1998

Approximate in inverse preconditioners for the conjugate gradient method

Chieko Honma
San Jose State University

Follow this and additional works at: https://scholarworks.sjsu.edu/etd_theses

Recommended Citation

Honma, Chieko, "Approximate in inverse preconditioners for the conjugate gradient method" (1998).
Master's Theses. 1751.
DOI: <https://doi.org/10.31979/etd.n8ur-yak6>
https://scholarworks.sjsu.edu/etd_theses/1751

This Thesis is brought to you for free and open access by the Master's Theses and Graduate Research at SJSU ScholarWorks. It has been accepted for inclusion in Master's Theses by an authorized administrator of SJSU ScholarWorks. For more information, please contact scholarworks@sjsu.edu.

INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI

A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor MI 48106-1346 USA
313/761-4700 800/521-0600

**APPROXIMATE INVERSE PRECONDITIONERS
FOR
THE CONJUGATE GRADIENT METHOD**

A Thesis

Presented to

The Faculty of the Department of Mathematics and Computer Science

San Jose State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Chieko Honma

December 1998

UMI Number: 1392816

**Copyright 1998 by
Honma, Chieko**

All rights reserved.

**UMI Microform 1392816
Copyright 1999, by UMI Company. All rights reserved.**

**This microform edition is protected against unauthorized
copying under Title 17, United States Code.**

UMI
300 North Zeeb Road
Ann Arbor, MI 48103

© 1998

Chieko Honma

ALL RIGHTS RESERVED

APPROVED FOR THE DEPARTMENT OF MATHEMATICS
AND COMPUTER SCIENCE

Mohammad Saleem

Dr. Mohammad Saleem

Leslie Foster

Dr. Leslie Foster

Samih Obaid

Dr. Samih Obaid

APPROVED FOR THE UNIVERSITY

William Fish

ABSTRACT

APPROXIMATE INVERSE PRECONDITIONERS FOR THE CONJUGATE GRADIENT METHOD

by
Chieko Honma

The method of Conjugate Gradients is known to converge for symmetric positive definite systems of equations. In this thesis, the method is applied to non-symmetric and ill-conditioned matrices. In order to facilitate convergence, an approximate inverse is used to precondition the Conjugate Gradient method. Although several preconditioning techniques are available, we use Newton's method and the Global Minimal Residual. This thesis introduces three versions of Newton's method to compute the approximate inverse and compares the convergence of each method. Also, the thesis experiments with different parameters for the initial guess of the approximate inverse for the Global Minimal Residual method and compares convergence properties.

ACKNOWLEDGMENT

I would like to thank Dr. Saleem for his guidance. I highly appreciate the countless hours he contributed and tiresome efforts he directed towards my thesis. His patience and willingness really helped me continue the work. I would also like to thank Dr. Foster and Dr. Obaid. My delay in completing the thesis caused inconvenience to them during the busiest period of the semester; nevertheless, they read the document thoroughly and offered me valuable suggestions and advice. From the bottom of my heart, I would like to express my gratitude for these three professors' support and contribution.

Also, I would like to thank all my friends for the friendship. The encouragement I received from them was one of the factors for which I was able to complete this thesis.

Finally, I would like to thank my family for their support from the other side of the Pacific. I hope that my completing the thesis will ease my parents' anxiety slightly.

TABLE OF CONTENTS

	Page
List of Tables.....	viii
List of Figures	x
Introduction	1
Chapter I. The Conjugate Gradient Method	4
1.1. Basic Idea of the Conjugate Gradient Method.....	4
1.2. A-Conjugate Gradient Search Directions	6
1.3. Derivations of the Conjugate Gradient Method	8
1.4. Algorithm and Implementation of the Conjugate Gradient Method.....	13
1.5. Convergence of the Conjugate Gradient Method.....	14
Chapter II. Preconditioning.....	17
2.1. Introduction to Preconditioners	17
2.2. Newton's Method.....	19
2.2.1. Acceleration by Scaling.....	20
2.2.2. Acceleration by Another Scaling	21
2.3. The Global Minimal Residual Method.....	23
2.4. Other Approximate Inverses	25
2.5. The Conjugate Gradient Method on the Normal Equation.....	25
2.6. The Preconditioned Conjugate Gradient Method of MATLAB	27
Chapter III. Numerical Results.....	28
3.1. The Conjugate Gradient Method on Symmetric and Non-Symmetric Matrices	28
3.1.1. The Conjugate Gradient Method on Symmetric Positive Definite	28
3.1.2. The Conjugate Gradient Method on Non-Symmetric Positive Definite	30
3.2. Newton's Method.....	32
3.2.1. Comparison of Acceleration Parameters for Norm-Scaled Newton's Method	32
3.2.2. Hilbert Matrix	34
3.2.3. Diagonally Dominant Matrix	36
3.2.4. Tridiagonal Matrix 1	38
3.2.5. Tridiagonal Matrix 2	40
3.2.6. Tridiagonal Matrix 3	41
3.2.7. Tridiagonal Matrix 4	43
3.2.8. Newton's Methods Summary	44
3.3. The Global Minimal Residual Method.....	47
3.3.1. Hilbert Matrix	48
3.3.2. Diagonally Dominant Matrix	49
3.3.3. Tridiagonal Matrix 1	51
3.3.4. Tridiagonal Matrix 2	53

3.3.5. Tridiagonal Matrix 3	55
3.3.6. Tridiagonal Matrix 4	57
3.3.7. Summary of GMR.....	59
3.4. Preconditioned Conjugate Gradient Method on Large Matrices	62
3.4.1. Hilbert Matrix	63
3.4.2. Example of Sparse Matrix 1	68
3.4.3. Example of Sparse Matrix 2	76
3.4.4. Example of Sparse Matrix 3	84
Chapter IV. Conclusion and Future Work.....	89
APPENDIX	92
REFERENCES.....	98

LIST OF TABLES

Table	Page
3.1 Non-preconditioned CG with $\mathbf{x}_0^T = [0 \ 0 \ 0 \ 0 \ 0]$	29
3.2 Non-preconditioned CG with $\mathbf{x}_0^T = 5*[1 \ 1 \ 1 \ 1 \ 1]$	29
3.3 Non-preconditioned CG with $\mathbf{x}_0^T = [\text{1st column of A}]$	29
3.4 Non-preconditioned CG with $\mathbf{x}_0^T = [\text{random}]$	30
3.5 Non-preconditioned CG on non-symmetric A with $\mathbf{x}_0^T = [0 \ 0 \ 0 \ 0 \ 0]$	30
3.6 Non-preconditioned CG on non-symmetric A with $\mathbf{x}_0^T = 5*[1 \ 1 \ 1 \ 1 \ 1]$	31
3.7 Non-preconditioned CG on non-symmetric A with $\mathbf{x}_0^T = [\text{1st column of A}]$	31
3.8 Comparison of Acceleration Parameters	32
3.9 Approximate Inverse of Hilbert Matrix by Newton's Methods.....	35
3.10 CG with Newton's Methods on Hilbert Matrix.....	35
3.11 Approximate Inverse of a Diagonally Dominant Matrix by Newton's Methods	37
3.12 CG with Newton's Methods on a Diagonally Dominant Matrix	37
3.13 Approximate Inverse of Tridiagonal Matrix 1 by Newton's Methods	39
3.14 CG with Newton's Methods on Tridiagonal Matrix 1	39
3.15 Approximate Inverse of Tridiagonal Matrix 2 by Newton's Methods	40
3.16 CG with Newton's Methods on Tridiagonal Matrix 2	40
3.17 Approximate Inverse of Tridiagonal Matrix 3 by Newton's Methods	42
3.18 CG with Newton's Methods on Tridiagonal Matrix 3	42
3.19 Approximate Inverse of Tridiagonal Matrix 4 by Newton's Methods	43
3.20 CG with Newton's Methods on Tridiagonal Matrix 4	43
3.21 Summary of CG with Newton's Methods	45
3.22 GMR on 5×5 Hilbert Matrix.....	48
3.23 GMR on 30×30 Hilbert Matrix.....	49
3.24 CGGMR on 30×30 Hilbert Matrix.....	49
3.25 GMR on 5×5 Diagonally Dominant Matrix.....	50
3.26 GMR on 30×30 Diagonally Dominant Matrix.....	50
3.27 CGGMR on 30×30 Diagonally Dominant Matrix.....	50
3.28 GMR on 5×5 Tridiagonal Matrix 1	51
3.29 GMR on 30×30 Tridiagonal Matrix 1	52
3.30 CGGMR on 30×30 Tridiagonal Matrix 1	52
3.31 GMR on 5×5 Tridiagonal Matrix 2.....	53
3.32 GMR on 30×30 Tridiagonal Matrix 2.....	54
3.33 CGGMR on 30×30 Tridiagonal Matrix 2.....	54
3.34 GMR on 5×5 Tridiagonal Matrix 3.....	55
3.35 GMR on 30×30 Tridiagonal Matrix 3	56
3.36 CGGMR on 30×30 Tridiagonal Matrix 3	56
3.37 GMR on 5×5 Tridiagonal Matrix 4.....	57
3.38 GMR on 30×30 Tridiagonal Matrix 4.....	58

3.39	CGGMR on 30×30 Tridiagonal Matrix 3	58
3.40	CGGMR Summary	60
3.41	CG on 128×128 Hilbert Matrix	63
3.42	CG on 128×128 Sparse Matrix 1	70
3.43	CG on 128×128 Sparse Matrix 2	79
3.44	CG on 128×128 Sparse Matrix 3	85

LIST OF FIGURES

Figures	Page
1.1 Gram-Schmidt Conjugation	9
1.2 Subspace in the Conjugate Gradient Method	11
3.1 CGNN(0.1) on 128×128 Hilbert Matrix	65
3.2 CGNN(0.01) on 128×128 Hilbert Matrix	65
3.3 CGN1 on 128×128 Hilbert Matrix.....	66
3.4 CGN2 on 128×128 Hilbert Matrix.....	66
3.5 CGGMR 0.5 on 128×128 Hilbert Matrix.....	67
3.6 CGGMR 0.5 on 128×128 Hilbert Matrix.....	67
3.7 Sparsity Pattern of 128×128 Sparse Matrix 1 from Nozzle Flow.....	69
3.8 CGNN on 128×128 Sparse Matrix 1 from Nozzle Flow	72
3.9 CGN1 on 128×128 Sparse Matrix 1 from Nozzle Flow	73
3.10 CGN2 on 128×128 Sparse Matrix 1 from Nozzle Flow	74
3.11 CGGMR 0.5 on 128×128 Sparse Matrix 1 from Nozzle Flow	75
3.12 CGGMR 0.05 on 128×128 Sparse Matrix 1 from Nozzle Flow.....	75
3.13 Sparsity Pattern of 128×128 Sparse Matrix 2	78
3.14 CGNN on 128×128 Sparse Matrix 2	81
3.15 CGN1 on 128×128 Sparse Matrix 2	81
3.16 CGN2 on 128×128 Sparse Matrix 2	82
3.17 CGGMR 0.5 on 128×128 Sparse Matrix 2	82
3.18 CGGMR 0.05 on 128×128 Sparse Matrix 2	83
3.19 CGNN on 128×128 Sparse Matrix 3	86
3.20 CGN1 on 128×128 Sparse Matrix 3	86
3.21 CGN2 on 128×128 Sparse Matrix 3	87
3.22 CGGMR 0.5 on 128×128 Sparse Matrix 3	87
3.23 CGGMR 0.05 on 128×128 Sparse Matrix 3	88

INTRODUCTION

The Conjugate Gradient method is a popular iterative method for solving a system of linear equations,

$$\mathbf{Ax} = \mathbf{b}$$

where $\mathbf{A}$ is a $n \times n$ square, symmetric, positive-definite matrix, $\mathbf{x} \in \mathbb{R}^n$ is an unknown vector, and $\mathbf{b} \in \mathbb{R}^n$ is known vector. The method was first introduced by Hestenes and Stiefel in 1952, but it took some time before it became widely used as a standard solver for many real-world problems [7]. If $\mathbf{A}$ is symmetric and positive definite, then the Conjugate Gradient method is known to converge in at most n iterations, where n is the dimension of $\mathbf{A}$. On the other hand, if $\mathbf{A}$ is not symmetric, the Conjugate Gradient method is not effective. This thesis deals with non-symmetric matrices, and some are ill-conditioned. In order to facilitate convergence, preconditioning is applied.

Preconditioners can be categorized into two groups - explicit and implicit. For both explicit and implicit preconditioning, several techniques have been developed to accelerate the method of Conjugate Gradient [7]. This thesis employs an explicit preconditioner in the form of an approximate inverse. If $\mathbf{M}$ denotes this approximate inverse, then the preconditioned system becomes

$$\mathbf{AMx} = \mathbf{Mb}$$

where $\mathbf{M}$ is an approximate inverse of $\mathbf{A}$.

Newton's method [23] and the Global Minimal Residual [11] method are used to compute the approximate inverse. Both methods are based on a very simple algorithm.

This thesis examines three variations of Newton's method and compares their numerical results. One of the variations introduced here is not found in the literature, and the potential of this method will be explored. Also, the thesis examines different parameters for the initial guess of the approximate inverse $\alpha\mathbf{M}$ for the Global Minimal Residual method, and compares their convergence properties.

This thesis applies the Conjugate Gradient method to systems of equations involving the Hilbert matrix, several tridiagonal matrices, and diagonally dominant matrices. The Hilbert matrix is often used in interpolation problems, while diagonally dominant matrices appear in the theory of vibrations in a string or beam. Tridiagonal matrices are related to partial differential equations such as the parabolic, elliptic, and hyperbolic equations. Examples of Navier Stokes Equation and the Heat Equation problem are also presented.

