1. (cf. [14, pp. 43-44])

Suppose that B is an $\ell \times \ell$ symmetric matrix with eigenvalues $\lambda_1 = 1.0$, $\lambda_2 = 0.9$, $\lambda_3 = 0.5$, $\ldots$, $\lambda_\ell = 0.0$. Let us apply the Block Lanczos method with $b = 2$ and $s = 10$. Then

$$\gamma_1 = \frac{1.0 - 0.5}{1.0 - 0.0} = \frac{1}{2} ,$$

$$\gamma_2 = \frac{0.9 - 0.5}{0.9 - 0.0} = \frac{4}{9} ,$$

$$T_9\left(\frac{1 + \gamma_1}{1 - \gamma_1}\right) = T_9(3) \doteq 3.9 \times 10^6$$

and

$$T_9\left(\frac{1 + \gamma_2}{1 - \gamma_2}\right) = T_9(2.6) \doteq 9.8 \times 10^5 .$$

Let $\tau = 0.04$, so that

$$\tan^2 \theta = 624 .$$

Now,

$$\epsilon_1^2 \doteq \frac{1.0 \times 624}{1.5 \times 10^{13}} \doteq 4.1 \times 10^{-11}$$

and

$$\epsilon_2^2 \doteq \frac{0.9 \times 624}{9.6 \times 10^{11}} \doteq 5.8 \times 10^{-10} .$$

Consequently, the two algebraically greatest eigenvalues μ_1 and μ_2 of the matrix $\bar{M}_s$ satisfy the inequalities

$$\lambda_1 \geq \mu_1 \geq \lambda_1 - 4.1 \times 10^{-11}$$

and

$$\lambda_2 \geq \mu_2 \geq \lambda_2 - 5.8 \times 10^{-10} .$$

4.3. Block Bidiagonalization Algorithm

We apply the Block Lanczos method to the $(m+n) \times (m+n)$ matrix

$$\tilde{A} = \begin{pmatrix} 0 & A \\ A^t & 0 \end{pmatrix} \qquad (1.2)$$

with the initial $(m+n) \times b$ orthonormal matrix

$$X_1 = \begin{pmatrix} 0 \\ Q_1 \end{pmatrix} , \qquad (3.1)$$

where

Q_1 is an $n \times b$ matrix.

It follows that

$$M_1 = X_1^{t\tilde{}} \tilde{A} X_1 = 0 ,$$

$$Z_1 = \begin{pmatrix} AQ_1 \\ 0 \end{pmatrix}$$

and

$$X_2 = \begin{pmatrix} P_1 \\ 0 \end{pmatrix},$$

where

P_1 is an $m \times b$ matrix.

Thus,

$$M_2 = X_2^{t\tilde{}} \tilde{A} X_2 = 0 .$$

Using the relations defining the Block Lanczos method (Algorithm 2.1), we can prove by induction that, for $j = 1,2,\ldots$,

$$X_{2j-1} = \begin{pmatrix} 0 \\ Q_j \end{pmatrix} \qquad \text{and} \qquad M_{2j-1} = 0 , \tag{3.2}$$

where

Q_j is an $n \times b$ orthonormal matrix,

and

$$X_{2j} = \begin{pmatrix} P_j \\ 0 \end{pmatrix} \qquad \text{and} \qquad M_{2j} = 0 , \tag{3.3}$$

where

P_j is an $m \times b$ orthonormal matrix.

Since the X_j's form a sequence of mutually orthonormal matrices, i.e.

$$X_i^t X_j = 0 \qquad \text{for } i \neq j ,$$

we deduce that the P_j's and Q_j's form two sequences of mutually orthonormal matrices.

Let us carry out $2s$ steps of the Block Lanczos scheme. We obtain the matrix equation

$$\tilde{A}\bar{X}_{2s} = \bar{X}_{2s}T_{2s} + \bar{Z}_{2s} , \tag{3.4}$$

where

$$T_{2s} = \begin{pmatrix} 0 & R_2^t & & & & \\ R_2 & 0 & R_3^t & & & \\ & \ddots & \ddots & \ddots & & \\ & & \ddots & \ddots & \ddots & \\ & & & R_{2s-1} & 0 & R_{2s}^t \\ & & & & R_{2s} & 0 \end{pmatrix} , \tag{3.5}$$

$$\bar{X}_{2s} = (X_1, X_2, \ldots , X_{2s}),$$

$$\bar{Z}_{2s} = (0, \ldots , 0, Z_{2s+1})$$

and

$$Z_{2s+1} = \begin{pmatrix} 0 \\ A^t P_s - Q_s R_{2s}^t \end{pmatrix} . \tag{3.6}$$

Let

$$\bar{P}_s = (P_1, P_2, \ldots , P_s) \tag{3.7}$$

and

$$\bar{Q}_s = (Q_1, Q_2, \ldots , Q_s) . \tag{3.8}$$

We can rewrite equation (3.4) as

$$\begin{pmatrix} 0 & A \\ A^t & 0 \end{pmatrix}\begin{pmatrix} \bar{P}_s & 0 \\ 0 & \bar{Q}_s \end{pmatrix} = \begin{pmatrix} \bar{P}_s & 0 \\ 0 & \bar{Q}_s \end{pmatrix}\begin{pmatrix} 0 & J_s \\ J_s^t & 0 \end{pmatrix} + \begin{pmatrix} 0 & 0 \\ \hat{Z}_{2s+1} & 0 \end{pmatrix}, \qquad (3.9)$$

where

$$J_s = \begin{pmatrix} R_2 & R_3^t & & & & \\ & R_4 & R_5^t & & & \\ & & \ddots & \ddots & & \\ & & & \ddots & \ddots & \\ & & & & R_{2s-2} & R_{2s-1}^t \\ & & & & & R_{2s} \end{pmatrix} \qquad (3.10)$$

and

$$\hat{Z}_{2s+1} = (0, \ldots, 0, Z_{2s+1}) . \qquad (3.11)$$

Furthermore, the matrix equation (3.9) is reducible into two lower-order matrix equations:

$$A\bar{Q}_s = \bar{P}_s J_s \qquad (3.12)$$

and

$$A^t\bar{P}_s = \bar{Q}_s J_s^t + \hat{Z}_{2s+1} . \qquad (3.13)$$

The Block Lanczos method therefore generates a block bidiagonal matrix J_s of order bs. As the R_i's are upper triangular matrices,

the matrix J_s is also a band upper triangular matrix with b super-diagonals. Therefore, we have constructed a scheme that produces a block bidiagonal matrix J_s from a given rectangular matrix A.

ALGORITHM 3.1. (Block Bidiagonalization method).

1. Let Q_1 be a given $n \times b$ orthonormal matrix.

   Compute

   $$W_1 := AQ_1 ,$$

   and factorize W_1 so that

   $$W_1 := P_1 R_2 ,$$

   where P_1 is orthonormal and R_2 is upper triangular.

2. For $i = 2,3,\ldots,s$, do

   (a) Compute

   $$Z_i := A^t P_{i-1} - Q_{i-1} R_{2i-2}^t ,$$

   and factorize Z_i so that

   $$Z_i := Q_i R_{2i-1} ,$$

   where Q_i is orthonormal and R_{2i-1} is upper triangular.

   (b) Compute

   $$W_i := AQ_i - P_{i-1} R_{2i-1}^t ,$$

   and factorize W_i so that

   $$W_i := P_i R_{2i} ,$$

   where P_i is orthonormal and R_{2i} is upper triangular. □

In Algorithm 3.1, if the matrix Z_i were rank deficient, we would choose the columns of Q_i so that they are orthogonal to those of all previous Q_j's (cf. Algorithm 2.1). The remedy is similar for a rank-deficient matrix W_i.

We assume that bs is small, so that we can use standard techniques (see [3] and [6]) to compute the singular value decomposition of J_s. A thorough discussion of various ways to compute the singular value decomposition of a band upper triangular matrix is given in [5].

Suppose that μ_i is a singular value of J_s with corresponding left and right singular vectors $\underset{\sim}{w}_i$ and $\underset{\sim}{z}_i$, respectively. Let

$$\underset{\sim}{p}_i = \bar{P}_s \underset{\sim}{w}_i$$

and

$$\underset{\sim}{q}_i = \bar{Q}_s \underset{\sim}{z}_i \ .$$

From (3.12) and (3.13), we get

$$A\underset{\sim}{q}_i = \mu_i \underset{\sim}{p}_i \tag{3.14}$$

and

$$A^t \underset{\sim}{p}_i = \mu_i \underset{\sim}{q}_i + \hat{Z}_{2s+1} \underset{\sim}{w}_i \ ,$$

or equivalently,

$$A^t \underset{\sim}{p}_i = \mu_i \underset{\sim}{q}_i + Z_{2s+1} \underset{\sim}{h}_i \ , \tag{3.15}$$

where the vector $\underset{\sim}{h}_i$ is a vector of order b consisting of the last b components of $\underset{\sim}{w}_i$. Accordingly, if we had that $Z_{2i+1} \underset{\sim}{h}_i = \underset{\sim}{0}$, then the value μ_i would be a singular value of the matrix A with

corresponding left and right singular vectors $\underset{\sim}{p}_i$ and $\underset{\sim}{q}_i$, respectively. In the next section, we give error bounds which indicate that the greatest singular values of J_s are usually accurate approximations to those of A.