When $\mathbf{A}$ is not a square matrix, the least squares problem consists of minimizing the functional,

$$F(\mathbf{x}) = \min_x \|\mathbf{Ax} - \mathbf{b}\|^2.$$

Setting the derivative of the above equation to zero, normal equations can be obtained, as:

$$\mathbf{A}^T \mathbf{Ax} = \mathbf{A}^T \mathbf{b}$$

These normal equations can be solved by the Conjugate Gradient method [2], [7], [17], [18], [25]. This method is applied to large systems of normal equations and the numerical results are compared to those of preconditioned Conjugate Gradient method.

Commercial software MATLAB is equipped with a function, **pcg**, which uses preconditioned Conjugate Gradient method. The function **pcg** uses the implicit preconditioning method. Section 3.4 presents the numerical results by **pcg** just for reference.

The thesis is organized as follows: Chapter 1 introduces the basic theory of the Conjugate Gradient method, derives the method, and develops the algorithm. In Chapter 2, preconditioning is introduced, and algorithms for Newton's method and the Global Minimal Residual method are presented. Chapter 3, which contains the numerical results, demonstrates the effectiveness of the Conjugate gradient method for a symmetric positive definite system and how ineffective it is for a non-symmetric system. Then the chapter examines how the preconditioners can improve the efficiency of the Conjugate Gradient method for non-symmetric, ill-conditioned systems. The matrices studied are 5×5 , 30×30 , and 128×128 . The thesis is concluded with Chapter 4, which summarizes all the observations and discusses various possibilities for future work.

All the computations are performed by MATLAB on a PC with 266 MHz capability and 64M RAM.

CHAPTER I

The Conjugate Gradient Method

1.1. Basic Idea of the Conjugate Gradient Method

The Conjugate Gradient method (CG) is an iterative method for solving a system of linear equations,

$$\mathbf{Ax} = \mathbf{b}.$$

The idea of the conjugate gradient method is to solve $\mathbf{Ax} = \mathbf{b}$ by finding the minimum point of a quadratic function of a vector with the form

$$f(\mathbf{x}) = \frac{1}{2} \mathbf{x}^T \mathbf{Ax} - \mathbf{b}^T \mathbf{x} + c, \quad (1.1)$$

where $\mathbf{A}$ is an $n \times n$ matrix, $\mathbf{x}$ and $\mathbf{b}$ are vectors or points in $\mathbb{R}^n$, and c is a scalar constant.

By differentiating, the gradient vector of the above function is obtained.

$$f'(\mathbf{x}) = \frac{1}{2} \mathbf{A}^T \mathbf{x} + \frac{1}{2} \mathbf{Ax} - \mathbf{b}. \quad (1.2)$$

If $\mathbf{A}$ is symmetric, the above equation becomes

$$f'(\mathbf{x}) = \mathbf{Ax} - \mathbf{b}. \quad (1.3)$$

The minimum point is found by letting $f'(\mathbf{x}) = \mathbf{0}$, and by looking for a direction that is opposite to $f'(\mathbf{x})$; i.e., $-f'(\mathbf{x}) = \mathbf{b} - \mathbf{Ax}$. From an arbitrary point $\mathbf{x}_0$, a series of points $\mathbf{x}_1, \mathbf{x}_2, \dots$ is taken until a satisfactory solution is obtained. The residual $\mathbf{r}_i = \mathbf{b} - \mathbf{Ax}_i$ measures how well $\mathbf{x}_i$ fits the equation. Hence, the residual can be considered as the direction of steepest descent [25]. Notice that,

$$\mathbf{r}_i = -f'(\mathbf{x}_i). \quad (1.4)$$

Also, since the error $\mathbf{e}_i = \mathbf{x}_i - \mathbf{x}$, the residual can be expressed in terms of the error.

$$\begin{aligned} \mathbf{r}_i &= \mathbf{b} - \mathbf{A}\mathbf{x}_i \\ &= \mathbf{b} - \mathbf{A}(\mathbf{x} + \mathbf{e}_i). \end{aligned}$$

Hence,
$$\mathbf{r}_i = -\mathbf{A}\mathbf{e}_i \quad (1.5)$$

In other words, the residual can be considered as being the error transformed by $\mathbf{A}$ into the same space as $\mathbf{b}$ [25].

1.2. A-Conjugate Search Directions

The aim of the Conjugate Gradient method is to construct vectors called conjugate search directions.

Definition:

Two nonzero vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ are said to be conjugate with respect to the positive definite n by n matrix or $\mathbf{A}$ -conjugate [8], or $\mathbf{A}$ -orthogonal [25] if $\mathbf{x}^T \mathbf{A} \mathbf{y} = 0$.

Basic Properties:

The basic properties of conjugate directions [8] and their proofs are as follows:

Property 1: $\mathbf{A}$ -conjugate directions are linearly independent.

Proof: Let $\mathbf{A}$ be an $n \times n$ positive definite matrix. Let $\{\mathbf{d}_0, \mathbf{d}_1, \dots, \mathbf{d}_{n-1}\}$ be a set of $\mathbf{A}$ -conjugate directions, where $\mathbf{d}_0, \mathbf{d}_1, \dots, \mathbf{d}_{n-1} \in \mathbb{R}^n$, and $a_0 \mathbf{d}_0 + a_1 \mathbf{d}_1 + \dots + a_{n-1} \mathbf{d}_{n-1} = \mathbf{0}$, where a_i are scalars.

Multiplying the equation by $\mathbf{d}_0^T \mathbf{A}$,

$$\mathbf{d}_0^T \mathbf{A} (a_0 \mathbf{d}_0 + a_1 \mathbf{d}_1 + \dots + a_{n-1} \mathbf{d}_{n-1}) = 0.$$

$a_0 \mathbf{d}_0^T \mathbf{A} \mathbf{d}_i = 0$ for each i except $i=0$ since $\mathbf{d}$ -vectors are $\mathbf{A}$ -conjugate.

Hence, $a_0 \mathbf{d}_0^T \mathbf{A} \mathbf{d}_0 = 0$. Repeating the same procedure using the rest of the set yields $a_i \mathbf{d}_i^T \mathbf{A} \mathbf{d}_i = 0, \forall i$. Since $\mathbf{A}$ is positive definite, $\mathbf{d}_i^T \mathbf{A} \mathbf{d}_i > 0$. This implies $a_i = 0, \forall i$.

Therefore, $\mathbf{d}_0, \mathbf{d}_1, \dots, \mathbf{d}_{n-1}$ are linearly independent.

Property 2: A set of $\mathbf{A}$ -conjugate directions has at most n members so that if $\mathbf{d}_0, \mathbf{d}_1, \dots, \mathbf{d}_{n-1}$ are $\mathbf{A}$ -conjugate and $\mathbf{x}^T \mathbf{A} \mathbf{d}_i = 0$ for each i , then $\mathbf{x} = 0$.

Proof: From property (1), $\{\mathbf{d}_0, \mathbf{d}_1, \dots, \mathbf{d}_{n-1}\}$ is a linearly independent set, which implies the set contains at most n vectors.

Let $\mathbf{x} \notin \{\mathbf{d}_0, \mathbf{d}_1, \dots, \mathbf{d}_{n-1}\}$. Suppose $\mathbf{x}^T \mathbf{A} \mathbf{d}_i = 0$ for each i . But $\mathbf{A} \neq 0$ and $\mathbf{d}_i \neq 0$.

Hence $\mathbf{x} = 0$.

Property 3: If $\mathbf{d}_0, \mathbf{d}_1, \dots, \mathbf{d}_{n-1}$ are $\mathbf{A}$ -conjugate, then they form a basis for $\mathbb{R}^n$, and if

$$\mathbf{x} \in \mathbb{R}^n, \text{ then } \mathbf{x} \text{ is given in terms of this basis by } \mathbf{x} = \sum_{i=0}^{n-1} \frac{\mathbf{x}^T \mathbf{A} \mathbf{d}_i}{\mathbf{d}_i^T \mathbf{A} \mathbf{d}_i} \mathbf{d}_i.$$

Proof: From properties (1) and (2), a set of $\mathbf{A}$ -conjugate directions has linearly independent n vectors. Thus, the set forms a basis for $\mathbb{R}^n$. Expressing $\mathbf{x} \in \mathbb{R}^n$ as a linear combination of $\mathbf{A}$ -conjugate vectors,

$$\mathbf{x} = a_0 \mathbf{d}_0 + a_1 \mathbf{d}_1 + \dots + a_{n-1} \mathbf{d}_{n-1}.$$

Taking the transpose, $\mathbf{x}^T = (a_0 \mathbf{d}_0 + a_1 \mathbf{d}_1 + \dots + a_{n-1} \mathbf{d}_{n-1})^T$.

Multiplying by $\mathbf{A}\mathbf{d}_0$,

$$\begin{aligned}\mathbf{x}^T \mathbf{A}\mathbf{d}_0 &= (a_0 \mathbf{d}_0^T + a_1 \mathbf{d}_1^T + \dots + a_{n-1} \mathbf{d}_{n-1}^T) \mathbf{A}\mathbf{d}_0 \\ &= a_0 \mathbf{d}_0^T \mathbf{A}\mathbf{d}_0 \quad \text{by A-conjugacy of } \mathbf{d}\text{-vectors.}\end{aligned}$$

Hence $a_0 = \frac{\mathbf{x}^T \mathbf{A}\mathbf{d}_0}{\mathbf{d}_0^T \mathbf{A}\mathbf{d}_0}$. Repeating the same procedure with $\mathbf{A}\mathbf{d}_1, \dots, \mathbf{A}\mathbf{d}_{n-1}$,

$$a_i = \frac{\mathbf{x}^T \mathbf{A}\mathbf{d}_i}{\mathbf{d}_i^T \mathbf{A}\mathbf{d}_i}, \forall i. \text{ Therefore, } \mathbf{x} = \sum_{i=0}^{n-1} \frac{\mathbf{x}^T \mathbf{A}\mathbf{d}_i}{\mathbf{d}_i^T \mathbf{A}\mathbf{d}_i} \mathbf{d}_i. \quad \square$$

1.3. Derivation of the Conjugate Gradient Method

The following derivation of the Conjugate Gradient Method is based on Shewchuk [25].

Suppose that we are going to solve a system of linear equations $\mathbf{A}\mathbf{x} = \mathbf{b}$. Define the iterates $\mathbf{x}_{i+1}$ as:

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \alpha_i \mathbf{d}_i, \quad (1.6)$$

where α_i is the step-length and $\mathbf{d}_i$ is the search direction. The step-length α_i are chosen to minimize f of equation (1.1).

$$\begin{aligned}\frac{d}{d\alpha_i} f(\mathbf{x}_{i+1}) &= f'(\mathbf{x}_{i+1})^T \frac{d}{d\alpha_i} \mathbf{x}_{i+1} \\ &= -\mathbf{r}_{i+1}^T \mathbf{d}_i \quad \text{by (1.4) and (1.6)} \\ &= -(\mathbf{b} - \mathbf{A}\mathbf{x}_{i+1})^T \mathbf{d}_i \\ &= -(\mathbf{b} - \mathbf{A}(\mathbf{x}_i + \alpha_i \mathbf{d}_i))^T \mathbf{d}_i \\ &= -(\mathbf{b} - \mathbf{A}\mathbf{x}_i - \mathbf{A}\alpha_i \mathbf{d}_i)^T \mathbf{d}_i\end{aligned} \quad (1.7)$$

$$\begin{aligned}
&= -(\mathbf{b} - \mathbf{Ax}_i)^\top \mathbf{d}_i + \alpha_i (\mathbf{Ad}_i)^\top \mathbf{d}_i \\
&= 0.
\end{aligned}$$

$$\text{Thus, } \alpha_i = \frac{\mathbf{r}_i^\top \mathbf{d}_i}{\mathbf{d}_i^\top \mathbf{A}^\top \mathbf{d}_i}. \quad (1.8)$$

Note equation (1.7) suggests that finding the minimum point requires $\mathbf{r}_{i+1}^\top \mathbf{d}_i = 0$. In the Conjugate Gradient method, a new residual is orthogonal to the previous search direction.

Next, a set of $\mathbf{A}$ -conjugate search directions $\{\mathbf{d}_i\}$ needs to be constructed. A conjugate Gram-Schmidt process (Figure 1.1) is applied.

Figure 1.1: Gram-Schmidt Conjugation

Set $\mathbf{d}_0 = \mathbf{u}_0$. $\mathbf{u}_1$ is composed of $\mathbf{u}^$, which is conjugate to $\mathbf{d}_0$, and $\mathbf{u}^+$, which is parallel to $\mathbf{d}_0$. After conjugation, only the conjugate portion remains.*

Let $\{\mathbf{u}_0, \mathbf{u}_1, \dots, \mathbf{u}_{n-1}\}$ be a set of n linearly independent vectors. Let $\mathbf{d}_0 = \mathbf{u}_0$, and for $i > 0$, $\mathbf{d}_i$'s are constructed by the following equation:

$$\mathbf{d}_i = \mathbf{u}_i + \sum_{k=0}^{i-1} \beta_{ik} \mathbf{d}_k, \quad (1.9)$$

where the β_{ik} are defined for $i > k$. Now find β_{ik} . By taking the transpose of (1.9) and multiplying by $\mathbf{Ad}_j$,

$$\mathbf{d}_i^T \mathbf{A} \mathbf{d}_j = \mathbf{u}_i^T \mathbf{A} \mathbf{d}_j + \sum_{k=0}^{i-1} \beta_{ik} \mathbf{d}_k^T \mathbf{A} \mathbf{d}_j$$

$$0 = \mathbf{u}_i^T \mathbf{A} \mathbf{d}_j + \beta_{ij} \mathbf{d}_j^T \mathbf{A} \mathbf{d}_j \quad i > j \text{ (by A-conjugacy of } \mathbf{d}\text{-vectors)}$$

$$\beta_{ij} = -\frac{\mathbf{u}_i^T \mathbf{A} \mathbf{d}_j}{\mathbf{d}_j^T \mathbf{A} \mathbf{d}_j}. \quad (1.10)$$

CG is a method where search directions $\mathbf{d}_i$'s are in $\text{span}(\mathbf{r}_0, \mathbf{r}_1, \dots, \mathbf{r}_{n-1})$, and the $\mathbf{d}_i$'s are A-conjugate. For now we will let the $\mathbf{u}_i$'s be general and on page 12 specialize $\mathbf{u}_i$'s so that $\mathbf{u}_i = \mathbf{r}_i$.