4.4. Error Bounds For the Singular Value Approximations

Let us consider the matrix

$$\tilde{A} = \begin{pmatrix} 0 & A \\ A^t & 0 \end{pmatrix} . \tag{1.2}$$

Its (b+1) algebraically greatest eigenvalues are $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{b+1}$, and its algebraically smallest eigenvalue is $-\sigma_1$. For $i = 1,2,\ldots,b$, the normalized eigenvector of $\tilde{A}$ corresponding to the eigenvalue σ_i is

$$\underset{\sim}{g}_i = \frac{1}{\sqrt{2}} \begin{pmatrix} \underset{\sim}{u}_i \\ \underset{\sim}{v}_i \end{pmatrix} .$$

Now, the eigenvalues of the block tridiagonal matrix T_{2s} of (3.5) are $\mu_1 \geq \mu_2 \geq \cdots \geq \mu_{bs} \geq -\mu_{bs} \geq \cdots \geq -\mu_2 \geq -\mu_1$. But the matrix T_{2s} is generated by 2s steps of the Block Lanczos method applied to $\tilde{A}$ with initial matrix

$$X_1 = \begin{pmatrix} 0 \\ Q_1 \end{pmatrix} .$$

If we let

$$G = (\underset{\sim}{g}_1, \underset{\sim}{g}_2, \ldots, \underset{\sim}{g}_b) ,$$

then

$$G_1^t X_1 = \frac{1}{\sqrt{2}} V_1^t Q_1 .$$

The next theorem is a direct consequence of Theorem 2.1.

THEOREM 4.1. Assume that $\sigma_b > \sigma_{b+1}$. Let $\mu_1 \geq \mu_2 \geq \cdots \geq \mu_{bs}$ be the singular values of the bs × bs matrix J_s generated by the Block Bidiagonalization method with an initial n × b orthonormal matrix Q_1. Suppose that the b × b matrix

$$W = \frac{1}{\sqrt{2}} V_1^t Q_1 ,$$

where

$$V_1 = (\underset{\sim}{v}_1, \underset{\sim}{v}_2, \cdots, \underset{\sim}{v}_b),$$

is nonsingular so that its smallest singular value τ is positive. Note that $\tau \leq 1/\sqrt{2}$. Then, for $i = 1,2,\ldots, b$, we have that

$$\sigma_i \geq \mu_i \geq \sigma_i - \epsilon_i^2 ,$$

where

$$\epsilon_i^2 = \frac{(\sigma_i - \sigma_1) \tan^2 \theta}{T_{2s-1}^2\left(\frac{1 + \gamma_i}{1 - \gamma_i}\right)} ,$$

$$\theta = \cos^{-1} \tau$$

$$\gamma_i = \frac{\sigma_i - \sigma_{b+1}}{\sigma_i + \sigma_1}$$

and

T_{2s-1} is the (2s-1)-st Chebyshev polynomial of the first kind.

EXAMPLE 4.1.

Suppose that A is an $m \times n$ matrix with singular values $\sigma_1 = 1.0$, $\sigma_2 = 0.9$, $\sigma_3 = 0.5$, ... Let us apply the Block Bidiagonalization algorithm with $b = 2$ and $s = 5$. Then

$$r_1 = \frac{1.0 - 0.5}{1.0 + 1.0} = 0.25 ,$$

$$r_2 = \frac{0.9 - 0.5}{0.9 + 1.0} \doteq 0.21 ,$$

$$T_9\left(\frac{1 + r_1}{1 - r_1}\right) \doteq T_9(1.67) \doteq 1.0 \times 10^4$$

and

$$T_9\left(\frac{1 + r_2}{1 - r_2}\right) \doteq T_9(1.53) \doteq 3.7 \times 10^3 .$$

Let

$$\tau = \frac{1}{\sqrt{2}} \times 0.04 ,$$

so that

$$\tan^2 \theta \doteq 1249 .$$

Thus,

$$\epsilon_1^2 \doteq \frac{2.0 \times 1249}{1.0 \times 10^8} \doteq 2.5 \times 10^{-5}$$

and

$$\epsilon_2^2 \doteq \frac{1.9 \times 1249}{1.37 \times 10^7} \doteq 1.7 \times 10^{-4} .$$

The two greatest singular values μ_1 and μ_2 of J_s therefore satisfy the inequalities

$$\sigma_1 \geq \mu_1 \geq \sigma_1 - 2.5 \times 10^{-5}$$

and

$$\sigma_2 \geq \mu_2 \geq \sigma_2 - 1.7 \times 10^{-4} .$$

However, we suspect that the bounds of Theorem 4.1 may be gross overestimates. Suppose that

$$Q_1 = V_1 .$$

Then we have

$$\tau = \frac{1}{\sqrt{2}}$$

and

$$\tan \theta = 1 .$$

The last value is quite unsatisfactory for an initial matrix consisting of the correct singular vectors.

We seek to construct tighter bounds. From the matrix equations

$$A\bar{Q}_s = \bar{P}_s J_s \tag{3.12}$$

and

$$A^t\bar{P}_s = \bar{Q}_s J_s^t + \hat{Z}_{2s+1} , \tag{3.13}$$

we get

$$\begin{aligned} A^tA\bar{Q}_s &= A^t\bar{P}_s J_s \\ &= \bar{Q}_s J_s^t J_s + \hat{Z}_{2s+1} J_s . \end{aligned}$$

Thus,

$$A^tA\bar{Q}_s = \bar{Q}_s J_s^t J_s + (0, \ldots , 0, Z_{2s+1} R_{2s}) . \tag{4.1}$$

We observe that the matrix $\bar{Q}_s$ is orthonormal and that the matrix $J_s^t J_s$ is block tridiagonal. It can be proved (cf. Chapter 2 and [14]) that the matrix equation (4.1) characterizes an application of the Block Lanczos method to the matrix A^tA with the initial matrix Q_1. Since the matrix A^tA has eigenvalues $\sigma_1^2 \geq \sigma_2^2 \geq \cdots \geq \sigma_n^2$ and corresponding normalized eigenvectors $\underset{\sim}{v}_1, \underset{\sim}{v}_2, \ldots, \underset{\sim}{v}_n$, we obtain the next result from Theorem 2.1.

THEOREM 4.2. Assume that $\sigma_b > \sigma_{b+1}$. Let $\mu_1 \geq \mu_2 \geq \cdots \geq \mu_{bs}$ be the singular values of the $bs \times bs$ matrix J_s generated by the Block Bidiagonalization method with an initial $n \times b$ orthonormal matrix Q_1. Suppose that the $b \times b$ matrix

$$W = V_1^t Q_1 ,$$

where

$$V_1 = (\underset{\sim}{v}_1, \underset{\sim}{v}_2, \ldots, \underset{\sim}{v}_b) ,$$

is nonsingular so that its smallest singular value τ is positive. Note that $\tau \leq 1$. Then, for $i = 1,2,\ldots,b$, we have that

$$\sigma_i \geq \mu_i \geq (\sigma_i^2 - \epsilon_i^2)^{1/2} ,$$

or

$$\sigma_i \geq \mu_i \geq \sigma_i - \frac{1}{2\sigma_i} \epsilon_i^2 \qquad \text{for } \frac{\epsilon_i^2}{\sigma_i^2} < 1 ,$$

where

$$\epsilon_i^2 = \frac{(\sigma_i^2 - \sigma_n^2)\tan^2\theta}{T_{s-1}^2\left(\frac{1+\gamma_i}{1-\gamma_i}\right)} \quad ,$$

$$\theta = \cos^{-1}\tau \ ,$$

$$\gamma_i = \frac{\sigma_i^2 - \sigma_{b+1}^2}{\sigma_i^2 - \sigma_n^2}$$

and

T_{s-1} is the (s-1)-st Chebyshev polynomial of the first kind.

EXAMPLE 4.2.

We use the same given data as in Example 4.1, with the additional assumption that $\sigma_n = 0.0$. Thus,

$$\sigma_1^2 = 1.00,\ \sigma_2^2 = 0.81,\ \sigma_3^2 = 0.25,\ \ldots\ ,\ \sigma_n^2 = 0.00 \ ,$$

so that

$$\gamma_1 = \frac{1.00 - 0.25}{1.00 - 0.00} = 0.75 \ ,$$

$$\gamma_2 = \frac{0.81 - 0.25}{0.81 - 0.00} \doteq 0.69 \ ,$$

$$T_4\left(\frac{1+\gamma_1}{1-\gamma_1}\right) = T_4(7) \doteq 1.88 \times 10^4 \ .$$

and

$$T_4\left(\frac{1+\gamma_2}{1-\gamma_2}\right) \doteq T_4(5.45) \doteq 6.82 \times 10^3 \ .$$

Also,

$$\tan^2\theta = 624.$$

Consequently,

$$\epsilon_1^2 \doteq \frac{1.00 \times 624}{3.55 \times 10^8} \doteq 1.7 \times 10^{-6}$$

and

$$\epsilon_2^2 \doteq \frac{0.81 \times 624}{4.65 \times 10^7} \doteq 1.1 \times 10^{-5} .$$

As

$$\frac{1}{2\sigma_1} \epsilon_1^2 \doteq 9.0 \times 10^{-7}$$

and

$$\frac{1}{2\sigma_2} \epsilon_2^2 \doteq 6.8 \times 10^{-6} ,$$

the two greatest singular values μ_1 and μ_2 of J_s therefore satisfies the inequalities

$$\sigma_1 \geq \mu_1 \geq \sigma_1 - 9.0 \times 10^{-7}$$

and

$$\sigma_2 \geq \mu_2 \geq \sigma_2 - 6.8 \times 10^{-6} .$$

We observe that the bounds given by Theorem 4.2 are much smaller than those given by Theorem 4.1.

4.5. Iterating to Improve Accuracy

Let us restate our computational procedure. We use the Block Bidiagonalization method to generate a block bidiagonal matrix J_s of small order, and then apply standard techniques to compute the singular value decomposition of J_s. Our convergence test depends on the next two lemmas.