The β_{ij} terms can be simplified, but before attempting that, more properties of the residual should be found out. Earlier, it was shown that the residual can be considered as the direction of steepest descent (1.4) and as being the error transformed by $\mathbf{A}$ into the same space as $\mathbf{b}$ (1.5). Taking the definition of the residual,

$$\begin{aligned} \mathbf{r}_{i+1} &= \mathbf{b} - \mathbf{A} \mathbf{x}_{i+1} \\ &= \mathbf{b} - \mathbf{A}(\mathbf{e}_{i+1} - \mathbf{x}) \quad \text{since } \mathbf{e}_i = \mathbf{x}_i - \mathbf{x} \\ &= \mathbf{b} - \mathbf{A} \mathbf{e}_{i+1} + \mathbf{A} \mathbf{x} \\ &= -\mathbf{A} \mathbf{e}_{i+1} \\ &= -\mathbf{A}(\mathbf{e}_i + \alpha_i \mathbf{d}_i) \quad \text{since } \mathbf{x}_{i+1} - \mathbf{x} = \mathbf{x}_i - \mathbf{x} + \alpha_i \mathbf{d}_i \\ \mathbf{r}_{i+1} &= \mathbf{r}_i - \alpha_i \mathbf{A} \mathbf{d}_i. \end{aligned} \quad (1.11)$$

Before continuing, it needs to be shown that CG converges in n steps [8], [13], [25]. Expressing the error term as a linear combination of search directions,

$$\mathbf{e}_0 = \sum_{j=0}^{n-1} \delta_j \mathbf{d}_j. \quad (1.12)$$

Multiplying (1.12) by $\mathbf{d}_k^T \mathbf{A}$, the following is obtained:

$$\mathbf{d}_k^T \mathbf{A} \mathbf{e}_0 = \delta_k \mathbf{d}_k^T \mathbf{A} \mathbf{d}_k \quad \text{by A-conjugacy of } \mathbf{d}$$

$$\delta_k = \frac{\mathbf{d}_k^T \mathbf{A} \mathbf{e}_0}{\mathbf{d}_k^T \mathbf{A} \mathbf{d}_k}$$

$$= \frac{\mathbf{d}_k^T \mathbf{A} \left(\mathbf{e}_0 + \sum_{i=0}^{k-1} \alpha_i \mathbf{d}_i \right)}{\mathbf{d}_k^T \mathbf{A} \mathbf{d}_k} \quad \text{by A-conjugacy of } \mathbf{d}$$

$$= \frac{\mathbf{d}_k^T \mathbf{A} \mathbf{e}_k}{\mathbf{d}_k^T \mathbf{A} \mathbf{d}_k}.$$

By equation (1.5), (1.8) becomes

$$\begin{aligned} \alpha_i &= \frac{\mathbf{r}_i^T \mathbf{d}_i}{\mathbf{d}_i^T \mathbf{A}^T \mathbf{d}_i} \\ &= \frac{-(\mathbf{A} \mathbf{e}_i)^T \mathbf{d}_i}{\mathbf{d}_i^T \mathbf{A}^T \mathbf{d}_i}. \end{aligned}$$

Hence $\alpha_i = \alpha_i^T = -\delta_i$. Using this result in (1.12),

$$\mathbf{e}_i = \mathbf{e}_0 + \sum_{j=0}^{i-1} \alpha_j \mathbf{d}_j \quad \text{by recursion of (1.6).}$$

Thus,

$$\mathbf{e}_i = \sum_{j=0}^{n-1} \delta_j \mathbf{d}_j - \sum_{j=0}^{i-1} \delta_j \mathbf{d}_j \quad \text{since } \alpha_i = -\delta_i,$$

or
$$\mathbf{e}_i = \sum_{j=i}^{n-1} \delta_j \mathbf{d}_j .$$

Thus, $\mathbf{e}_n = 0$. CG converges at most in n iterations if the matrix $\mathbf{A}$ is symmetric positive definite. Since the iterations are finite, CG will stop in a finite number of steps even for large matrices.

Now, using the above result, (1.5) can be written as

$$\begin{aligned} \mathbf{r}_i &= -\mathbf{A}\mathbf{e}_i \\ &= -\sum_{k=i}^{n-1} \delta_k \mathbf{A}\mathbf{d}_k . \end{aligned} \tag{1.13}$$

By multiplying (1.13) by $\mathbf{d}_j^T$,
$$\mathbf{d}_j^T \mathbf{r}_i = -\mathbf{d}_j^T \sum_{k=i}^{n-1} \delta_k \mathbf{A}\mathbf{d}_k$$

$$\mathbf{d}_j^T \mathbf{r}_i = 0 \quad i > j \quad \text{by } \mathbf{A}\text{-conjugacy of } \mathbf{d}\text{-vectors.} \tag{1.14}$$

(1.14) suggests that the residuals are orthogonal to all the previous residuals. Multiplying the transpose of (1.9) by $\mathbf{r}_j$,

$$\mathbf{d}_i^T \mathbf{r}_j = \mathbf{u}_i^T \mathbf{r}_j + \sum_{k=0}^{i-1} \beta_{ik} \mathbf{d}_k^T \mathbf{r}_j \quad i < j.$$

Thus,
$$0 = \mathbf{u}_i^T \mathbf{r}_j, \quad i < j \text{ by (1.14).} \tag{1.15}$$

By (1.14) and (1.15),

$$\mathbf{d}_i^T \mathbf{r}_j = \mathbf{u}_i^T \mathbf{r}_j. \tag{1.16}$$

This result will be used later.

CG is simply the method of conjugate directions where the search directions are constructed by conjugation of the residuals [8]. Hence, setting $u_i = r_i$, equation (1.15) becomes $r_i^T r_j = 0$, which implies that the two residuals are orthogonal.

Figure 1.2: Subspace in the Conjugate Gradient Method

Each new residual (r_1, r_2) is orthogonal to all the previous residuals and search directions. A new search direction (d_1, d_2) is constructed to be A-orthogonal to all the previous residuals and search directions. The endpoints of d_2 and r_2 lie on a plane parallel to shaded subspace because d_2 is constructed by r_2 . The vector d_2 is a linear combination of d_1 and r_2 .

From the above discussion, the following can be said about the residuals and the search directions in the Conjugate Gradient method [13], [25]: A new residual is orthogonal to all the previous residuals and search directions; A new search direction is constructed from the residual to be conjugate to all the previous residuals and search directions.

Now, going back to the equation (1.10), simplify β_{ij} . Since $u_i = r_i$,

$$\beta_{ij} = -\frac{r_i^T A d_j}{d_j^T A d_j}. \quad (1.17)$$

Replacing j by i in (1.11) and multiplying the result with $i = j$ by r_i^T ,

$$r_i^T r_{j+1} = r_i^T r_j - r_i^T \alpha_j A d_j,$$

or

$$\alpha_j \mathbf{r}_i^T \mathbf{A} \mathbf{d}_j = \mathbf{r}_i^T \mathbf{r}_j - \mathbf{r}_i^T \mathbf{r}_{j+1}.$$

The last relation implies

$$\mathbf{r}_i^T \mathbf{A} \mathbf{d}_j = \begin{cases} \frac{1}{\alpha_j} \mathbf{r}_i^T \mathbf{r}_j, & i = j \\ -\frac{1}{\alpha_{i-1}} \mathbf{r}_i^T \mathbf{r}_i, & i = j+1 \\ 0, & \text{otherwise.} \end{cases}$$

Thus, (1.17) becomes

$$\beta_{ij} = \begin{cases} \frac{1}{\alpha_{i-1}} \frac{\mathbf{r}_i^T \mathbf{r}_i}{\mathbf{d}_{i-1}^T \mathbf{A} \mathbf{d}_{i-1}}, & i = j+1 \\ 0, & i > j+1. \end{cases} \quad (1.17a)$$

By (1.11),

$$\mathbf{r}_i = \mathbf{r}_{i-1} - \alpha_{i-1} \mathbf{A} \mathbf{d}_{i-1}$$

$$\alpha_{i-1} \mathbf{A} \mathbf{d}_{i-1} = \mathbf{r}_{i-1} - \mathbf{r}_i.$$

Multiplying by $\mathbf{d}_i^T$,

$$\begin{aligned} \alpha_{i-1} \mathbf{d}_{i-1}^T \mathbf{A} \mathbf{d}_{i-1} &= \mathbf{d}_{i-1}^T \mathbf{r}_{i-1} - \mathbf{d}_{i-1}^T \mathbf{r}_i \\ &= \mathbf{d}_{i-1}^T \mathbf{r}_{i-1} \quad \text{by (1.14).} \end{aligned} \quad (1.17b)$$

Letting $\beta_i = \beta_{i,i-1}$, (1.17a) becomes

$$\beta_i = \frac{\mathbf{r}_i^T \mathbf{r}_i}{\mathbf{d}_{i-1}^T \mathbf{r}_{i-1}}. \quad (1.17c)$$

Also, by (1.9)

$$\begin{aligned} \mathbf{r}_{i-1}^T \mathbf{d}_{i-1} &= \mathbf{r}_{i-1}^T (\mathbf{r}_{i-1} + \sum_{k=0}^{i-2} \beta_{ik} \mathbf{d}_k) \\ &= \mathbf{r}_{i-1}^T \mathbf{r}_{i-1} \quad \text{by (1.14).} \end{aligned}$$

Hence, (1.17c) becomes as

$$\beta_i = \frac{\mathbf{r}_i^T \mathbf{r}_i}{\mathbf{r}_{i-1}^T \mathbf{r}_{i-1}} \quad \text{by (1.16).} \quad (1.18)$$

Putting all these pieces together, the outline of the Conjugate Gradient method is as follows:

1. Choose an initial guess $\mathbf{x}_0$ to $\mathbf{Ax} = \mathbf{b}$.

2. $\mathbf{d}_0 = \mathbf{r}_0 = \mathbf{b} - \mathbf{Ax}_0$

3. $\alpha_i = \frac{\mathbf{r}_i^T \mathbf{r}_i}{\mathbf{d}_i^T \mathbf{A}^T \mathbf{d}_i}$ by (1.8)

4. $\mathbf{x}_{i+1} = \mathbf{x}_i + \alpha_i \mathbf{d}_i$ by (1.6)

5. $\mathbf{r}_{i+1} = \mathbf{r}_i - \alpha_i \mathbf{A} \mathbf{d}_i$ by (1.11)

6. $\beta_{i+1} = \frac{\mathbf{r}_{i+1}^T \mathbf{r}_{i+1}}{\mathbf{r}_i^T \mathbf{r}_i}$ by (1.18)

7. $\mathbf{d}_{i+1} = \mathbf{r}_{i+1} + \beta_{i+1} \mathbf{d}_i$ by (1.9) and (1.17b)

Go to 3.

1.4. Algorithm and Implementation of the Conjugate Gradient Method

The above algorithm needs to be modified slightly to allow introduction of a preconditioner $\mathbf{M}$ which is discussed later. If $\mathbf{M} = \mathbf{I}$, then Algorithm 1.1 is the same as steps 1 to 7 above with some minor rewriting of some of the form [2].

The algorithm of the Conjugate Gradient Method, which is based on [2], is as follows:

```

Compute  $\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0$  for some initial guess  $\mathbf{x}_0$ 
for  $i = 1, 2, \dots$ ,
    solve  $\mathbf{M}\mathbf{z}_{i-1} = \mathbf{r}_{i-1}$  *
     $\rho_{i-1} = \mathbf{r}_{i-1}^T \mathbf{z}_{i-1}$ 
    if  $i = 1$ 
         $\mathbf{d}_1 = \mathbf{z}_0$ 
    else
         $\beta_{i-1} = \rho_{i-1} / \rho_{i-2}$ 
         $\mathbf{d}_i = \mathbf{z}_{i-1} + \beta_{i-1} \mathbf{d}_{i-1}$ 
    endif
     $\mathbf{q}_i = \mathbf{A}\mathbf{d}_i$ 
     $\alpha_i = \rho_{i-1} / \mathbf{d}_i^T \mathbf{q}_i$ 
     $\mathbf{x}_i = \mathbf{x}_{i-1} + \alpha_i \mathbf{d}_i$ 
     $\mathbf{r}_i = \mathbf{r}_{i-1} - \alpha_i \mathbf{q}_i$ 

```

Algorithm 1.1 The Conjugate Gradient Method

**The matrix $\mathbf{M}$ is a preconditioner, which will be discussed in a later section. Without a preconditioner, $\mathbf{M} = \mathbf{I}$ [2], [13], [25].*

The Conjugate Gradient Method without a preconditioner involves one matrix-vector multiplication and two inner products per iteration. The computation is relatively simple. Also, three vectors are updated during each iteration, which requires less memory. The flops of the method will be computed in Chapter 3.

1.5. Convergence of the Conjugate Gradient Method

The previous section has already discussed the convergence of CG, but this section will present more theoretical aspects. The theoretical convergence of CG has been

studied by many authors [1], [20], [25], [27], [28]. Chebyshev polynomials were used to estimate the rate of convergence [20], [25], and it was found that the convergence rate depends on the distribution of eigenvalues of A . When eigenvalues are clustered together, CG converges faster [4], [9], [20], [27]. This can be seen from the error bound that is expressed in terms of the spectral condition number of A and the energy norm. The spectral condition number of A is defined as

$$\kappa_2 = \lambda_{\max}(A) / \lambda_{\min}(A), \quad (1.19)$$

where $\lambda_{\max}(A)$ and $\lambda_{\min}(A)$ are the largest and the smallest eigenvalues of A in absolute value, respectively. The energy norm of a vector y is defined as

$$\|y\|_A = (y^T A y)^{1/2}. \quad (1.20)$$

Suppose x is the exact solution of $Ax = b$. Then the following error bound can be obtained [2], [9], [14], [25].

$$\|x_i - x\|_A \leq 2\sigma^i \|x_0 - x\|_A, \quad (1.21)$$

where i is the number of iterations, and

$$\sigma = (\sqrt{\kappa_2} - 1) / (\sqrt{\kappa_2} + 1), \quad (1.22)$$

The inequality (1.21) suggests that the convergence is faster when σ is smaller. When eigenvalues are clustered around 1, σ becomes very small, but when they are clustered around 0, the error increases [20].

CHAPTER II

Preconditioning

2.1. Introduction to Preconditioners

As discussed earlier, the rate of convergence of CG depends on the spectral condition number $\kappa_2 = \lambda_{\max}(\mathbf{A}) / \lambda_{\min}(\mathbf{A})$ of the coefficient matrix. Using a preconditioner, the distribution of eigenvalues can be changed in such a way that the difference between $\lambda_{\max}$ and $\lambda_{\min}$ becomes small; e.i., $\kappa_2 \approx 1$. Hence, σ in (1.22) becomes small and the rate of convergence can be improved. Instead of $\mathbf{Ax} = \mathbf{b}$, in the preconditioned system, $\mathbf{AMx} = \mathbf{Mb}$ is solved. If $\mathbf{M} = \mathbf{A}^{-1}$, the system can be solved directly by $\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}$, but since it is not easy to compute $\mathbf{A}^{-1}$, instead, the approximate inverse $\mathbf{M} \approx \mathbf{A}^{-1}$ is computed.

When the size of the coefficient matrix $\mathbf{A}$ is large, preconditioning is helpful in reducing the number of the iterations. Also, when the coefficient matrix is ill-conditioned and thus the condition number is too large, a preconditioner can reduce the large condition number and make the CG method more efficient.

Preconditioners can be classified into two types - the implicit and the explicit types [5]. A preconditioner is implicit if it requires the solution of a linear system within each step of the iteration and will be expensive in terms of CPU time. With implicit preconditioning, nonsingular matrix $\mathbf{M} \approx \mathbf{A}^{-1}$ is defined implicitly, and applying the matrix requires that we solve a linear system with $\mathbf{M}$ [12]. On the other hand, in explicit

preconditioning, $\mathbf{M}$ is known. It is simple to implement [7] and the application of an explicit preconditioner requires matrix-vector products only.

The most important example of implicit preconditioning is an incomplete LU (ILU) decomposition. Here, $\mathbf{M} = \tilde{\mathbf{L}}\tilde{\mathbf{U}}$, where $\tilde{\mathbf{L}}$ and $\tilde{\mathbf{U}}$ are triangular matrices that approximate the $\mathbf{L}$ and $\mathbf{U}$ factors of $\mathbf{A}$. When this method is applied, two triangular systems need to be solved. Fortunately this is not expensive in computational terms. Other implicit preconditioning include Successive Overrelaxation (SOR) [22], Symmetric Successive Overrelaxation (SSOR) [2], [6], [22], and Incomplete Cholesky (IC) [4].

The most popular example of explicit preconditioning is based on polynomial approximations of $\mathbf{A}^{-1}$. The use of polynomial preconditioners based on Chebyshev polynomials started in 1959. Other methods such as Neumann series approximations were developed later [7], [14]. Another example of explicit preconditioning is Sparse Approximate Inverse, which directly constructs a matrix that approximates the inverse of $\mathbf{A}$. This method has been proven to be efficient [5], and even if the matrix $\mathbf{A}$ does not have a stable ILU decomposition, the construction of an approximate inverse is sometimes possible [5]. However, this method is rather expensive [5], [7], [9].

In this thesis, approximate inverses and other matrices are used as preconditioners. Newton's iteration and Global Minimal Residual iteration are used to compute right approximate inverses. These methods can be implemented before the main iteration loop in the algorithm of the CG method.

2.2 Newton's Method

This method is modeled after Newton's method for solving the scalar problem,

$f(x) = \frac{1}{x} - a$ used to compute the reciprocal of a number of without performing division

[11]. The iteration form is defined by

$$\mathbf{M}_{n+1} = \mathbf{M}_n(2\mathbf{I} - \mathbf{A}\mathbf{M}_n) \quad (2.1)$$

with the initial guess

$$\mathbf{M}_0 = \alpha \mathbf{A}^T. \quad (2.2)$$

The iterate (2.1) converges if

$$0 < \alpha < \frac{1}{\|\mathbf{A}\mathbf{A}^T\|_2}$$

is satisfied [3],[11].

$$\alpha = \frac{1}{\|\mathbf{A}\|_1 \|\mathbf{A}\|_\infty} \quad (2.3)$$

is suggested by Pan and Schreiber [23]. Ben-Israel and Cohen [3] suggest

$$\alpha = \frac{1}{\|\mathbf{A}\mathbf{A}^T\|_1}.$$

This thesis uses (2.3) so that a matrix-matrix multiplication is avoided.

Newton's iteration is numerically stable, but since it includes two matrix-matrix multiplications, the computation can be expensive. In order to reduce the cost, Chow and Saad [11] suggest to drop some elements in $\mathbf{M}$, but they warn that this may destroy the convergence properties of Newton's iteration. Pan and Schreiber [23] introduce two acceleration methods: by scaling and by cubic polynomials. The former method is

examined in this thesis. Another scaling method, not found in the literature, is also introduced in Section 2.2.2.

2.2.1. Acceleration by Scaling

The iteration is defined by

$$\mathbf{M}_{k+1} = \alpha_{k+1}(2\mathbf{I} - \mathbf{M}_k\mathbf{A})\mathbf{M}_k \quad k = 0, 1, \dots \quad (2.4)$$

where $\mathbf{M}_0$ is given by (2.2). This scaled iteration was first presented by Schreiber. In [23], Pan and Schreiber choose α_{k+1} so that it can minimize a bound on the maximum distance of any nonzero singular value of $\mathbf{M}_k\mathbf{A}$ from 1. In other words, $\alpha(\mathbf{M}_k\mathbf{A})$ needs to be close to 1 so that $2\mathbf{I} - \mathbf{M}_k\mathbf{A}$ will converge to $\mathbf{I}$. Assuming $\sigma_{\min}$ and $\sigma_{\max}$ such that

$$\sigma_{\min} \leq \alpha_1^2 \leq \sigma_1^2 \leq \sigma_{\max},$$

where α_1 and σ_1 are the smallest and largest singular values of $\mathbf{A}$, respectively, are known, they define

$$\alpha_0 = \frac{2}{\sigma_{\min} + \sigma_{\max}} \quad (2.5)$$

for $\mathbf{M}_0$ of (2.2), and take

$$\alpha_{k+1} = \frac{2}{1 + (2 - \rho_k)\rho_k}, \quad (2.6)$$

where $\rho_0 = \alpha_0\sigma_{\min}$ and $\rho_{k+1} = \alpha_{k+1}(2 - \rho_k)\rho_k$. See [25] for more details concerning the optimality of the acceleration parameters.

Since $\sigma_{\min}$ and $\sigma_{\max}$ are unknown, instead of (2.5), take

$$\alpha_0 = \frac{2}{\sigma_r^2 + \sigma_1^2}. \quad (2.7)$$

Putting the above equations together, the algorithm is constructed as follows:

Find singular values ($\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r$), of $\mathbf{A}$.

$$\alpha_0 = 2/(\sigma_r^2 + \sigma_1^2)$$

$$\rho_0 = \alpha_0 \sigma_r$$

$$\mathbf{M}_0 = \alpha_0 \mathbf{A}^\top$$

for $k = 1, 2, \dots$

$$\alpha_k = 2/[(1 + 2 - \rho_{k-1})\rho_{k-1}]$$

$$\mathbf{M}_k = \alpha(2\mathbf{I} - \mathbf{M}_{k-1}\mathbf{A})\mathbf{M}_{k-1}$$

$$\rho_k = \alpha(2 - \rho_{k-1})\rho_{k-1}$$

Algorithm 2.1 Accelerated Newton's Iteration I

The sequence of $\{\mathbf{M}_i\}$ converges to $\mathbf{A}^{-1}$.

Numerical results will be presented in the next chapter.

2.2.2. Acceleration by Another Scaling

This method uses the norms of $\mathbf{A}$ as an acceleration parameter. Unlike the former scaling method, this method uses the same iteration form (2.1) as the non-scaled Newton's iterations. Besides, while the former method updates its acceleration parameter at each iteration, the latter does not - it scales only before the iteration. Before the iteration, $\mathbf{A}$ is multiplied by

$$\frac{1}{\max(\|\mathbf{A}\|_1, \|\mathbf{A}\|_\infty)}, \quad (2.8)$$

where $\|A\|_1 = \max_{1 \leq j \leq n} \sum_{i=1}^n |a_{ij}|$ and $\|A\|_\infty = \max_{1 \leq i \leq n} \sum_{j=1}^n |a_{ij}|$. And then, after an approximate inverse

of the scaled matrix is computed, it is scaled again so that an approximate inverse of A is obtained. Here (2.8) is again used as the second scalar since

$$B = \beta A$$

$$A^{-1} = [(1/\beta) B]^{-1} = \beta B^{-1},$$

where $\beta = 1/\max(\|A\|_1, \|A\|_\infty)$.

The motivation for this approach is that Newton's method is often accelerated by a scalar as shown above. Also, it has been observed that a non-scaled Newton's iteration converges to the inverse of a matrix A when column sums of A are all ones. Matrices whose column sums are all ones are called probability matrices and often used in statistics.

The algorithm of the scaled Newton's iterations is as follows.

$$\begin{aligned} \beta &= 1/\max(\|A\|_1, \|A\|_\infty) \\ A &= \beta A \\ \alpha &= 1/\|A\|_1 \|A\|_\infty \\ M_0 &= \alpha A^T \\ \text{for } n &= 1, 2, \dots \\ M_{n+1} &= M_n(2I - AM_n) \\ A^{-1} &= \beta M_{n+1} \end{aligned}$$

Algorithm 2.2 Accelerated Newton's Method II

The sequence of $\{M_i\}$ converges to A^{-1} .

When $\beta = 1/\|A\|$ instead of (2.8), similar results are obtained. Numerical results are given in Chapter 3.

2.3. The Global Minimal Residual Method

The Global Minimal Residual method, presented by Chow and Saad [11], is used to compute the approximate inverse of a matrix A . This is based on minimizing the Frobenius norm of the residual matrix $R_n = I - AM_n$, where M_n is an approximate inverse. The algorithm for this method is similar to that of the Conjugate Gradient Method. The iteration is defined by taking a step along a steepest descent search direction matrix G as

$$M_{n+1} = M_n + \alpha G. \quad (2.9)$$

α is selected to minimize $\|I - AM_n\|_F^2$. Hence, using (2.9),

$$\begin{aligned} \frac{\partial}{\partial \alpha} \|I - A(M + \alpha G)\|_F^2 &= \frac{\partial}{\partial \alpha} \text{tr} \{ [I - A(M + \alpha G)]^T [I - A(M + \alpha G)] \} \\ &= \frac{\partial}{\partial \alpha} \text{tr} [(R - \alpha AG)^T (R - \alpha AG)] \\ &= 0 \end{aligned}$$

and

$$\alpha = \frac{\text{tr}(R^T AG)}{\text{tr}[(AG)^T AG]} \quad (2.10)$$

$$= \frac{\text{tr}(R^T AG)}{\|AG\|_F^2}. \quad (2.11)$$

The residual matrix $\mathbf{R}_n = \mathbf{I} - \mathbf{A}\mathbf{M}_n$ is suggested for the search direction [11]. Then (2.9)

$$\text{becomes } \alpha = \frac{\text{tr}(\mathbf{R}^T \mathbf{A} \mathbf{R})}{\text{tr}[(\mathbf{A} \mathbf{R})^T \mathbf{A} \mathbf{R}]} \quad (2.12)$$

After some iterations, the approximate inverse $\mathbf{M}$ may lose its sparsity. As suggested for Newton's iterations, numerical dropping is suggested for this method too [11]. After $\mathbf{M}$ is computed, the elements close to zero are to be dropped, but it is found that the strategy can spoil the minimization and thus the quality of the preconditioner [7], [11].

The following is the algorithm of Global Minimal Residual Iteration:

```

Choose  $\mathbf{M}_0$ 
For  $n = 1, 2, \dots$ 
     $\mathbf{R}_n = \mathbf{I} - \mathbf{A}\mathbf{M}_{n-1}$ 
     $\alpha_n = \text{tr}(\mathbf{R}_n^T \mathbf{A} \mathbf{R}_n) / \text{tr}[(\mathbf{A} \mathbf{R}_n)^T \mathbf{A} \mathbf{R}_n]$ 
     $\mathbf{M}_n = \mathbf{M}_{n-1} + \alpha_n \mathbf{R}_n$ 
    Apply numerical dropping to  $\mathbf{M}_n$ 

```

Algorithm 2.3 Global Minimal Residual

Numerical dropping can be applied to elements of $\mathbf{M}$ that are close to 0. By limiting the maximum number of elements per column is set, the sparsity of $\mathbf{M}$ can be maintained. However, that may spoil the quality of the preconditioner [11]; hence, this thesis will not employ the strategy.