LEMMA 5.1 (Weinstein's Inequality) [10, p. 56].

Given a symmetric matrix B and a normalized vector $\underset{\sim}{x}$, if there is a scalar μ such that

$$\|B\underset{\sim}{x} - \mu\underset{\sim}{x}\| \leq \delta ,$$

then there is an eigenvalue λ of B within δ of μ.

LEMMA 5.2. [10, pp. 59-60].

Let B be an $\ell \times \ell$ symmetric matrix with eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_\ell$ and let $\|B\underset{\sim}{x} - \lambda\underset{\sim}{x}\| \leq \delta$ for some normalized vector $\underset{\sim}{x}$. Suppose that

$$|\lambda - \lambda_i| \geq d > 0 \qquad \text{for } i \neq j .$$

Then B has a normalized eigenvector $\underset{\sim}{y}_j$ corresponding to λ_j such that

$$\|\underset{\sim}{x} - \underset{\sim}{y}_j\| \leq \gamma(1 + \gamma^2)^{1/2} ,$$

where $\gamma = \delta/d$.

From equations (3.14) and (3.15), we obtain

$$\begin{pmatrix} 0 & A \\ A^t & 0 \end{pmatrix} \begin{pmatrix} \underset{\sim}{p}_i \\ \underset{\sim}{q}_i \end{pmatrix} = \mu_i \begin{pmatrix} \underset{\sim}{p}_i \\ \underset{\sim}{q}_i \end{pmatrix} + \begin{pmatrix} \underset{\sim}{0} \\ Z_{2s+1}\underset{\sim}{h}_i \end{pmatrix} . \tag{5.1}$$

Thus, if

$$\|Z_{2s+1}\underset{\sim}{h}_i\| \leq \delta \tag{5.2}$$

for some given tolerance δ, then there is a singular value σ_j of A such that

$$|\mu_i - \sigma_j| \leq \delta .$$

If it is also true that

$$|\mu_i - \sigma_k| \geq d > 0 \qquad \text{for } k \neq j ,$$

where $\sigma_1, \sigma_2, \ldots, \sigma_n$ are the singular values of A, then A has normalized left and right singular vectors $\underset{\sim}{u}_j$ and $\underset{\sim}{v}_j$, respectively, corresponding to σ_j such that

$$\|\underset{\sim}{p}_i - \underset{\sim}{u}_j\|^2 + \|\underset{\sim}{q}_i - \underset{\sim}{v}_j\|^2 \leq \gamma^2(1 + \gamma^2) ,$$

where $\gamma = \delta/d$. For the more complicated case of multiple singular values, we refer the reader to an excellent paper of Stewart [13].

Suppose that our procedure has not computed all the k greatest singular values to the desired accuracy. In Theorem 4.2, the error bounds contain the term $\tan\theta$, where θ is the angle between the two subspaces spanned by the columns of V_1 and Q_1. Let

$$\hat{Q}_1 = (\underset{\sim}{q}_1, \underset{\sim}{q}_2, \ldots, \underset{\sim}{q}_b) .$$

The results of the last section indicate that the matrix $\hat{Q}_1$ can be a better approximation to the matrix V_1 than the initial matrix Q_1. Thus, we expect to compute more accurate approximations to the greatest singular values of A if we reapply the Block Bidiagonalization method with the initial matrix $\hat{Q}_1$. This idea leads to the following iterative scheme, where we assume that b equals k.

ALGORITHM 5.1.

1. Let b, s and δ be given parameters and let Q_1 be a given $n \times b$ orthonormal matrix.
2. Repeat until all b greatest singular values have converged:
   (a) Use the Block Bidiagonalization method with initial matrix Q_1 to generate the matrices J_s, $\bar{P}_s$ and $\bar{Q}_s$.
   (b) Compute the singular value μ_i and corresponding left and right singular vectors $\underset{\sim}{w}_i$ and $\underset{\sim}{z}_i$, respectively, of J_s, for $i = 1,2,\ldots, bs$.
   (c) Estimate the accuracy of μ_i as an approximation to σ_i, for $i = 1,2,\ldots, b$ (cf. inequality (5.2)).
   (d) Let

$$Q_1 := (\underset{\sim}{g}_1, \underset{\sim}{g}_2, \ldots, \underset{\sim}{g}_b),$$

   where

$$\underset{\sim}{g}_i = \bar{Q}_s \underset{\sim}{z}_i \qquad \text{for } i = 1,2,\ldots,b. \qquad \square$$

This iterative algorithm provides a convenient means for estimating the accuracy of the computed singular values. We can show that the i-th column of the matrix Z_2 in the Block Bidiagonalization method computed in each iteration of Algorithm 5.1 after the first is the residual vector for the singular value μ_i computed in the previous iteration. It is thus possible to determine at the start of an iteration the accuracy of the singular values computed at the end of the previous iteration. We also observe that the matrices P_1 and R_2 of the Block Bidiagonalization method are readily available from the prior iteration.

However, once a few singular values and singular vectors have converged, we need not iterate with them any longer. Since the desired singular values may have different rates of convergence, as indicated by the error bounds of Theorems 4.1 and 4.2, we should modify Algorithm 5.1 so that (1) it does not iterate with those singular values and singular vectors that have converged, and (2) it allows the values of b and s to change from one iteration to the next. We are going to examine these issues in the next two sections.

4.6. Block Bidiagonalization Method With Reorthogonalization

Suppose that we are given accurate approximations to the k_0 greatest singular values $\sigma_1, \sigma_2, \ldots, \sigma_{k_0}$ and corresponding singular vectors of the matrix A. Let P_0 and Q_0 be the $m \times k_0$ and $n \times k_0$ orthonormal matrices consisting of those given left and right singular vectors, respectively. We want now to compute the next $(k-k_0)$ greatest singular values of A.

Let us consider the matrix

$$\tilde{A} = \begin{pmatrix} 0 & A \\ A^t & 0 \end{pmatrix} . \tag{1.2}$$

If we apply Algorithm 2.1 to compute the eigenvalues $\pm \sigma_{k_0+1}$, $\pm \sigma_{k_0+2}$, ... , $\pm \sigma_k$ of the matrix $\tilde{A}$, then we must maintain the orthogonality of the matrices $X_1, X_2, \ldots, X_s$ with respect to the eigenvectors corresponding to the eigenvalues $\pm \sigma_1, \pm \sigma_2, \ldots, \pm \sigma_{k_0}$ (cf. [14]). But these eigenvectors are accurately approximated by the columns of the matrix (see [8, Chap. 3])

$$\frac{1}{\sqrt{2}} \begin{pmatrix} P_0 & P_0 \\ Q_0 & -Q_0 \end{pmatrix} .$$

We have that

$$I - \frac{1}{\sqrt{2}} \begin{pmatrix} P_0 & P_0 \\ Q_0 & -Q_0 \end{pmatrix} \frac{1}{\sqrt{2}} \begin{pmatrix} P_0 & P_0 \\ Q_0 & -Q_0 \end{pmatrix}^t = \begin{pmatrix} I - P_0 P_0^t & 0 \\ 0 & I - Q_0 Q_0^t \end{pmatrix} .$$

Therefore, if we want to apply the Block Bidiagonalization method to compute the $(k-k_0)$ next greatest singular values of A, we need to maintain the orthogonality of the matrices $P_1, P_2, \ldots, P_s$ with respect to P_0, and the orthogonality of the matrices $Q_1, Q_2, \ldots, Q_s$ with respect to Q_0. We are thus computing the $(k-k_0)$ greatest

singular values of the "deflated" matrix

$$\hat{A} = (I - P_0 P_0^t)\, A(I - Q_0 Q_0^t) \qquad (6.1)$$

Suppose that

$$\|AQ_0 - P_0\Sigma_0\|_F^2 + \|A^t P_0 - Q_0\Sigma_0\|_F^2 \leq \delta^2 , \qquad (6.2)$$

where $\|\cdot\|_F$ is the Frobenius matrix norm and $\Sigma_0 = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_{k_0})$. Note that if both matrices P_0 and Q_0 had been computed by the Block Lanczos procedure (cf. equation (3.12)), then

$$\|AQ_0 - P_0\Sigma_0\| = mk_0\, O(\epsilon) , \qquad (6.3)$$

where ϵ is the machine precision. Using arguments similar to those in Underwood's thesis [14, pp. 62-66], we get from (6.2) that the $(k-k_0)$ greatest singular values of $\hat{A}$ differ from the $(k-k_0)$ next greatest singular values of A by quantities which are less than $|\delta|$ in modulus.

Now, the Block Bidiagonalization method may be numerically unstable. Although the matrices P_i's and Q_i's form two sequences

of orthonormal matrices in exact arithmetic, they lose orthogonality rapidly in practice due to the loss of figures in the computation of the matrices Z_i's and W_i's. To maintain stability in the scheme, we choose to reorthogonalize the matrices P_i and Q_i with respect to all the previous P_j's and Q_j's, respectively (cf. [14]).

We have incorporated the two reorthogonalization procedures into the next algorithm.

ALGORITHM 6.1. (Block Bidiagonalization method with Reorthogonalization)

1. Let Q_1 be a given $n \times b$ orthonormal matrix.

   Compute

$$W_1 := AQ_1 .$$

   Orthogonalize W_1 with respect to P_0. Factorize W_1 so that

$$W_1 := P_1 R_2 ,$$

   where P_1 is orthonormal and R_2 is upper triangular.

2. For $i = 2, 3, \ldots , s$, do

   (a) Compute

$$Z_i := A^t P_{i-1} - Q_{i-1} R_{2i-2}^t .$$

   Orthogonalize Z_i with respect to $Q_0, Q_1, \ldots , Q_{i-1}$.