Two choices for the initial guesses are given: $\mathbf{M}_0 = \alpha \mathbf{I}$ and $\mathbf{M}_0 = \alpha \mathbf{A}^T$. α is chosen to minimize the spectral radius $\rho(\mathbf{I} - \alpha \mathbf{A} \mathbf{M})$, but finding α requires a new approximate

inverse $\mathbf{M}$. Therefore, the thesis chooses α by experimenting with several different numbers. $\mathbf{M}_0 = \alpha \mathbf{I}$ is cheaper to use. However, since $\mathbf{M}_0 = \alpha \mathbf{A}^T$ is often a much better initial guess [11], the thesis uses the transpose of $\mathbf{A}$. Chapter 3 presents the results.

2.4. Other Approximate Inverses

Besides Newton's iteration and the Global Minimal Residual iteration, other methods to compute an approximate inverse have been developed.

Chow and Saad present column-oriented Minimal Residual iteration [11]. The similar methods minimizing the Frobenius norm

$$\|\mathbf{I} - \mathbf{A}\mathbf{M}\|_F^2 = \sum_{i=1}^n \|\mathbf{e}_i - \mathbf{A}\mathbf{m}_i\|_2^2,$$

where $\mathbf{e}_i$ is the i th unit vector and $\mathbf{m}_i$ is the i th column of $\mathbf{M}$, are discussed by Grote and Huckle [17] and Cosgrove, Diaz and Griewank [12]. Kolotilina and Yeremin have proposed a factorized sparse approximated inverse preconditioner of the form $\mathbf{G} = \mathbf{G}_L^T \mathbf{G}_L$, where $\mathbf{G}_L$ is lower triangular [21]. Recently Benzi and Tuma have introduced another method based on inverse triangular factorization, which is similar to the classical ILU-type implicit techniques [5].

2.5 The Conjugate Gradient Method on the Normal Equations

When $\mathbf{A}$ is not a square matrix, the least squares problem

$$\min_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2 \tag{2.13}$$

is often applied to solve the system of equations. Setting the derivative of (2.13) to zero, the normal equation can be obtained, as:

$$\mathbf{A}^T \mathbf{A} \mathbf{x} = \mathbf{A}^T \mathbf{b}. \quad (2.14)$$

The conjugate gradient method can be applied to equation (2.13) [2], [7], [17], [18], [25].

The method is abbreviated as CGNE. An important technique is that $\mathbf{A}^T \mathbf{A}$ does not need to be explicitly computed [5], [25]. Instead, when finding α and β , (1.8) and (1.18) become as

$$\alpha_i = \frac{\mathbf{r}_i^T \mathbf{A}^T \mathbf{A} \mathbf{r}_i}{\mathbf{d}_i^T \mathbf{A}^T \mathbf{A} \mathbf{d}_i} = \frac{(\mathbf{A} \mathbf{r}_i)^T \mathbf{A} \mathbf{r}_i}{(\mathbf{A} \mathbf{d}_i)^T \mathbf{A} \mathbf{d}_i}, \quad (2.15)$$

$$\beta_i = \frac{\mathbf{r}_{i+1}^T \mathbf{A}^T \mathbf{A} \mathbf{r}_{i+1}}{\mathbf{r}_i^T \mathbf{A}^T \mathbf{A} \mathbf{r}_i} = \frac{(\mathbf{A} \mathbf{r}_{i+1})^T \mathbf{A} \mathbf{r}_{i+1}}{(\mathbf{A} \mathbf{r}_i)^T \mathbf{A} \mathbf{r}_i}. \quad (2.16)$$

Formulas (2.15) and (2.16) can be written as quotients of norms. The algorithm [18] is shown below:

$$\begin{aligned} \mathbf{r}_0 &= \mathbf{b} - \mathbf{A} \mathbf{x}_0 \\ \mathbf{d} &= \mathbf{A}^T \mathbf{r}_0 \\ i &= 0, 1, 2, \dots \\ \alpha &= \|\mathbf{A}^T \mathbf{r}_i\|^2 / \|\mathbf{A} \mathbf{d}\|^2 \\ \mathbf{x}_{i+1} &= \mathbf{x}_i + \alpha \mathbf{d} \\ \mathbf{r}_{i+1} &= \mathbf{r}_i - \alpha \mathbf{A} \mathbf{d} \\ \beta &= \|\mathbf{A}^T \mathbf{r}_{i+1}\|^2 / \|\mathbf{A}^T \mathbf{r}_i\|^2 \\ \mathbf{d} &= \mathbf{A}^T \mathbf{r}_{i+1} + \beta \mathbf{d} \end{aligned}$$

Algorithm 2.4 CGNE

The matrix $\mathbf{A}^T \mathbf{A}$ is positive definite, but CGNE becomes inefficient for some problems with a large condition number, and converges very slowly [2], [7]. The results will be compared to those of other preconditioning methods.

The preconditioned Conjugate Gradient method on Normal Equation (PGCNE) has also been developed [4], [6], [7], [22]. The algorithm usually uses an implicit preconditioner.

2.6. The Preconditioned Conjugate Gradient Method of MATLAB

A commercial software MATLAB (version 5)[31] is equipped with the function called **pcg**. This function solves a system of linear equations $\mathbf{Ax} = \mathbf{b}$ by implicitly preconditioned Conjugate Gradient Method. The system of preconditioner $\mathbf{My} = \mathbf{z}$ is solved by a backslash $\backslash$ i.e., $\mathbf{y} = \mathbf{M} \backslash \mathbf{z}$, instead of computing an approximate inverse of $\mathbf{M}$. The results of experimenting with **pcg** are included in Chapter 3.

CHAPTER III

Numerical Results

This chapter employs the preconditioned CG to solve $A\mathbf{x} = \mathbf{b}$. As a possible coefficient matrix A , the following matrices are used: the Hilbert matrix, a diagonally dominant matrix, and several tridiagonal matrices. The Hilbert matrix is often used in interpolation problems. Diagonally dominant matrices appear in the theory of vibrations in a string or beam. Tridiagonal matrices are seen in partial differential equations such as the parabolic, elliptic, and hyperbolic equations.

3.1. The Conjugate Gradient Method on Symmetric and Non-symmetric Matrices

This section will solve $A\mathbf{x} = \mathbf{b}$ by CG with no preconditioning, and will show that CG converges to the exact solution in n steps if the $n \times n$ matrix A is symmetric positive definite but does not if A is non-symmetric.

3.1.1. CG on Symmetric Positive Definite

A 5×5 random matrix and its transpose are multiplied so that the result is a symmetric positive definite matrix. Using the matrix as the coefficient matrix A , $A\mathbf{x} = \mathbf{b}$ will be solved. The right-hand side is chosen so that the exact solution is a vector of 1's. First it will be examined if different initial guesses affect the convergence of CG method. Four different vectors are experimented as the initial guess: zero vector, vector of 0.5's, first column of A , and a random vector. The results of each case are shown in table 3.1,

table 3.2, table 3.3, and table 3.4, respectively. The first column shows the numbers of iterations. The second column shows the corresponding residuals.

Initial Guess	$x_0^T = [0 \ 0 \ 0 \ 0 \ 0]$
Iteration	Residual
1	0.1382
2	0.0175
3	0.0067
4	3.2653e-006
5	7.7072e-007
Final Sol = [1.0000 1.0000 1.0000 1.0000 1.0000]	

Table 3.1 Non-preconditioned CG with $x_0^T = [0 \ 0 \ 0 \ 0 \ 0]$

Initial Guess	$x_0^T = 0.5*[1 \ 1 \ 1 \ 1 \ 1]$
Iteration	Residual
1	0.0691
2	0.0087
3	0.0034
4	1.6327e-006
5	3.8536e-007
Final Sol = [1.0000 1.0000 1.0000 1.0000 1.0000]	

Table 3.2 Non-preconditioned CG with $x_0^T = 0.5*[1 \ 1 \ 1 \ 1 \ 1]$

Initial Guess	$x_0^T = [1\text{st column of } A]$
Iteration	Residual
1	0.4001
2	0.0337
3	0.0108
4	3.2670e-006
5	3.0557e-007
Final Sol = [1.0000 1.0000 1.0000 1.0000 1.0000]	

Table 3.3 Non-preconditioned CG with $x_0^T = [1\text{st column of } A]$

Initial Guess	$x_0^T = [\text{random}]$
Iteration	Residual
1	0.3458
2	0.0453
3	0.0041
4	4.2512e-006
5	5.3419e-007

Final Sol = [1.0000 1.0000 1.0000 1.0000 1.0000]

Table 3.4 Non-preconditioned CG with $x_0^T = [\text{random}]$

For this 5×5 case, non-preconditioned CG converges to the exact solution in 5 iterations as expected. The choice of the initial guess has very little effect on the convergence.

3.1.2. CG on Non-Symmetric Positive Definite

Next, it will be shown that the CG method with no preconditioning is not effective at all if A is not symmetric. Again a random matrix A whose entries are all positive is used as the coefficient matrix, but this time it is not multiplied by its transpose; thus, the matrix is non-symmetric. Three different initial guesses are used: zero vector, [0.5 0.5 0.5 0.5 0.5], and the first column of A . The results are shown in table 3.5, table 3.6, and 3.7, respectively.

Initial Guess	$x_0^T = [0 \ 0 \ 0 \ 0 \ 0]$
Iteration	Residual
50	2.5244e+008

Table 3.5 Non-preconditioned CG on non-symmetric A with $x_0^T = [0 \ 0 \ 0 \ 0 \ 0]$

Initial Guess $\mathbf{x}_0^T = 0.5*[1 \ 1 \ 1 \ 1 \ 1]$

Iteration	Residual
50	1.2622e+008

Table 3.6 Non-preconditioned CG on non-symmetric A with $\mathbf{x}_0^T = 0.5*[1 \ 1 \ 1 \ 1 \ 1]$

Initial Guess $\mathbf{x}_0^T = [\text{1st column of A}]$

Iteration	Residual
50	342.4176

Table 3.7 Non-preconditioned CG on non-symmetric A with $\mathbf{x}_0^T = [\text{1st column of A}]$

None of the above cases converge, which suggests that CG is not effective for non-symmetric matrices if CG is not preconditioned. The following sections will examine how the preconditioned CG can improve the convergence.

3.2 Newton's Method

3.2.1. Comparison of Acceleration Parameters for Norm-Scaled Newton's Method

The algorithm of Accelerated Newton's method uses acceleration parameter β . This section compares two acceleration parameters: $1/\max(\|A\|_1, \|A\|_\infty)$ and $1/\|A\|_2$. Table 3.8 shows flops, the number of iterations, three norms, $\|I - A\tilde{A}^{-1}\|_1$, $\|I - A\tilde{A}^{-1}\|_\infty$, and $\|I - A\tilde{A}^{-1}\|_2$, where $\tilde{A}^{-1}$ is an approximated inverse for each β . A is a 30×30 random matrix.

	$1/\max(\ A\ _1, \ A\ _\infty)$	$1/\ A\ _2$
Flops to Compute Norm	900+900	98450
Iterations	22	22
$\ I - A\tilde{A}^{-1}\ _1$	3.5909e-014	1.8124e-014
$\ I - A\tilde{A}^{-1}\ _\infty$	1.9609e-014	1.5363e-014
$\ I - A\tilde{A}^{-1}\ _2$	1.2754e-014	6.8204e-015

Table 3.8 Comparison of Acceleration Parameters

With this random matrix, the numbers of iterations for both acceleration parameters are identical. Also, all the norms yield similar results. However, since computing $\|A\|_2$ requires more flops than computing $\|A\|_1$ and $\|A\|_\infty$, $1/\max(\|A\|_1, \|A\|_\infty)$ will be used for β in this thesis.

Now, an approximate inverse of A will be computed by three variations of Newton's methods - non-scaled, scaled by singular values, and scaled by norms. These methods are applied to a variety of matrices, e.g., the Hilbert matrix, a random matrix with diagonal dominance, and four tridiagonal matrices. All of these matrices are 30×30 .

Also, $A\mathbf{x} = \mathbf{b}$ will be solved by the Conjugate Gradient method preconditioned by the above Newton's methods. The following abbreviations are used: CG with non-scaled Newton's method as CGNN, CG with the singular-value-scaled Newton's method as CGN1, and CG with the norm-scaled Newton's method as CGN2. The system will be solved by the non-preconditioned Conjugate Gradient method (NPCG) as well. The right-hand side vector $\mathbf{b}$ of $A\mathbf{x} = \mathbf{b}$ is set so that the exact solution is the vector of 1's. The zero vector is used as the initial guess for CG so that $\mathbf{r}_0 = \mathbf{b} - A\mathbf{x}_0 = \mathbf{b}$. The stopping criterion for CG is set to $\|\mathbf{x}_n - \mathbf{x}_{n+1}\| < 1.0 \times 10^{-7}$.

Two tables are presented for each matrix. The first table shows the stopping criterion, the numbers of Newton's iterations, the numbers of Newton's flops, the 1-norm, the infinite norm, and the Euclidean norm of $I - A\tilde{A}^{-1}$, and the trace of $A\tilde{A}^{-1}$, which also examines the accuracy.

The second table contains the numbers of CG iterations, the error norm $\|\mathbf{x} - \tilde{\mathbf{x}}\|_2$, where $\tilde{\mathbf{x}}$ is an approximate solution, the residual norm $\|A\tilde{\mathbf{x}} - \mathbf{b}\|_2$, and the total flops. The maximum number of iterations is set to 100.

3.2.2. The Hilbert Matrix

An approximate inverse $\tilde{\mathbf{A}}^{-1}$ for the Hilbert matrix is computed by three versions of Newton's methods. First the stopping criterion is set to $\|\mathbf{M}_{n+1} - \mathbf{M}_n\| < 0.1$, where $\mathbf{M}_n$ converges to $\tilde{\mathbf{A}}^{-1}$. Identification for each method is as follows: method I for non-scaled Newton's method, method II for the scaled by singular values, and method III for the norm-scaled method. Table 3.9 contains the numerical results.

In the non-scaled Newton's method, because the iteration is stopped only after 1 iteration, the criterion is changed to $\|\mathbf{M}_{n+1} - \mathbf{M}_n\| < 0.01$ to find out if the change would make any difference. The change does not affect the scaled methods.

As expected, the flop counts for Newton's methods are huge. Although the approximate inverses computed are not accurate, $\mathbf{H}\mathbf{x} = \mathbf{b}$, where $\mathbf{H}$ is a Hilbert matrix, will be solved by Newton's method.

Table 3.10 shows the results of CG with Newton's methods. NPCG makes the smallest error, and the number of the iterations is not large. CGNN yields the similar error and residual to the NPCG, but the total flops are much larger. CGN1 converges in one iteration and the residual is better than the former two methods, but due to the number of iterations for Newton's method, the total flops are much larger. In CGN2, the number of iterations for Newton's method is almost twice as large as in CGN1. The error is also large, though the best residual is yielded. NPCG is the best method for the Hilbert matrix.

	I. Non-Scaled Newton's		II. Scaled by Singular Values	III. Scaled by Norms
Stopping Criterion	$\ M_{n+1} - M_n\ _1 < 0.1$	$\ M_{n+1} - M_n\ _1 < 0.01$	$\ M_{n+1} - M_n\ _1 < 0.1$	$\ M_{n+1} - M_n\ _1 < 0.1$
Newton's Iterations	1	71	36	70
Newton's flops	336425	24005131	12210578	23676497
$\ I - A\tilde{A}^{-1}\ _1$	1.6738	2.2801	56.6757	2.2874
$\ I - A\tilde{A}^{-1}\ _\infty$	1.6738	2.2801	56.6757	2.2874
$\ I - A\tilde{A}^{-1}\ _2$	1.0000	1.0000	25.7514	1.0000
$\text{Trace}(A\tilde{A}^{-1})$	0.4755	11.1251	11.0314	11.0646

Table 3.9 Approximate Inverse of Hilbert Matrix by Newton's Methods

The stopping criterion for Non-scaled Newton's is changed to 0.01.

Methods	Non preconditioned CG (NPCG)	CG with non- scaled Newton's (CGNN)	CG with scaled Newton's 1 (CGN1)	CG with scaled Newton's 2 (CGN2)
CG iterations	9	14	1	4
$\ x - \tilde{x}\ _2$	0.0061	0.0066	0.0862	1.2240
$\ A\tilde{x} - b\ _2$	2.7040e-008	1.0789e-008	6.4197e-009	1.4640e-010
Total flops	38177	420062	12217689	23701614

Table 3.10 CG with Newton's Methods on Hilbert Matrix

The stopping criterion for preconditioners is set to 0.1.

3.2.3. A Diagonally Dominant Matrix

A random matrix with entries less than 1.0 is generated by MATLAB. Adding 1's to the main diagonal created a diagonally dominant matrix. This is done by adding the identity matrix to the random matrix. The numerical results are presented in the next page. Again two different stopping criteria, 0.1 and 0.01, are experimented.

Table 3.11 shows all three methods - non-scaled Newton's method (method I), the scaled by singular values (method II), and the norm-scaled method (method III) - converge, but method II converges fastest. Again, the change in the stopping criterion affects the convergence of method I, whose numerical results with $\|M_{n+1} - M_n\| < 0.01$ are almost identical to those of method III with $\|M_{n+1} - M_n\| < 0.1$. This time method III is also affected by the change in the stopping criterion, but not method II. The norms are improved but one more iteration is required in computing the approximate inverse.

Table 3.12 presents the results of CG with Newton's Methods on the diagonally dominant matrix. NPCG and CGNN fail to converge while both CGN1 and CGN2 successfully converge. The error of CGN2 is smaller than CGN1, but the former is still expensive.

	I. Non-Scaled Newton's		II. Scaled by Values Singular	III. Scaled by Norms	
Stopping Criterion	$\ M_{n+1} - M_n\ < 0.1$	$\ M_{n+1} - M_n\ < 0.01$	$\ M_{n+1} - M_n\ < 0.1$	$\ M_{n+1} - M_n\ < 0.1$	$\ M_{n+1} - M_n\ < 0.01$
Newton's Iterations	1	26	18	26	27
Newton's flops	367884	9256445	6620639	9263692	9618359
$\ I - A\tilde{A}^{-1}\ _1$	1.9501	2.7291e-010	1.7616e-008	2.7298e-010	1.6032e-013
$\ I - A\tilde{A}^{-1}\ _\infty$	1.9501	2.7299e-010	1.7616e-008	2.7296e-010	9.8255e-014
$\ I - A\tilde{A}^{-1}\ _2$	1.0000	1.6868e-010	9.7345e-009	1.6868e-010	4.5826e-014
Trace($A\tilde{A}^{-1}$)	1.5280	30.0000	30.0000	30.0000	30.0000

Table 3.11 Approximate Inverse of a Diagonally Dominant Matrix by Newton's Methods

The stopping criterion for Non-scaled Newton's and the norm-scaled Newton are changed to 0.01.

Methods	Non preconditioned CG (NPCG)	CG with non- scaled Newton's (CGNN)	CG with scaled Newton's 1 (CGN1)	CG with scaled Newton's 2 (CGN2)
CG iterations	100	100	1	1
$\ x - \tilde{x}\ _2$	1.5702e+004	0.3966	2.7508e-008	4.9504e-012
$\ A\tilde{x} - b\ _2$	2.0187e+004	0.1696	4.6426e-008	8.0784e-008
Total flops	406969	957498	6275287	8934160

Table 3.12 CG with Newton's Methods on a Diagonally Dominant Matrix

The stopping criteria for all methods are set to 0.1.

3.2.4. Tridiagonal Matrix 1

Tridiagonal matrices are often seen in applications such as heat equation and wave equation problems. This and following sections will experiment with 4 tridiagonal matrices.

This matrix, which is identified as tridiagonal matrix 1, contains 1's in its main diagonal, 2's in the upper off-diagonal and -2's in lower off-diagonal, as shown below:

$$\begin{bmatrix} 1 & 2 & 0 & \dots & 0 \\ -2 & 1 & 2 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -2 & 1 & 2 \\ 0 & \dots & 0 & -2 & 1 \end{bmatrix}$$

Table 3.13 on the following page shows the numerical results. All Newton's methods converge, but method II is the best. The change in the stopping criterion affects all the convergence. The results of the method I with $\|\mathbf{M}_{n+1} - \mathbf{M}_n\| < 0.01$ and those of method III with $\|\mathbf{M}_{n+1} - \mathbf{M}_n\| < 0.1$ are almost identical.

Table 3.14 shows that NPCG fails to converge. The other three methods converge. The approximate inverse by non-scaled Newton's method (method I) is not as good as the ones by Newton's methods II and III. Besides, CGNN takes more CG iterations, but the smallest error and residual are yielded. The approximate inverse by CGN2 is the best and the number of CG iteration is smallest; however, due to Newton's iteration, the method is rather expensive.

	I. Non-Scaled Newton's		II. Scaled by Singular Values		III. Scaled by Norms
Stopping Criterion	$\ M_{n+1} - M_n\ < 0.1$	$\ M_{n+1} - M_n\ < 0.01$	$\ M_{n+1} - M_n\ < 0.1$	$\ M_{n+1} - M_n\ < 0.01$	$\ M_{n+1} - M_n\ < 0.1$
Newton's Iterations	1	8	4	5	8
Newton's flops	342305	2761400	1464576	1819468	2763060
$\ I - A\tilde{A}^{-1}\ _1$	0.9216	2.3487e-005	0.0014	4.9249e-007	2.3487e-005
$\ I - A\tilde{A}^{-1}\ _\infty$	0.9216	2.3487e-005	0.0014	4.9249e-007	2.3487e-005
$\ I - A\tilde{A}^{-1}\ _2$	0.9185	1.8669e-005	6.4436e-004	2.0812e-007	1.8669e-005
Trace($A\tilde{A}^{-1}$)	15.8176	30.0000	29.9985	30.0000	30.0000

Table 3.13 Approximate Inverse of Tridiagonal Matrix1 by Newton's Methods

The stopping criterion for Non-scaled Newton's and the singular-valued-scaled Newton are changed to 0.01.

Methods	Non preconditioned CG (NPCG)	CG with non- scaled Newton's (CGNN)	CG with scaled Newton's 1 (CGN1)	CG with scaled Newton's 2 (CGN2)
CG iterations	100	15	3	2
$\ x - \tilde{x}\ _2$	362.1956	1.1957e-015	6.0394e-004	1.1361e-011
$\ A\tilde{x} - b\ _2$	1.0851e+003	3.1244e-015	1.9347e-010	1.8551e-011
Total flops	406969	431794	1483841	2776473

Table 3.14 CG with Newton's Methods on Tridiagonal Matrix 1

The stopping criteria for all methods are set to 0.1.

3.2.5. Tridiagonal Matrix 2

This tridiagonal matrix consists of 1's in the main diagonal and -2's in the off-diagonals. The numerical results are presented in Table 3.15.

	I. Non-Scaled Newton's		II. Scaled by Singular Values	III. Scaled by Norms
Stopping Criterion	$\ M_{n+1} - M_n\ < 0.1$	$\ M_{n+1} - M_n\ < 0.01$	$\ M_{n+1} - M_n\ < 0.1$	$\ M_{n+1} - M_n\ < 0.1$
Newton's Iterations	1	27	20	27
Newton's flops	352052	9556790	7283794	9566058
$\ I - A\tilde{A}^{-1}\ _1$	1.3312	2.2737e-013	7.4545e-012	2.6468e-013
$\ I - A\tilde{A}^{-1}\ _\infty$	1.3312	1.8741e-013	1.3615e-011	1.9851e-013
$\ I - A\tilde{A}^{-1}\ _2$	1.0000	8.1420e-014	6.5589e-012	8.0527e-014
Trace($A\tilde{A}^{-1}$)	14.3328	30.0000	30.0000	30.0000

Table 3.15 Approximate Inverse of Tridiagonal Matrix2 by Newton's Methods

The stopping criterion for Non-scaled Newton's is changed to 0.01.

Methods	Non preconditioned CG (NPCG)	CG with non- scaled Newton's (CGNN)	CG with scaled Newton's 1 (CGN1)	CG with scaled Newton's 2 (CGN2)
CG iterations	15	15	1	2
$\ x - \tilde{x}\ _2$	2.0582e-010	4.4032e-011	2.5589e-013	1.3265e-011
$\ A\tilde{x} - b\ _2$	5.9921e-010	1.1936e-010	5.1637e-013	5.3121e-013
Total flops	62489	441541	7291335	9573619

Table 3.16 CG with Newton's Methods on Tridiagonal Matrix 2

The stopping criteria for all methods are set to 0.1.

Table 3.15 shows that all three methods converge to the inverse of A , but method II converges fastest. The change in the stopping criterion affects convergence of method I, but not methods II or III. Again, the results of method I with $\|M_{n+1} - M_n\| < 0.01$ and those of method III with $\|M_{n+1} - M_n\| < 0.1$ are similar.

Also, in Table 3.16, it is observed that all the methods converge to the exact solution. Both NPCG and CGNN converge in 15 iterations. CGNN takes more total flops because of the preconditioner. CGN1 and CGN2 converge very fast, but because of their preconditioners, the numbers of total flops are not small.

3.2.6. Tridiagonal Matrix 3

Tridiagonal matrix 3 is constructed by replacing two entries of off-diagonals of tridiagonal matrix 2. Now the matrix is not symmetric as shown below:

$$\begin{bmatrix} 1 & 2 & 0 & \dots & 0 \\ -2 & 1 & -2 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -2 & 1 & -2 \\ 0 & \dots & 0 & 2 & 1 \end{bmatrix}$$

Table 3.17 and Table 3.18 on the next page present the numerical results of Newton's methods and the results from CG with Newton's methods, respectively.

	I. Non-Scaled Newton's		II. Scaled by Singular Values	III. Scaled by Norms
Stopping Criterion	$\ M_{n+1} - M_n\ < 0.1$	$\ M_{n+1} - M_n\ < 0.01$	$\ M_{n+1} - M_n\ < 0.1$	$\ M_{n+1} - M_n\ < 0.1$
Newton's Iterations	1	14	10	14
Newton's flops	347192	4895387	3632892	4894561
$\ I - A\tilde{A}^{-1}\ _1$	1.3312	1.0790e-008	5.2683e-007	1.0790e-008
$\ I - A\tilde{A}^{-1}\ _\infty$	1.3312	1.0790e-008	5.2683e-007	1.0790e-008
$\ I - A\tilde{A}^{-1}\ _2$	0.9977	8.4823e-009	3.3467e-007	8.4823e-009
Trace($A\tilde{A}^{-1}$)	14.4352	30.0000	30.0000	30.0000

Table 3.17 Approximate Inverse of Tridiagonal Matrix 3 by Newton's Methods

The stopping criterion for Non-scaled Newton's is changed to 0.01.