   Factorize Z_i so that

$$Z_i := Q_i R_{2i-1} ,$$

   where Q_i is orthonormal and R_{2i-1} is upper triangular.

(b) Compute

$$W_i := AQ_i - P_{i-1}R_{2i-1}^t .$$

Orthogonalize W_i with respect to $P_0, P_1, \ldots, P_{i-1}$.

Factorize W_i so that

$$W_i := P_i R_{2i} ,$$

where P_i is orthonormal and R_{2i} is upper triangular. □

In case of rank deficiency of the matrix Z_i or W_i, we apply a remedial procedure similar to that for Algorithm 3.1.

4.7. Iterative Block Lanczos Method

We should point out that the reorthogonalization process of the last section not only requires a large number of arithmetic operations but also requires that each of the P_j's and Q_j's be in memory during each step of Algorithm 6.1. Since m and n are large numbers in this application, the available computer memory places an upper bound c on the product bs. It is then necessary to determine optimal values for b and s subject to this upper bound constraint. But the error bounds of Theorem 4.2 indicate that we need accurate knowledge of the singular value spectrum of the given matrix, which is precisely the same information we are trying to obtain.

A good initial choice of the block size b is the number k of singular values to be computed (see [2] and [14]). This may not be

the best choice, as we can see from Example 4 of the next section. Our experiments have shown that it seldom pays to have $b > k$ (cf. Example 3 of the next section). Underwood [14] made the same observation, but he used a different way to update the block size.

Having chosen b, we compute the number s of blocks by

$$s := \left\lfloor \frac{c}{b} \right\rfloor ,$$

where $\lfloor \alpha \rfloor$ denotes the integer part of a real number α. If s is less than 2, we compute

$$b := \left\lfloor \frac{c}{2} \right\rfloor$$

and

$$s := \left\lfloor \frac{c}{b} \right\rfloor .$$

The last computation is necessary so that s would equal 3 if the value of c were 3.

Let us describe how we update the values of c, b, s and k. Suppose that k_0 singular values and associated singular vectors have converged in an iteration. Then

$$c := c - k_0$$

because those computed singular vectors must reside in the computer memory for the reorthogonalization process. Now, if $b \geq k$, then we decrease the value of b by k_0, i.e.

$$b := b - k_0 \ ;$$

otherwise we choose the new block size as the smaller value of the old block size and the number of singular values left to be computed, i.e.

$$b := \min(b, \ k - k_0) \ .$$

We update the value of k by

$$k := k - k_0 \ .$$

The new value for s is then computed in the same manner as described in the previous paragraph, with the same modification to the value of b if necessary. Our scheme for updating b differs from Underwood's [14] only in the case where $b > k$.

ALGORITHM 7.1. (Iterative Block Lanczos method)

1. Let c, b, s and δ be given parameters and let Q_1 be a given $n \times b$ orthonormal matrix. The matrices P_0 and Q_0 are null.
2. Repeat until all k singular values have converged:
   (a) Use Algorithm 6.1 with initial matrix Q_1 to generate the matrices J_s, $\bar{P}_s$ and $\bar{Q}_s$.
   (b) Compute the singular value μ_i and corresponding left and right singular vectors $\underset{\sim}{w}_i$ and $\underset{\sim}{z}_i$, respectively, of $\bar{J}_s$, for $i = 1,2,\ldots, bs$.

(c) Estimate the accuracy of μ_i as an approximate singular value, for $i = 1,2,\ldots,$ bs. Assume that k_0 singular values have converged.

(d) Update the values of c, b and s.

(e) For $i = 1,2,\ldots, k_0 + b$, compute

$$\underset{\sim}{p}_i := \bar{P}_s \underset{\sim}{w}_i$$

and

$$\underset{\sim}{q}_i := \bar{Q}_s \underset{\sim}{z}_i \ .$$

(f) Let

$$P_0 := (P_0 | \underset{\sim}{p}_1, \ \ldots \ , \underset{\sim}{p}_{k_0})$$

and

$$Q_0 := (Q_0 | \underset{\sim}{q}_1, \ \ldots \ , \underset{\sim}{q}_{k_0}) \ .$$

(g) Let

$$Q_1 := (\underset{\sim}{q}_{k_0+1}, \ \underset{\sim}{q}_{k_0+2}, \ \ldots \ , \underset{\sim}{q}_{k_0+b}) \ . \quad \square$$

4.8. Test Examples

Rectangular diagonal matrices are chosen for all our examples. Such matrices are sufficiently general for the Lanczos method which does not transform the given matrix. We can thus specify the singular value spectrum and study the behavior of the algorithm as a function of b and s.

A set of FORTRAN routines has been written to implement Algorithm 7.1 (see [5]). We ran our tests on an IBM 370/168 computer

at the Stanford Linear Accelerator Center. The code was compiled by the H EXTENDED compiler with optimization level 2.

The computed singular value μ_i and associated singular vectors p_i and q_i are accepted if they satisfy the inequality

$$(\|Aq_i - \mu_i p_i\|^2 + \|A^t p_i - \mu_i q_i\|^2)^{1/2} \leq 10^{-3} .$$

In Sections 4.5 and 4.6, we have described a way to test for convergence with very little additional work. We have chosen the upper bound c for the product bs to be 12.

The following notations are used in the examples:

$\sigma_1, \sigma_2, \sigma_3, \ldots$		are the computed singular values in the order of convergence.
Iter	=	total number of iterations.
Time	=	machine execution time in seconds.
m' - n	=	$m \times 10^{-n}$.

EXAMPLE 1.

A is a 905×904 matrix with diagonal elements -1.00, -0.99, -0.98, and 0.000, 0.001, ... , 0.900.

k = 3	b = 3
σ_1	1.00 - 4' - 7
σ_2	0.99 - 9' - 9
σ_3	0.98 - 3' - 6
Iter	5
Time	11.36

We note that the computed singular values, as Rayleigh quotients, are accurate to twice the number of digits of the error tolerance.

EXAMPLE 2.

A is a 905×904 matrix with diagonal elements -1.000, -0.999, -0.998, and 0.9000, 0.9001, ..., 0.9900. This example is essentially the same as the previous one except that the gaps between the singular values have been reduced by a factor of 10.

k = 3	b = 3
σ_1 σ_2 σ_3	1.000 - 9' - 7 0.999 - 4' - 8 0.998 - 2' - 8
Iter	6
Time	13.83

The first two examples illustrate the fact that the convergence rate of the Lanczos algorithm depends on the relative spread of the singular values (cf. Theorems 4.1 and 4.2).

EXAMPLE 3.

A is an 806 × 805 matrix with diagonal elements 1.0, -1.0, 0.9, -0.9, and 0.000, 0.001, ... , 0.800.

k = 1	b = 1	b = 2
σ_1	1.0 - 3' - 10	1.0 - 1' - 12
Iter	1	3
Time	2.35	6.52

k = 2	b = 1	b = 2
σ_1	1.0 - 3' - 10	1.0 - 1' - 12
σ_2	0.9 - 3' - 16	1.0 - 1' - 11
Iter	2	3
Time	4.27	6.52

EXAMPLE 3 continued

k = 3	b = 1	b = 2	b = 3	b = 4
σ_1	1.0 - 3' - 10	1.0 - 1' - 12	1.0 - 3' - 8	1.0 - 1' - 7
σ_2	0.9 + 3' - 16	1.0 - 1' - 11	1.0 - 4' - 9	1.0 - 8' - 7
σ_3	1.0 + 0' - 16	0.9 - 1' - 9	0.9 - 3' - 9	0.9 - 6' - 9
Iter	5	4	4	5
Time	9.60	8.28	8.23	10.53

k = 4	b = 1	b = 2	b = 3	b = 4
σ_1	1.0 - 3' - 10	1.0 - 1' - 12	1.0 - 3' - 8	1.0 - 1' - 7
σ_2	0.9 + 3' - 16	1.0 - 1' - 11	1.0 - 6' - 8	1.0 - 8' - 7
σ_3	1.0 + 0' - 16	0.9 - 3' - 8	0.9 - 2' - 7	0.9 - 6' - 9
σ_4	0.9 + 6' - 14	0.9 - 7' - 8	0.9 + 8' - 10	0.9 - 2' - 7
Iter	7	5	5	5
Time	12.70	10.00	9.59	10.38

These tests illustrate a couple of important points. First, the Lanczos method does not always compute the greatest singular values (cf. the case where $k = 2$ and $b = 1$). Second, the supposition of Theorems 4.1 and 4.2 that $\sigma_b > \sigma_{b+1}$ is not necessary for the convergence of the Lanczos method.

EXAMPLE 4.

A is a 902×901 matrix with diagonal elements 0.000, 0.001, ... , 0.900.

k = 3	b = 1	b = 2	b = 3
σ_1	0.900 - 5' - 6	0.900 - 1' - 5	0.900 - 9' - 6
σ_2	0.899 + 3' - 6	0.899 - 2' - 6	0.899 - 9' - 6
σ_3	0.898 - 2' - 5	0.898 - 4' - 7	0.898 - 7' - 5
Iter	13	27	23
Time	28.73	61.85	52.53

This example of a dense singular value spectrum is one in which the point algorithm ($b = 1$) works better than the block algorithm ($b \geq 2$).

Our Iterative Block Lanczos method is therefore a good procedure for computing a few greatest singular values of a matrix. A block method with an appropriate block size can (1) have a fast convergence rate, and (2) handle well the case of multiple singular values (see Example 3). For problems where the given matrix has to be read from secondary storage, economics may dictate that we multiply the matrix into a block of vectors and thus choose a block method.