Methods	Non preconditioned CG (NPCG)	CG with non- scaled Newton's (CGNN)	CG with scaled Newton's 1 (CGN1)	CG with scaled Newton's 2 (CGN2)
CG iterations	100	1	1	1
$\ x - \tilde{x}\ _2$	1.8484e+004	1.3867e-015	4.7233e-015	7.9867e-015
$\ A\tilde{x} - b\ _2$	3.1859e+003	4.7416e-015	1.4869e-014	1.5857e-014
Total flops	745813	354753	3640453	4902122

Table 3.18 CG with Newton's Methods on Tridiagonal Matrix 3

The stopping criteria for all methods are set to 0.1.

In Table 3.17, it is observed that the results of method I with the stopping criterion 0.01 are almost identical to those of method III with the stopping criterion 0.01. Method II converges to the inverse faster than the other two methods.

Table 3.18 shows NPCG fails to converge while CGNN, CGN1, and CGN2 converge to the exact solution. The stopping criterion is set to 0.1 for all the methods.

Method I does not yield an accurate approximate inverse (Table 3.17); yet, CGNN converges to the exact solution in one iteration (Table 3.18).

3.2.7. Tridiagonal Matrix 4

This tridiagonal matrix is constructed from tridiagonal matrix 3. The 29th and 30th rows of the matrix are almost identical so that it is close to singular. Table 3.19 and Table 3.20 present the numerical results of Newton's methods and CG with Newton's methods, respectively.

	I. Non-Scaled Newton's		II. Scaled by Singular Values	III. Scaled by Norms
Stopping Criterion	$\ M_{k+1} - M_k\ < 0.1$	$\ M_{k+1} - M_k\ < 0.01$	$\ M_{k+1} - M_k\ < 0.1$	$\ M_{k+1} - M_k\ < 0.1$
Newton's Iterations	1	14	100	14
Newton's flops	357661	4925160	35585987	4925584
$\ I - A\tilde{A}^{-1}\ _1$	1.3420	1.0000	1.4363	1.0000
$\ I - A\tilde{A}^{-1}\ _\infty$	1.3420	1.0000	1.4363	1.0000
$\ I - A\tilde{A}^{-1}\ _2$	1.0000	1.0000	0.4622	1.0000
Trace($A\tilde{A}^{-1}$)	11.6669	29.0000	30.0000	29.0000

Table 3.19 Approximate Inverse of Tridiagonal Matrix 4 by Newton's Methods

Methods	Non preconditioned CG (NPCG)	CG with non- scaled Newton's (CGNN)	CG with scaled Newton's 1 (CGN1)	CG with scaled Newton's 2 (CGN2)
CG iterations	100	1	2	1
$\ x - \tilde{x}\ _2$	1.2668e+004	0.2181	7.3433e-008	0.2039
$\ A\tilde{x} - b\ _2$	2.2159e+003	0.0911	1.5384e-015	1.4092e-009
Total flops	406969	944630	35599400	4933145

Table 3.20 CG with Newton's Methods on Tridiagonal Matrix 4

In Table 3.19 shows the results of the three Newton's methods with the stopping criterion $\|\mathbf{M}_{n+1} - \mathbf{M}_n\| < 0.1$ and non-scaled Newton with $\|\mathbf{M}_{n+1} - \mathbf{M}_n\| < 0.01$. All the methods yield rather large norms. The trace from method I is improved by the change in the stopping criterion. Method II yields the best trace but takes more iteration than the others.

When applied to CG, the stopping criteria for all Newton's methods are set to 0.1. CGNN and CGN2 yield similar results though the latter yields much smaller residual. The error is smallest for CGN1, but the number of total flops is large.

3.2.8. Summary of Newton's Methods

The stopping criterion does not affect the quality of the convergence of the singular-value-scaled Newton's method and the norm-scaled Newton's method. On the other hand, the convergence of non-scaled Newton's iteration is almost always improved when the stopping criterion is changed to 0.01,

For the Hilbert matrix, Newton's methods do not converge to $\mathbf{A}^{-1}$, but with CG, CGNN, CGN1, and CGN2 converge to the exact solution. CGN2 yields larger residual than the others. For this matrix, NPCG converges to the exact solution.

For the diagonally dominant matrix, all the three methods converge to $\mathbf{A}^{-1}$. CGN1 and CGN2 converge to the exact solution. The latter takes more flops but yields smaller error. NPCG does not converge. The preconditioning is a helpful tool for this matrix.

For tridiagonal matrix 1, Newton's methods converge to A^{-1} . CGNN, CGN1, and CGN2 converge to the exact solution, while NPCG does not.

For tridiagonal matrix 2, which is symmetric, all the Newton's methods yield a good approximate inverse. Like the Hilbert matrix, which is also symmetric, NPCG converges to the exact solution without aid of preconditioning.

For tridiagonal matrix 3, all the preconditioning methods are helpful. All the errors and residuals are small.

For tridiagonal matrix 4, which is close to singular, Newton's methods do not yield an accurate approximate inverse; however, when they are applied to CG, the preconditioned CGNN, CGN1, and CGN2 converge to the exact solution though the errors from CGNN and CGN2 are slightly large.

Table 3.21 is a summary of CG with Newton's methods. The stopping criteria for all Newton's methods are set to 0.1. The symbols represent the following cases:

○ : CG converges; ■ : CG does not converge; × : CG diverges.

Matrices	Non-precond. CG	CG with Non-scaled Newton	CG with Scaled Newton 1	CG with Scaled Newton 2
I. Hilbert	○	○	○	■
II. Diagonally Dominant	×	■	○	○
III. Tridiagonal 1	×	○	○	○
IV. Tridiagonal 2	○	○	○	○
V. Tridiagonal 3	×	○	○	○
VI. Tridiagonal 4	×	○	○	○

Table 3.21 Summary of CG with Newton's Methods

CGNN, CGN1, and CGN2 indeed improve the convergence of CG, but when a matrix is symmetric, NPCG may be better than the others in terms of flops.

The preconditioned CG methods are expensive. Particularly, CGN2 always takes more flops than CGN1. Even if the iterations of CG are saved, sometimes the saving is too small to make up the cost of expensive preconditioners. However, in terms of accuracy, CGN1, and CGN2 are good methods. When the approximate inverse is accurate, CG can converge almost instantly and accurately to the exact solution.

3.3. The Global Minimal Residual Method

This section will examine the Global Minimal Residual method (GMR) [11]. Instead of comparing different versions as done in 3.2., this section will compare different scalar α for the initial guess $\mathbf{M}_0 = \alpha \mathbf{A}^T$. First, GMR is applied to 5×5 matrices, and then to 30×30 systems in order to examine if the size of a matrix affects the efficiency of α . Types of matrices are the same as those used in 3.2. In Newton's iterations, α for the initial guess $\alpha \mathbf{M}$ is $1/(\|\mathbf{A}\|_1, \|\mathbf{A}\|_\infty)$ (2.3), which is used for $\mathbf{M}_0 = \alpha \mathbf{A}^T$ in GMR as well. As α , $1/\max(\|\mathbf{A}\|_1, \|\mathbf{A}\|_\infty)$, which is the acceleration parameter for the scaled Newton's method III, is experimented. Also, constants such as $\alpha = 1, 0.5$, and 0.05 are used. The stopping criterion is $\|\mathbf{M}_n - \mathbf{M}_{n+1}\| < 0.1$. The maximum iterations are set to 100 for each case.

Following the experiment with GMR on 30×30 matrices, $\mathbf{Ax} = \mathbf{b}$ is solved by the Conjugate Gradient method with the Global Minimal Residual (CGGMR). The coefficient matrices are 30×30 . The right-hand side vector $\mathbf{b}$ is set so that the exact solution is the vector of 1's. The zero vector is used as the initial guess for CG so that, as shown in Chapter 2, $\mathbf{r}_0 = \mathbf{b} - \mathbf{Ax}_0 = \mathbf{b}$. The stopping criterion for CG is set to $\|\mathbf{x}_n - \mathbf{x}_{n+1}\| < 1.0 \times 10^{-7}$.

For each matrix, three tables are presented. The first table contains the results of a 5×5 matrix. The number of GMR iterations, the norms $\|\mathbf{I} - \mathbf{A}\tilde{\mathbf{A}}^{-1}\|_1$ and $\|\mathbf{I} - \mathbf{A}\tilde{\mathbf{A}}^{-1}\|_2$, and $\text{trace}(\mathbf{A}\tilde{\mathbf{A}}^{-1})$ for each α are presented. The accuracy of the approximate inverse is

examined in the aspect of the trace. The second table contains the results of a 30×30 matrix. The number of GMR iterations, the number of flops for GMR, the norms $\|I - A\tilde{A}^{-1}\|_1$ and $\|I - A\tilde{A}^{-1}\|_2$, and $\text{trace}(A\tilde{A}^{-1})$ for each α are presented. The third table contains the number of CG iterations, the error $\|x - \tilde{x}\|_2$, the residual $\|A\tilde{x} - b\|_2$, and the total flops.

3.3.1. Hilbert Matrix

Table 3.22 shows that the results of the experiment with GMR on the 5×5 Hilbert matrix. After 100 iterations, none of the cases converge. Each shows similar results.

α	$1/(\ A\ _1 \ A\ _\infty)$ = 0.1918	$1/\max(\ A\ _1, \ A\ _\infty)$ = 0.4380	1	0.5	0.05
GMR iterations	100	100	100	100	100
$\ I - A\tilde{A}^{-1}\ _1$	1.4608	1.4125	1.4148	1.4553	1.4436
$\ I - A\tilde{A}^{-1}\ _2$	0.9995	0.9990	0.9991	0.9995	0.9994
$\text{Trace}(A\tilde{A}^{-1})$	2.8070	3.0620	3.0820	2.8757	2.9234

Table 3.22 GMR on 5×5 Hilbert Matrix

Table 3.23 on the next page presents the results of the experiment with GMR on the 30×30 Hilbert matrix. Like the 5×5 Hilbert matrix, none of α 's help GMR converge and similar results are found for each α .

Table 3.24 shows the results of CGGMR for each α . GMR for each α fails to converge to the inverse of A as shown in Table 3.23, but CGGMR converges to the exact solution for each α . Because of 100 iterations of GMR, the total flops are quite large.

α	$1/(\ A\ _1\ A\ _\infty)$ = 0.0627	$1/\max(\ A\ _1, \ A\ _\infty)$ = 0.2503	1	0.5	0.05
GMR Iterations	100	100	100	100	100
GMR flops	43683331	43723916	43699070	43681117	43683124
$\ I - A\tilde{A}^{-1}\ _1$	2.1009	2.1072	2.1128	2.1275	2.1007
$\ I - A\tilde{A}^{-1}\ _2$	1.0000	1.0000	1.0000	1.0000	1.0000
$\text{Trace}(A\tilde{A}^{-1})$	3.9636	4.0135	4.0256	4.1212	3.9711

Table 3.23 GMR on 30×30 Hilbert Matrix

α	$1/(\ A\ _1\ A\ _\infty)$ = 0.0627	$1/\max(\ A\ _1, \ A\ _\infty)$ = 0.2503	1	0.5	0.05
CG iterations	7	7	7	7	7
$\ x - \tilde{x}\ _2$	0.0061	0.0061	0.0061	0.0061	0.0061
$\ A\tilde{x} - b\ _2$	9.8479e-009	9.8470e-008	9.8468e-008	9.8478e-008	9.8478e-008
Total flops	43726004	43766589	43741743	43723790	43725797

Table 3.24 CGGMR on Hilbert Matrix

Though GMR does not converge to the inverse of A, CGGMR converges to the exact solution.

3.3.2. Diagonally Dominant Matrix

This matrix is constructed by adding the identity matrix to a randomly generated matrix. In Table 3.25, it is observed that the numbers of iterations may be too large for the 5×5 matrix, but the residuals are small for each α , and the traces are close to 5. For

this matrix, the choice of α seems not to affect the norms; yet, the constants $\alpha = 1, 0.5$, and 0.05 yield smaller norms than the other choices of α .

α	$1/(\ A\ _1 \ A\ _\infty)$ = 0.0776	$1/\max(\ A\ _1, \ A\ _\infty)$ = 0.2777	1	0.5	0.05
GMR Iterations	7	7	12	8	10
$\ I - A\tilde{A}^{-1}\ _1$	0.1813	0.1421	0.0011	0.0023	0.0026
$\ I - A\tilde{A}^{-1}\ _2$	0.1111	0.1016	0.0011	0.0016	0.0018
$\text{Trace}(A\tilde{A}^{-1})$	5.0800	5.0056	5.0012	4.9996	4.9990

Table 3.25 GMR on 5x5 Diagonally Dominant Matrix

α	$1/(\ A\ _1 \ A\ _\infty)$ = 0.025	$1/\max(\ A\ _1, \ A\ _\infty)$ = 0.0497	1	0.5	0.05
GMR Iterations	60	58	1	60	58
GMR flops	27502619	26468092	371926	27544204	26454749
$\ I - A\tilde{A}^{-1}\ _1$	2.1938	1.9226	27.1727	2.0255	1.9174
$\ I - A\tilde{A}^{-1}\ _2$	0.9422	0.9217	13.1063	0.8222	0.9210
$\text{Trace}(A\tilde{A}^{-1})$	24.2531	27.0179	167.7657	25.4714	27.0519

Table 3.26 GMR on 30x30 Diagonally Dominant Matrix

α	$1/(\ A\ _1 \ A\ _\infty)$ = 0.0024	$1/\max(\ A\ _1, \ A\ _\infty)$ = 0.0484	1	0.5	0.05
CG iterations	5	5	100	8	5
$\ x - \tilde{x}\ _2$	1.5210e-009	3.8085e-008	6.3913	7.4216e-009	4.5328e-008
$\ A\tilde{x} - b\ _2$	3.5975e-009	4.4700e-008	97.1768	1.5666e-008	4.8607e-008
Total flops	27533588	26498788	958895	27592729	26485718

Table 3.27 CGGMR on 30x30 Diagonally Dominant Matrix

Though GMR does not converge to the inverse of A , CGGMR converges to the exact solution except for $\alpha = 1$.