REFERENCES

[1] Cullum, J., "The simultaneous computation of a few algebraically largest and smallest eigenvalues of a large, sparse, symmetric matrix," Report RC 6827, IBM Thomas J. Watson Research Center, Yorktown Heights (1977).

[2] Cullum, J., and Donath, W. E., "A block Lanczos algorithm for computing the q algebraically largest eigenvalues and a corresponding eigenspace of large, sparse, real symmetric matrices," Proc. 1974 IEEE Conf. on Decision and Control, Phoenix, Arizona (1974), 505-509.

[3] Garbow, B.S., Boyle, J.M., Dongarra, J.J., and Moler, C.B., Matrix Eigensystem Routines--EISPACK Guide Extension, Springer-Verlag, Berlin (1977).

[4] Golub, G.H., and Luk, F.T., "Singular value decomposition: applications and computations," ARO Report 77-1, Transactions of the 22nd Conference of Army Mathematicians (1977), 577-605.

[5] Golub, G.H., Luk, F.T., and Overton, M.L., "A block Lanczos method to compute the singular values and corresponding singular vectors of a matrix," Report STAN-CS-77-635, Computer Science Dept., Stanford University (1977).

[6] Golub, G.H., and Reinsch, C., "Singular value decomposition and least squares solutions," Numer. Math. 14 (1970), 403-420.

[7] Golub, G.H., and Underwood, R., "The Block Lanczos method for computing eigenvalues," in Mathematical Software III, (J. R. Rice, Ed.), Academic Press, New York (1977), 361-377.

[8] Lanczos, C., Linear Differential Operators, Van Nostrand, London (1961).

[9] Lewis, J.G., "Algorithms for sparse matrix eigenvalue problems," Ph.D. thesis. Report STAN-CS-77-595, Computer Science Dept., Stanford University (1977).

[10] Ortega, J.M., Numerical Analysis: A Second Course, Academic Press, New York (1972).

[11] Ruhe, A., "Implementation aspects of Band Lanczos algorithms for computation of eigenvalues of large sparse symmetric matrices," Report, Dept. of Mathematics, University of California, San Deigo (February 1978).

[12] Smith, B.T., Boyle, J.M., Dongarra, J.J., Garbow, B.S., Ikebe, Y,., Klema, V.C., and Moler, C.B., Matrix Eigensystem Routines--EISPACK Guide, Second Edition, Springer-Verlag, Berlin (1976).

[13] Stewart, G.W., "Error and perturbation bounds for subspaces associated with certain eigenvalue problems," SIAM Review 15 (1973), 727-764.

[14] Underwood, R., "An iterative Block Lanczos method for the solution of large sparse symmetric eigenproblems," Ph.D. thesis, Report STAN-CS-75-496, Computer Science Dept., Stanford University (1975).

[15] Wilkinson, J.H., and Reinsch, C., Handbook for Automatic Computation, Volume II, Linear Algebra, Part 2, Springer-Verlag, Berlin (1971).

V. COMPUTING THE SINGULAR VALUE DECOMPOSITION ON THE ILLIAC IV

5.1. Introduction

We study the computation of the singular value decomposition on the ILLIAC IV computer. Suppose that we have a real $m \times n$ matrix A. Its singular value decomposition (SVD) can be defined as

$$A = U\Sigma V^t \tag{1.1}$$

with

$$U^tU = V^tV = I_k \qquad \text{and} \qquad \Sigma = \text{diag}(\sigma_1, \ldots, \sigma_k) ,$$

where

$$k = \min(m,n) . \tag{1.2}$$

The matrices U and V consist of the orthonormalized eigenvectors associated with the k largest eigenvalues of AA^t and A^tA, respectively. The diagonal elements of Σ are the non-negative square roots of the k largest eigenvalues of AA^t, and are called the singular values. We assume that

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0 \qquad \text{and} \qquad \sigma_{r+1} = \cdots = \sigma_k = 0, \tag{1.3}$$

i.e. the rank of A equals r. An alternative definition of the singular value decomposition is

$$A = U_r\Sigma_rV_r^t \tag{1.4}$$

with

$$U_r^tU_r = V_r^tV_r = I_r \qquad \text{and} \quad \Sigma_r = \text{diag}(\sigma_1, \ldots, \sigma_r).$$

The singular value decomposition is a very useful matrix decomposition (see [7]). Various methods have been proposed for its computation. The standard method was introduced by Golub and Kahan in 1965 [6]. They use first the Householder transformation to bidiagonalize the given matrix, and then the QR method to compute the singular values of the resultant bidiagonal form. Their method superceded a one-sided orthogonalization method given by Hestenes in 1958[10]. Hestenes' method is, however, easily adaptable to special purpose computations. It was suggested by Chartres (1962) [2] for a computer with a magnetic backing store, and was implemented on a mini-computer by Nash [12]. In this chapter, we study the implementation of Hestenes' method on the ILLIAC IV computer and show that the method makes very efficient use of the parallel computing abilities of the ILLIAC machine.

We are going to use the Frobenius norm for matrices, i.e.

$$\|A\| = \|A\|_F = \Big(\sum_{i,j} a_{ij}^2\Big)^{1/2} \quad \text{for} \quad A \equiv (a_{ij}) ,$$

and the Euclidean norm for vectors, i.e.

$$\|\underset{\sim}{x}\| = \|\underset{\sim}{x}\|_2 = (\underset{\sim}{x}^t \underset{\sim}{x})^{1/2} .$$

5.2. The ILLIAC IV Computer

The ILLIAC IV computer was built by the Burroughs Corporation and is located at NASA/Ames Research Center, Moffett Field, California. The computer consists of 64 synchronous processing elements (PE's)

under the direction of a single control unit (CU). Each PE has 2048 words of 64-bit memory with an access time of 188 nanoseconds, and is capable of performing a general floating-point operation in about 1.7 microseconds and a typical bookkeeping operation in about 1.2 microseconds. The PE instruction set is similar to that of conventional machines with two exceptions. First, each PE can communicate data to four other PE's through routing instructions. Second, the PE's can set their own mode registers to effectively disable or enable themselves. The CU takes about 0.7 microseconds to perform a bookkeeping operation.

The main memory of the ILLIAC is logically a 16-million word drum, which is divided into 52 bands and has a 40 millisecond rotation period. Data transfers to or from the PE memory are program initiated and are performed in blocks of 1024 words. The transfer time for 1024 words is about 66 microseconds; it takes about 4.2 milliseconds to refresh half the PE memory.

A floating-point number on the ILLIAC consists of a 1-bit sign, a 15-bit exponent to the radix 2, and a normalized 48-bit mantissa. The machine precision ϵ is thus about 3.55×10^{-15}. A fixed-point number has a 1-bit sign and a 48-bit mantissa.

5.3. Programming Languages for the ILLIAC

There are three languages available for programming the ILLIAC; its assembly language, ASK; a FORTRAN-like language, CFD [15]; and an ALGOL 60-like language, GLYPNIR [11]. Both CFD and

GLYPNIR do not hide the basic 64-wide architecture of the ILLIAC. We must restructure our data and algorithm so that the computation can be done in parallel in "strips" of width 64 or less.

Let us briefly describe the data declarations in GLYPNIR. The PE memory of the ILLIAC can be viewed as a two-dimensional structure where each word can be addressed by an ordered pair which specifies the PE memory module and the address within that module. A group of 64 words, each in a different module but each having the same address within its module, is called a superword or sword. We can divide the variable types in GLYPNIR into two major categories. The first represents words or vectors of words; they are called the CU variables. The second represents swords or vectors of swords; they are called the PE variables. There are also the Boolean variables and the so-called ADB variables.

A sword vector of length n represents an indexable vector of swords. It is thus in some sense an $n \times 64$ array. A GLYPNIR program cannot directly handle two-dimensional arrays whose row and column dimensions exceed 64.

5.4. A Row Orthogonalization Method

There are two reasons why the standard SVD method of Golub and others (see [6] and [8]) may be undesirable on a parallel processor. First, although the Householder transformation is inherently parallel, the effective vector length decreases at each step, causing inefficiencies.

Second, the parallel QR method [14] may be numerically unstable (see [9]). In contrast, the one-sided orthogonalization method of Hestenes [10] is easily adaptable to computation on a parallel machine.

The method of Hestenes consists of generating an orthogonal matrix V such that the non-null column vectors of the matrix

$$H = AV$$

are mutually orthogonal and non-increasing in norm. The nonzero columns of H are then normalized so that

$$H = (U_r|0) \left(\begin{array}{c|c} \Sigma_r & 0 \\ \hline 0 & 0 \end{array} \right)$$

with

$$U_r^t U_r = I_r \qquad \text{and} \qquad \Sigma_r = \text{diag}(\sigma_1, \ \ldots \ , \sigma_r).$$

Consequently,

$$A = U_r \Sigma_r V_r^t \ , \tag{1.4}$$

where

V_r is an $n \times r$ matrix consisting of the first r columns of V.

Nash [12] followed Hestenes' approach, but Chartres [2] chose to orthogonalize the rows of the given matrix A. We have decided on the row orthogonalization scheme, for it is easily adaptable to solving overdetermined linear equations.

We aim to generate an orthogonal matrix U^t so that the non-null row vectors of the matrix

$$K = U^t A$$

are mutually orthogonal and non-increasing in norm. We then normalize the nonzero rows of K to obtain

$$K = \left(\begin{array}{c|c} \Sigma_r & 0 \\ \hline 0 & 0 \end{array}\right)\left(\begin{array}{c} V_r^t \\ \hline 0 \end{array}\right)$$

with

$$\Sigma_r = \mathrm{diag}(\sigma_1, \ldots, \sigma_r) \quad \text{and} \quad V_r^t V_r = I_r .$$

It follows that

$$A = U_r \Sigma_r V_r^t , \tag{1.4}$$

where

U_r is an $m \times r$ matrix consisting of the first r columns of U.

We are going to construct the matrix U as a product of plane rotations. Let us write the matrix A as

$$A = \begin{pmatrix} \underset{\sim}{a}_1^t \\ \underset{\sim}{a}_2^t \\ \vdots \\ \underset{\sim}{a}_m^t \end{pmatrix} , \tag{4.1}$$

where

$\underset{\sim}{a}_i^t$ is an $1 \times n$ row vector, for $i = 1,2,\ldots, m$.

Given any two rows $\underset{\sim}{a}_i^t$ and $\underset{\sim}{a}_j^t$, with $i < j$. We would consider them orthogonal if

$$\|\underset{\sim}{a}_i\| < \epsilon \qquad \text{or} \qquad \|\underset{\sim}{a}_j\| < \epsilon, \tag{4.2}$$

where ϵ is the machine precision, or if

$$\frac{\underset{\sim}{a}_i^t \underset{\sim}{a}_j}{\|\underset{\sim}{a}_i\| \, \|\underset{\sim}{a}_j\|} < \tau \, , \tag{4.3}$$

where τ is a previously chosen tolerance. We do not transform orthogonal rows, but would permute them if

$$\|\underset{\sim}{a}_i\| < \|\underset{\sim}{a}_j\| \, .$$

Suppose now that the two given rows do not satisfy the orthogonality condition (4.2) or (4.3). Let us consider the action of a plane rotation:

$$\begin{pmatrix} \cos\varphi & \sin\varphi \\ -\sin\varphi & \cos\varphi \end{pmatrix} \begin{pmatrix} \underset{\sim}{a}_i^t \\ \underset{\sim}{a}_j^t \end{pmatrix} = \begin{pmatrix} \hat{\underset{\sim}{a}}_i^t \\ \hat{\underset{\sim}{a}}_j^t \end{pmatrix} . \tag{4.4}$$

The idea is to choose φ such that

$$\hat{\underset{\sim}{a}}_i^t \hat{\underset{\sim}{a}}_j = 0 \qquad \text{and} \qquad \|\hat{\underset{\sim}{a}}_i\| \geq \|\hat{\underset{\sim}{a}}_j\| \, .$$

The second condition ensures that the computation always proceeds towards an ordering of row norms. Following Nash [12], we let

$$\alpha = 2\underset{\sim}{a}_i^t \underset{\sim}{a}_j$$

$$\beta = \|\underset{\sim}{a}_i\|^2 - \|\underset{\sim}{a}_j\|^2 , \qquad (4.5)$$

and

$$\gamma = (\alpha^2 + \beta^2)^{1/2} .$$

Note that γ is positive since α is nonzero. Then, if β is positive, we compute

$$\cos \varphi = \left(\frac{\gamma + \beta}{2\gamma} \right)^{1/2} \quad \text{and} \quad \sin \varphi = \frac{\alpha}{2\gamma \cos \varphi} , \qquad (4.6)$$

otherwise we let

$$\sin \varphi = \left(\frac{\gamma - \beta}{2\gamma} \right)^{1/2} \quad \text{and} \quad \cos \varphi = \frac{\alpha}{2\gamma \sin \varphi} . \qquad (4.7)$$

In (4.4), we could use the Fast Givens transformations (see [4]) which requires only $2n$ multiplications, an apparent 50% work reduction. But a heavy overhead in maintaining the scaling factors eats up the savings unless the row length n is moderately large [16].

As in the traditional Jacobi algorithm, the plane rotations are performed in a set sequence called a sweep, which consists of the $[m(m-1)]/2$ plane rotations on the row pairs

$$(1,2), (1,3), \ldots , (1,m), (2,3), \ldots , (2,m), (3,4), \ldots , (m-1,m).$$

The iterative procedure terminates if (1) all the rows are pairwise orthogonal, and (2) no row permutations have occurred, in one complete sweep.

Our orthogonalization method is in essence the Jacobi method implicitly applied to the matrix AA^t to compute its eigenvalues. We can refer to the literature [17] for the convergence properties of our method. We see that the convergence is quadratic and takes the order of 6 to 10 sweeps, i.e. from $3m^2$ to $5m^2$ plane rotations (see [13]).

We now present our method in its entirety. Two Boolean variables are introduced:

withu : true if matrix U is desired, false otherwise.

withv : true if matrix V is desired, false otherwise.

We make the arbitrary choice that

$$\tau = 10^{-12} ,$$

and write the matrices U and V as

$$U = (\underset{\sim}{u}_1, \underset{\sim}{u}_2, \ldots , \underset{\sim}{u}_m) , \tag{4.8}$$

and

$$V = (\underset{\sim}{v}_1, \underset{\sim}{v}_2, \ldots , \underset{\sim}{v}_n) .$$

ALGORITHM 1 (SVD).

I. Initialize:

(1) Let

$$\epsilon := 3.55 \times 10^{-15},$$

$$\tau := 10^{-12} ,$$

and

$$c := 0 .$$

(2) If (withu) then let U := I.

II. Repeat until $c = [m(m-1)]/2$:

(1) Let $c := 0$

(2) For $(i,j) := (1,2),(1,3),\ldots,(1,m),(2,3),\ldots,(2,m),$
$(3,4),\ldots,(3,m),\ldots,(m-1,m)$ do

If $\|\underset{\sim}{a}_j\| < \epsilon$ then

(a) Let $c := c + 1$.

Else if $\|\underset{\sim}{a}_i\| < \epsilon$ then

(a) Exchange $\underset{\sim}{a}_i$ and $\underset{\sim}{a}_j$.

(b) If (withu) then exchange $\underset{\sim}{u}_i$ and $\underset{\sim}{u}_j$.

Else if

$$\frac{\underset{\sim}{a}_i^t \underset{\sim}{a}_j}{\|\underset{\sim}{a}_i\| \|\underset{\sim}{a}_j\|} < \tau$$

then

(a) Let $c := c + 1$.

(b) If $\|\underset{\sim}{a}_i\| < \|\underset{\sim}{a}_j\|$ then

(i) Exchange $\underset{\sim}{a}_i$ and $\underset{\sim}{a}_j$.

(ii) If (withu) then exchange $\underset{\sim}{u}_i$ and $\underset{\sim}{u}_j$.

Else

(a) Compute

$$\alpha := 2\underset{\sim}{a}_i^t \underset{\sim}{a}_j$$

$$\beta := \|\underset{\sim}{a}_i\|^2 - \|\underset{\sim}{a}_j\|^2 ,$$

and

$$\gamma := (\alpha^2 + \beta^2)^{1/2}.$$

(b) If $\beta > 0$ then

(i) Compute

$$\cos \varphi := \left(\frac{\gamma + \beta}{2\gamma} \right)^{1/2} ,$$

and

$$\sin \varphi := \frac{\alpha}{2\gamma \cos \varphi} .$$

Else

(i) Compute

$$\sin \varphi := \left(\frac{\gamma - \beta}{2\gamma} \right)^{1/2} ,$$

and

$$\cos \varphi := \frac{\alpha}{2\gamma \sin \varphi} .$$

(c) Compute

$$\underset{\sim}{w} := \cos \varphi \cdot \underset{\sim}{a}_i + \sin \varphi \cdot \underset{\sim}{a}_j ,$$

$$\underset{\sim}{a}_j := -\sin \varphi \cdot \underset{\sim}{a}_i + \cos \varphi \cdot \underset{\sim}{a}_j ,$$

and

$$\underset{\sim}{a}_i := \underset{\sim}{w} .$$

(d) If (withu) then

$$\underset{\sim}{z} := \cos \varphi \cdot \underset{\sim}{u}_i + \sin \varphi \cdot \underset{\sim}{u}_j ,$$

$$\underset{\sim}{u}_j := -\sin \varphi \cdot \underset{\sim}{u}_i + \cos \varphi \cdot \underset{\sim}{u}_j ,$$

and

$$\underset{\sim}{u}_i := \underset{\sim}{z} .$$

III. Compute singular values:

(1) Let $i := 1$.

(2) Repeat until $i > m$ or $\|\underset{\sim}{a}_i\| < \epsilon$:

(a) Let $\sigma_i := \|\underset{\sim}{a}_i\|$.

(b) Compute $\underset{\sim}{v}_i := \dfrac{\underset{\sim}{a}_i}{\|\underset{\sim}{a}_i\|}$.

(c) Let $i := i + 1$.

(3) Let $r := i - 1$.

We wish to compare the required work of Algorithm 1 and the Golub-Reinsch method [8]. One sweep of Algorithm 1 takes about $(7n + 4m)m^2/2$ multiplications if the U matrix is desired, and $7nm^2/2$ multiplications otherwise. We assume that

$$m \leq n ,$$

for we can compute the SVD of A^t if $m > n$. We further suppose that our Jacobi-like method takes 8 sweeps to converge and that only two QR steps are required per singular value for the Golub-Reinsch algorithm (cf. [1]). The following table gives the number of multiplications required by the two methods in four different cases.

Matrices Desired	Algorithm 1	Golub-Reinsch
U_r, Σ_r, V_r	$28m^2n + 16m^3$	$7m^2n + 11m^3/3$
U_r, Σ_r	$28m^2n + 16m^3$	$7m^2n - m^3$
Σ_r, V_r	$28m^2n$	$2m^2n + 4m^3$
Σ_r	$28m^2n$	$2m^2n - 2m^3/3$

We see that Algorithm 1 is about four times slower than the standard SVD algorithm in computing the full singular value decomposition. However, the special architecture of the ILLIAC IV computer can reduce the number of required multiplications by an asymptotic factor of 64. Our Jacobi-like algorithm is therefore very efficient on a parallel computer. We should mention that Chan [1] described a modified Golub-Reinsch algorithm that could save up to 50% of machine execution time if $m \ll n$.

5.5. Least Squares Solutions

We refer to an $n \times m$ matrix S as the pseudoinverse of an $m \times n$ matrix A if S satisfies the following four conditions:

$$\begin{aligned} &\text{(i)} \quad ASA = A \,, \\ &\text{(ii)} \quad SAS = S \,, \\ &\text{(iii)} \quad (AS)^t = AS, \\ \text{and} \quad &\text{(iv)} \quad (SA)^t = SA \,. \end{aligned} \tag{5.1}$$

Let us denote the pseudoinverse by A^+. Now, we have defined the singular value decomposition of A as

$$A = U\Sigma V^t \tag{1.1}$$

We can therefore write the pseudoinverse as

$$A^+ = V\Omega U^t \,, \tag{5.2}$$

where Ω is an $n \times m$ matrix to be determined. Using the four conditions of (5.1), we easily determine that Ω is given uniquely by

$$\Omega = \left(\begin{array}{c|c} \begin{matrix} \sigma_1^{-1} & & \\ & \ddots & \\ & & \sigma_r^{-1} \end{matrix} & \begin{matrix} 0 \\ \vdots \\ 0 \end{matrix} \\ \hline \begin{matrix} 0 & \cdots & 0 \end{matrix} & 0 \end{array}\right) \,. \tag{5.3}$$

An important application of the pseudoinverse is in solving linear equations

$$AX = B \ , \tag{5.4}$$

where B is a given $m \times s$ matrix. We assume that

$$m \geq n \ , \tag{5.5}$$

and we seek an $n \times s$ matrix X such that

$$\|B - AX\| = \min \ . \tag{5.6}$$

The solution matrix X is not unique unless the matrix A is of full rank. We therefore impose the condition that we want the matrix $\hat{X}$ of minimum norm in the solution space. It is well known (see, e.g., [6]) that $\hat{X}$ is unique and is given by

$$\hat{X} = A^{+}B \ . \tag{5.7}$$

Thus, we have that

$$\hat{X} = V\Omega C \ , \tag{5.8}$$

where

$$C = U^{t}B \ .$$

The matrix C can be generated by applying to the rows of B those plane rotations that we use to orthogonalize the rows of A. It is unnecessary to accumulate the plane rotations.

We now present an algorithm based on Algorithm SVD for computing the minimum norm solution to the overdetermined system (5.4). There is an input parameter "cutoff." Our method sets to zero all those singular values of A that are smaller than "cutoff." The $1 \times s$ row vector $\underset{\sim}{b}_i^t$ denotes the i-th row of B, for $i = 1,2,\ldots,m$.

ALGORITHM 2. (MINFIT)

I. Initialize:

(1) Let

$$\epsilon := 3.55 \times 10^{-15} ,$$

$$\tau := 10^{-12}$$

and

$$c := 0 .$$

II. Repeat until $c = [m(m-1)]/2$:

(1) Let $c := 0$.

(2) For $(i,j) := (1,2),(1,3),\ldots,(1,m),(2,3),\ldots,(2,m),$
$(3,4),\ldots,(3,m),\ldots,(m-1,m)$ do

If $\|\underset{\sim}{a}_j\| < \epsilon$ then

(a) Let $c := c + 1$.

Else if $\|\underset{\sim}{a}_i\| < \epsilon$ then

(a) Exchange $\underset{\sim}{a}_i$ and $\underset{\sim}{a}_j$.

(b) Exchange $\underset{\sim}{b}_i$ and $\underset{\sim}{b}_j$.

Else if

$$\frac{\underset{\sim}{a}_i^t \underset{\sim}{a}_j}{\|\underset{\sim}{a}_i\|\|\underset{\sim}{a}_j\|} < \tau$$

then

(a) Let $c := c + 1$.

(b) If $\|\underset{\sim}{a}_i\| < \|\underset{\sim}{a}_j\|$ then

(i) Exchange $\underset{\sim}{a}_i$ and $\underset{\sim}{a}_j$.

(ii) Exchange $\underset{\sim}{b}_i$ and $\underset{\sim}{b}_j$.

Else

(a) Compute

$$\alpha := 2\underset{\sim}{a}_i^t\underset{\sim}{a}_j$$

$$\beta := \|\underset{\sim}{a}_i\|^2 - \|\underset{\sim}{a}_j\|^2 ,$$

and

$$\gamma := (\alpha^2 + \beta^2)^{1/2} .$$

(b) If $\beta > 0$ then

(i) Compute

$$\cos \varphi := \left(\frac{\gamma + \beta}{2\gamma} \right)^{1/2}$$

and

$$\sin \varphi := \frac{\alpha}{2\gamma \cos \varphi} .$$

Else

(i) Compute

$$\sin \varphi := \left(\frac{\gamma - \beta}{2\gamma} \right)^{1/2}$$

and

$$\cos \varphi := \frac{\alpha}{2\gamma \sin \varphi} .$$

(c) Compute

$$\underset{\sim}{w} := \cos \varphi \cdot \underset{\sim}{a}_i + \sin \varphi \cdot \underset{\sim}{a}_j ,$$

$$\underset{\sim}{a}_j := -\sin \varphi \cdot \underset{\sim}{a}_i + \cos \varphi \cdot \underset{\sim}{a}_j ,$$

and $\underset{\sim}{a}_i := \underset{\sim}{w}$.

(d) Compute

$$\underset{\sim}{z} := \cos \varphi \cdot \underset{\sim}{b}_i + \sin \varphi \cdot \underset{\sim}{b}_j ,$$

$$\underset{\sim}{b}_j := -\sin \varphi \cdot \underset{\sim}{b}_i + \cos \varphi \cdot \underset{\sim}{b}_j ,$$

and

$$\underset{\sim}{b}_i := \underset{\sim}{z} .$$

III. Compute least squares solution:

(1) Let

$$V := 0 ,$$

$$Y := 0 ,$$

and

$$i := 1 .$$

(2) Repeat until $i > n$ or $\|\underset{\sim}{a}_i\| < \text{cutoff}$:

(a) Compute

$$\underset{\sim}{v}_i := \frac{\underset{\sim}{a}_i}{\|\underset{\sim}{a}_i\|}$$

and

$$\underset{\sim}{b}_i := \frac{\underset{\sim}{b}_i}{\|\underset{\sim}{a}_i\|} .$$

(b) Let $i := i + 1$.

(3) Let $r := i - 1$.

(4) Compute $X := VB$.

One sweep of Algorithm 2 takes about $7m^2n/2$ multiplications. We neglect the terms involving s because $s \ll n$ in most applications. With the same assumptions on convergence rates as in the last section, the required work for our MINFIT algorithm is about $28m^2n$ multiplications while that for a similar method based on the Golub-Reinsch algorithm (see [8]) is about $2mn^2 + 4n^3$ multiplications. If $m \gg n$, it saves work to first reduce the regression matrix A to upper triangular form using Householder transformations, before applying the MINFIT algorithm (cf. [1]). Such a two-stage scheme requires about $mn^2 + 83n^2/3$ multiplications. Fortunately, the parallel computing abilities of the ILLIAC machine reduce the work of our algorithm by an asymptotic factor of 64. Thus, our MINFIT algorithm is an effective solver for least squares problems.

5.6. Data Structures.

Let us first assume that $n \leq 64$. As our algorithms access A by rows, we lay out the rows of the matrix across the processing elements of the ILLIAC. We thus represent the matrix by a sword vector A[*] of order m.

We work with the rows of A to compute (a) the pairwise inner product, and (b) the new rows after a plane rotation. The GLYPNIR language provides a built-in function ROWSUM that sums the 64 numbers of a sword in 6 additions. The GLYPNIR expression

```
ROWSUM(A[I] * A[J])
```

computes the inner product of the i-th and j-th rows of A. If $n < 64$, we must disable the last $(64 - n)$ processing elements when we call the ROWSUM function. An alternative is to apply our algorithms to an $m \times 64$ matrix $\hat{A}$, given by

$$\hat{A} = (A|0) .$$

The following lines of GLYPNIR code compute the new i-th and j-th rows of A at the end of a plane rotation:

```
T    := A[I] * COSPHI + A[J] * SINPHI ;
A[J] := -A[I] * SINPHI + A[J] * COSPHI ;
A[I] := T ;
```

where T is a sword used for temporary storage.

We now consider the case when $n > 64$. Let

$$\ell = \left\lceil \frac{n}{64} \right\rceil ,$$

i.e. ℓ equals the smallest integer $\geq n/64$. We construct an $m \times 64\ell$ matrix $\hat{A}$, given by

$$\hat{A} = (A|0) \ .$$

The rows of $\hat{A}$ are then divided into ℓ equal segments:

$$\hat{A} = (A_1|A_2|\cdots|A_\ell) \ .$$

Thus, we represent the matrix $\hat{A}$ by the ℓ sword vectors A1[*], ... , Aℓ[*], each of order m.

The GLYPNIR expression

```
ROWSUM(A1[I] * A1[J] + ··· + Aℓ[I] * Aℓ[J])
```

computes the inner product of the i-th and j-th rows of the matrix A. Plane rotations are applied to individual segments of the rows. For example, we write the ℓ lines of code

```
A1[J] := -A1[I] * SINPHI + A1[J] * COSPHI ;
A2[J] := -A2[I] * SINPHI + A2[J] * COSPHI ;
   ⋮
Aℓ[J] := -Aℓ[I] * SINPHI + Aℓ[J] * COSPHI ;
```

to compute the new vector

$$\underset{\sim}{a}_j := -\underset{\sim}{a}_i \cdot \sin \varphi + \underset{\sim}{a}_j \cdot \cos \varphi \ .$$

Since the columns of the matrix U are transformed in the same manner as the rows of A, we lay out U so that its columns lie across the processing elements. Thus, we represent the matrix by a sword vector U[*] of dimension m. For the two different cases of $m \leq 64$ and $m > 64$, we apply techniques similar to those discussed in the previous paragraphs.

The rows of the data matrix B are modified in an identical fashion as the rows of A. Therefore, we lay out the rows of B across the processing elements. The two cases of column dimension $s \leq 64$ and $s > 64$ for B are dealt with in the same manner as the corresponding cases for the matrix A. We note that the execution time of Algorithm MINFIT is independent of s, for $s \leq 64$. If s is much greater than m, then the execution time will be proportional to $\left\lceil \frac{s}{64} \right\rceil$.

5.7. Numerical Properties

Let us examine the question of numerical stability. An error analysis of the action of plane rotations on a matrix was given by Wilkinson in his classic text [18]. His error bounds were later improved by Gentleman [5]. We use their results to study the effects of the plane rotations in one sweep of our algorithm.

Let

$$M = \frac{1}{2} m(m-1) , \tag{7.1}$$

and let R_j represent the j-th plane rotation, for $j = 1,2,\ldots,M$. We can show that the computed matrix $\bar{A}_M$ after one sweep of rotations satisfies the inequality

$$\|\bar{A}_m - R_M R_{M-1} \cdots R_1 A\| \leq 2^{-48}(m+n-2)(1+2^{-48})^{m+n-2}\|A\| . \tag{7.2}$$

The right-hand side of the inequality (7.2) is an extreme upper bound. We expect the statistical distribution of the rounding errors to reduce the error to well below the level of the bound; for this reason alone, a factor of the order of $(m+n-2)^{1/2}$ in place of $(m+n-2)$ might be more realistic. We see that our algorithm is extremely stable.

As the matrix U is formed as a product of plane rotations, we examine here the deviation from orthogonality of such a product. Let $\bar{Q}_M$ represent the computed product of the plane rotations in one sweep. We have the inequality that

$$\|\bar{Q}_M - R_M R_{M-1} \cdots R_1\| \leq 2^{-48}\, m^{1/2}(m+n-2)(1+2^{-48})^{m+n-2} . \quad (7.3)$$

Again statistical consideration indicates that a factor of the order of $m^{1/4}(m+n-2)^{1/2}$ instead of $m^{1/2}(m+n-2)$ is probably more realistic. The matrix U is thus very close to an orthogonal matrix.

The tolerance τ controls the accuracy of the solution. At convergence of our algorithm, we have that

$$\|V_r^t V_r - I_r\| \leq r^{1/2}\tau . \quad (7.4)$$

Indeed, our numerical experiments show that the accuracy of the computed singular values and vectors of A is of the order of τ.

5.8. Test Results

We have written two GLYPNIR programs implementing Algorithms SVD and MINFIT. Tests were carried out on the ILLIAC IV computer.

EXAMPLE 1 (see [8]).

We have chosen the following matrices:

$$A = \begin{bmatrix} 22 & 10 & 2 & 3 & 7 \\ 14 & 7 & 10 & 0 & 8 \\ -1 & 13 & -1 & -11 & 3 \\ -3 & -2 & 13 & -2 & 4 \\ 9 & 8 & 1 & -2 & 4 \\ 9 & 1 & -7 & 5 & -1 \\ 2 & -6 & 6 & 5 & 1 \\ 4 & 5 & 0 & -2 & 2 \end{bmatrix}, \quad B = \begin{bmatrix} -1 & 1 & 0 \\ 2 & -1 & 1 \\ 1 & 10 & 11 \\ 4 & 0 & 4 \\ 0 & -6 & -6 \\ -3 & 6 & 3 \\ 1 & 11 & 12 \\ 0 & -5 & -5 \end{bmatrix}.$$

The singular values of A are $\sqrt{1248}$, 20, $\sqrt{384}$, 0 and 0. Our SVD program computed those values to machine precision. The minimum norm solution to the overdetermined system

$$AX = B$$

is given by

$$X = \begin{pmatrix} -\frac{1}{12} & 0 & -\frac{1}{12} \\ 0 & 0 & 0 \\ \frac{1}{4} & 0 & \frac{1}{4} \\ -\frac{1}{12} & 0 & -\frac{1}{12} \\ \frac{1}{12} & 0 & \frac{1}{12} \end{pmatrix}.$$

Our MINFIT program returned a solution accurate to 14 decimal digits.

EXAMPLE 2 (see [6]).

We have chosen the matrix

$$A = \begin{pmatrix} 1 & -1 & -1 & -1 & -1 & \cdots \\ & 1 & -1 & -1 & -1 & \cdots \\ & & 1 & -1 & -1 & \cdots \\ & & & 1 & -1 & \cdots \\ & \bigcirc & & & 1 & \cdots \\ & & & & & \cdots \end{pmatrix}_{n \times n},$$

which is ill-conditioned as it has a very small singular value.

The matrix becomes singular if we add -2^{-n+2} to its $(n,1)$ position.

We applied our SVD program to this choice of A for different values of n. For comparison, we have chosen the SVD subroutine in the EISPACK eigenvalue package from the Argonne National Laboratory [3]. The EISPACK routine implements the method of Golub and Reinsch [8] and has been coded for high execution efficiency. We applied the routine to the matrix A on an IBM 370/168 computer at the Stanford Linear Accelerator Center. The code was compiled by the FORTRAN H EXTENDED compiler with optimization level 2.

n	ILLIAC IV		IBM 370/168	ILLIAC TIME / IBM TIME
	iter	time	time	
16	7	0.26	0.101	2.57
32	8	1.25	0.57	2.19
48	8	2.89	1.76	1.64
64	9	5.57	4.03	1.38
96	10	15.94	12.81	1.24
128	9	26.81	29.68	0.90

We should point out that the GLYPNIR compiler produces very inefficient code. We have written another program in CFD implementing our SVD algorithm. The ILLIAC execution time with $n = 64$ was 3.31 seconds, a saving of 41% over the GLYPNIR code.

It is unfortunate that due to certain limitations we were not able to run examples with larger values of n on the ILLIAC. Nonetheless, we observe that our ILLIAC routine becomes more efficient relative to the EISPACK routine with increasing values of n. The execution time of the former is crudely proportional to

$$\left(\text{iter} \times \left\lceil \frac{n}{64} \right\rceil \times n^2 \right),$$

and that of the latter to n^3. There is thus a great potential in matrix computations of a parallel computer with many processors.

REFERENCES

[1] Chan, T. F. C., "On computing the singular value decomposition," Report STAN-CS-77-588, Computer Science Dept., Stanford University (1977).

[2] Chartres, B.A., "Adaptation of the Jacobi method for a computer with a magnetic-tape backing store," Computer J. 5 (1962), 51-60.

[3] Garbow, B.S., Boyle, J.M., Dongarra, J.J., and Moler, C.B., Matrix Eigensystem Routines--EISPACK Guide Extension Springer-Verlag, Berlin (1977).

[4] Gentleman, W.M., "Least squares computations by Givens' transformations without square roots," J. Inst. Maths. Applics. 12 (1973), 329-336.

[5] Gentleman, W.M., "Error analysis of QR decomposition by Givens transformations," Lin. Alg. Applics. 10 (1975), 189-197.

[6] Golub, G.H., and Kahan, W., "Calculating the singular values and pseudoinverse of a matrix," J. SIAM Ser. B: Numer. Anal. 2 (1965), 205-224.

[7] Golub, G.H., and Luk, F.T., "Singular value decomposition: applications and computations," ARO Report 77-1, Transactions of the 22-nd Conference of Army Mathematicians (1977), 577-605.

[8] Golub, G.H., and Reinsch, C., "Singular value decomposition and least squares solutions," Numer. Math. 14 (1970), 403-420.

[9] Heller, D., "A survey of parallel algorithms in numerical linear algebra," Report, Computer Science Dept., Carnegie-Mellon University (February 1976).

[10] Hestenes, M.R., "Inversion of matrices by biorthogonalization and related results," J. Soc. Indust. Appl. Math. 6 (1958), 51-90.

[11] Lawrie, D.H., Layman, T., Baer, D., and Randal, J.M., "GLYPNIR--a programming language for ILLIAC IV," Comm. ACM 18 (1975), 157-164.

[12] Nash, J.C., "A one-sided transformation method for the singular value decomposition and algebraic eigenproblem," Computer J. 18 (1975), 74-76.

[13] Rutishauser, H., "The Jacobi method for real symmetric matrices," Numer. Math. 9 (1966), 1-10.

[14] Sameh, A.H., and Kuck, D.J., "A parallel QR algorithm for tridiagonal symmetric matrices," Report, Computer Science Dept., University of Illinois, Urbana-Champaign (July 1974).

[15] Stevens, K.G., Jr., "CFD--a FORTRAN-like language for the ILLIAC IV," ACM Sigplan Notices 10 (1975), 72-76.

[16] Stewart, G.W., private communication (1978).

[17] Wilkinson, J.H., "Note on the quadratic convergence of the cyclic Jacobi process," Numer. Math. 4 (1962), 296-300.

[18] Wilkinson, J.H., The Algebraic Eigenvalue Problem, Clarendon, Oxford (1965).