Table 3.25 on the previous page shows that large numbers of GMR iterations except for $\alpha = 1$, and rather large norms. GMR is seemingly a poor preconditioner. However, Table 3.26 shows that CGGMR converges to the exact solution with each α except for $\alpha = 1$. The total flops are huge.

3.3.3. Tridiagonal Matrix 1

This matrix contains 1 in the main diagonal, 2 in the upper sub-diagonal, and -2 in the lower off-diagonal as shown in Section 3.2.4.

Table 3.28 shows the results of experiment with GMR on the 5×5 matrix. For $1/(\|A\|_1, \|A\|_\infty)$, 0.5, and 0.05, the traces are close to 5, but the norms are not small enough. Comparing $1/(\|A\|_1, \|A\|_\infty) = 0.04$ and 0.05, similar results are observed. On the other hand, similar norms are found for $1/\max(\|A\|_1, \|A\|_\infty) = 0.2$ and 0.5, but the numbers of iterations differ largely.

α	$1/(\ A\ _1, \ A\ _\infty)$ = 0.0400	$1/\max(\ A\ _1, \ A\ _\infty)$ = 0.200	1	0.5	0.05
GMR Iterations	3	8	1	38	4
$\ I - A\tilde{A}^{-1}\ _1$	1.1049	1.9208	18.6230	1.5250	0.9594
$\ I - A\tilde{A}^{-1}\ _2$	0.5777	1.1645	11.5313	1.2030	0.4866
Trace($A\tilde{A}^{-1}$)	4.3787	2.8204	34.3770	4.9829	5.4789

Table 3.28 GMR on 5×5 Tridiagonal Matrix 1

Table 3.29 shows the results of GMR for each α . Like the above diagonal dominant, when $\alpha = 1$, GMR fails to converge. The iteration stops after 1 step. The trace is the closest to 30 when $\alpha = 0.5$, but the number of iterations is large. When $\alpha = 0.04$ and 0.05, the numbers of iterations are small and the Euclidean norms are less than 1.

α	$1/(\ A\ _1 \ A\ _\infty)$ = 0.0400	$1/\max(\ A\ _1, \ A\ _\infty)$ = 0.2	1	0.5	0.05
GMR Iterations	3	17	1	53	3
GMR flops	1341181	7794605	358669	24189737	1338611
$\ I - A\tilde{A}^{-1}\ _1$	1.3760	2.2190	19.4085	2.8962	1.2521
$\ I - A\tilde{A}^{-1}\ _2$	0.5810	1.4259	15.3773	1.5025	0.5411
Trace($A\tilde{A}^{-1}$)	26.2111	26.0733	245.5255	30.7501	28.7709

Table 3.29 GMR on 30×30 Tridiagonal Matrix 1

α	$1/(\ A\ _1 \ A\ _\infty)$ = 0.0400	$1/\max(\ A\ _1, \ A\ _\infty)$ = 0.2000	1	0.5	0.05
CG iterations	100	100	100	100	100
$\ x - \tilde{x}\ _2$	1.0185e+003	222.4546	0.2960	0.5728	1.1480e+003
$\ A\tilde{x} - b\ _2$	2.1690e+003	806.0568	0.3897	2.0358	2.2967e+003
Total flops	1928150	8381574	945638	24776706	1925580

Table 3.30 CGGMR on Tridiagonal Matrix 1

In Table 3.30, it is observed that none of CGGMR converge. When $\alpha = 1$, GMR yields larger norms and trace than other choices of α (Table 3.29), but CGGMR yields the smallest errors and residual.

3.3.4. Tridiagonal Matrix 2

This tridiagonal matrix contains 1's in the main diagonal and -2 on the both off-diagonals so that it is symmetric.

In Table 3.31 for GMR on the 5×5 matrix, we almost all the norms are less than 1. For $\alpha = 0.04$ and 0.05 , the 1-norm is larger than 1, and the traces are smaller than the others. $1/\max(\|A\|_1, \|A\|_\infty) = 0.2$ and 0.5 yield similar norms and traces, but the number of iterations for the former is more than double.

α	$1/(\ A\ _1, \ A\ _\infty)$ = 0.0400	$1/\max(\ A\ _1, \ A\ _\infty)$ = 0.2	1	0.5	0.05
GMR Iterations	1	29	11	12	1
$\ I - A\tilde{A}^{-1}\ _1$	1.2800	0.5644	0.2709	0.5512	1.3481
$\ I - A\tilde{A}^{-1}\ _2$	0.9600	0.4341	0.2177	0.3883	0.9866
$\text{Trace}(A\tilde{A}^{-1})$	1.4800	4.2205	5.2988	4.4204	1.8519

Table 3.31 GMR on 5×5 Tridiagonal Matrix 2

Table 3.32 on the next page presents the results of GMR on the 30×30 tridiagonal matrix 2. When $\alpha = 1$, GMR fails to converge. For $\alpha = 0.04$ and 0.05 , the iteration stops

after one step. The norms are close to 1, but the traces are too small. For $\alpha = 0.2$ and 0.5, the iterations reach the maximum, but the norms and traces are better than the others.

α	$1/(\ A\ _1 \ A\ _\infty)$ = 0.0400	$1/\max(\ A\ _1, \ A\ _\infty)$ = 0.2	1	0.5	0.05
GMR Iterations	1	100	1	100	1
GMR flops	461694	44993199	362899	44996087	456938
$\ I - A\tilde{A}^{-1}\ _1$	1.2800	1.2697	12.8513	1.2694	1.3930
$\ I - A\tilde{A}^{-1}\ _2$	1.0109	0.9996	12.6215	0.9996	1.0147
$\text{Trace}(A\tilde{A}^{-1})$	10.4380	27.0048	121.3126	27.4851	13.4612

Table 3.32 GMR on 30×30 Tridiagonal Matrix 2

α	$1/(\ A\ _1 \ A\ _\infty)$ = 0.0400	$1/\max(\ A\ _1, \ A\ _\infty)$ = 0.2000	1	0.5	0.05
CG iterations	16	6	17	6	18
$\ x - \tilde{x}\ _2$	1.9628e-009	2.1745e-008	2.7195e-010	2.9069e-016	1.7358e-009
$\ A\tilde{x} - b\ _2$	9.2103e-009	4.4429e-008	2.3131e-009	5.9191e-012	8.4466e-009
Total flops	557035	45030020	464092	45032908	563983

Table 3.33 CGGMR on 30×30 Tridiagonal Matrix 2

Although the norms in Table 3.32 are GMR large, in Table 3.33 it is seen that CGGMR converges with each α . Particularly, when $\alpha = 1$, GMR yields large norms and trace, but CGGMR converges nicely with less flops.

3.3.5. Tridiagonal Matrix 3

Like tridiagonal matrix 2, tridiagonal matrix 3 consists of 1's in the main diagonal and -2's in both off-diagonals. However, the boundary condition is created by replacing two elements of the off-diagonals by 2. The matrix is shown as below:

$$\begin{bmatrix} 1 & 2 & 0 & \dots & 0 \\ -2 & 1 & -2 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -2 & 1 & -2 \\ 0 & \dots & 0 & 2 & 1 \end{bmatrix}.$$

Table 3.34 shows the results of GMR on the 5×5 tridiagonal matrix 2. GMR fails to converge to the inverse of A.

α	$1/(\ A\ _1 \ A\ _\infty)$ = 0.0400	$1/\max(\ A\ _1, \ A\ _\infty)$ = 0.2000	1	0.5	0.05
GMR Iterations	1	2	1	2	1
$\ I - A\tilde{A}^{-1}\ _1$	1.4149	2.0225	13.6471	5.9546	1.4468
$\ I - A\tilde{A}^{-1}\ _2$	1.0359	1.3054	10.1767	4.5033	1.0204
$\text{Trace}(A\tilde{A}^{-1})$	1.6374	5.5987	21.9412	11.6558	1.9401

Table 3.34 GMR on 5×5 Tridiagonal Matrix 3

Table 3.35 and Table 3.36 on the next page presents the results of GMR and CGGMR on the 30×30 tridiagonal matrix 3, respectively. The norms yielded by GMR are not as large as those of tridiagonal matrix 2. However, unlike tridiagonal matrix 2, CGGMR does not converge to the exact solution for any α .

α	$1/(\ A\ _1 \ A\ _\infty)$ = 0.0204	$1/\max(\ A\ _1, \ A\ _\infty)$ = 0.1429	1	0.5	0.05
GMR Iterations	2	3	8	5	2
GMR flops	903682	1366061	3641230	2279247	908962
$\ I - A\tilde{A}^{-1}\ _1$	1.3823	1.7962	2.1093	1.9825	1.4025
$\ I - A\tilde{A}^{-1}\ _2$	1.2072	1.0966	1.3987	1.2523	1.0997
Trace($A\tilde{A}^{-1}$)	17.5515	21.7576	23.2919	20.6497	17.4717

Table 3.35 GMR on 30×30 Tridiagonal Matrix 3

α	$1/(\ A\ _1 \ A\ _\infty)$ = 0.0204	$1/\max(\ A\ _1, \ A\ _\infty)$ = 0.1429	1	0.5	0.05
CG iterations	100	100	100	100	100
$\ x - \tilde{x}\ _2$	94.7870	17.4961	34.6174	20.6497	1.9054e+004
$\ A\tilde{x} - b\ _2$	369.4083	24.3580	34.4525	17.2017	1.1774e+004
Total flops	1490651	1953030	4228199	2866216	1495931

Table 3.36 CGGMR on 30×30 Tridiagonal Matrix 3

3.3.6. Tridiagonal Matrix 4

This tridiagonal matrix consists of 3's in the main diagonal, 2's in the upper off-diagonal, and -2's in the lower off-diagonal. The matrix is diagonally dominant but not symmetric.

Table 3.37 shows the results of GMR on the 5×5 matrix. When $\alpha = 1$, the iteration stops after 1 iteration. For the other α 's, the norms are less than 1, and the traces are closed to 5.

α	$1/(\ A\ _1 \ A\ _\infty)$ = 0.0204	$1/\max(\ A\ _1, \ A\ _\infty)$ = 0.1429	1	0.5	0.05
GMR Iterations	2	5	1	11	5
$\ I - A\tilde{A}^{-1}\ _1$	0.6419	0.8716	25.3091	0.6815	0.1702
$\ I - A\tilde{A}^{-1}\ _2$	0.3638	0.4953	15.2381	0.4428	0.1122
$\text{Trace}(A\tilde{A}^{-1})$	5.1825	4.0599	41.6545	4.7511	5.0403

Table 3.37 GMR on 5×5 Tridiagonal Matrix 4

Table 3.38 on the next page presents the results of GMR on the 30×30 matrix. When $\alpha = 1$, GMR fails to converge. For other values of α , small norms and traces close to 30 are yielded. Particularly, when $\alpha = 0.05$, the number of iterations is also small.

Table 3.39 shows that in this diagonally dominant system, CGGMR converges to the exact solution with almost all α 's except for $\alpha = 1$.

α	$1/(\ A\ _1 \ A\ _\infty)$ = 0.0204	$1/\max(\ A\ _1, \ A\ _\infty)$ = 0.1429	1	0.5	0.05
GMR Iterations	11	7	1	24	8
GMR flops	5000370	7695177	359246	10911105	3635126
$\ I - A\tilde{A}^{-1}\ _1$	0.0942	0.7447	27.5557	0.0926	0.0877
$\ I - A\tilde{A}^{-1}\ _2$	0.0519	0.5399	19.3092	0.0542	0.0565
Trace($A\tilde{A}^{-1}$)	30.2295	30.1144	292.2123	29.7657	30.2989

Table 3.38 GMR on 30×30 Tridiagonal Matrix 4

α	$1/(\ A\ _1 \ A\ _\infty)$ = 0.0204	$1/\max(\ A\ _1, \ A\ _\infty)$ = 0.1429	1	0.5	0.05
CG iterations	8	10	100	8	5
$\ x - \tilde{x}\ _2$	1.2198e-008	1.2700e-008	0.2691	1.4373e-008	1.9168e-008
$\ A\tilde{x} - b\ _2$	5.7798e-008	6.1884e-008	1.0093	7.0980e-008	8.5925e-008
Total flops	5048895	3723959	946215	10959630	1474653

Table 3.39 CGGMR on 30×30 Tridiagonal Matrix 4

3.3.7. Summary of GMR

For both the smaller and larger Hilbert matrices, GMR yields similar norms for each α . Though GMR fails to converge to the inverse of A , CGGMR converges to the exact solution for each α .

GMR on the smaller diagonally dominant matrix converges for each α . On the other hand, the larger diagonally dominant matrix yields large norms. When $\alpha = 1$, the contrast is significant. Though GMR fails to converge for the 30×30 matrix, CGGMR successfully converges to the exact solution for each α .

For tridiagonal matrix 1, GMR yields similar results for both 5×5 and 30×30 matrices. When $\alpha = 1$, both yield larger norms and traces. When $\alpha = 1/(\|A\|_1 \|A\|_\infty)$ and 0.5, the convergence is better than other choices of α . CGGMR fails to converge for any α .

For tridiagonal matrix 2, both sizes do not give an accurate approximate inverse when $\alpha = 1/(\|A\|_1 \|A\|_\infty)$ and 0.05. For $\alpha = 1$, GMR on the 5×5 matrix yields the smaller norms than the other choices of α , but on the 30×30 matrix, the largest norms are yielded. CGGMR converges to the exact solution for each α .

For tridiagonal matrix 3, whose elements are similar to those of tridiagonal matrix 2 except 2 elements in the top and bottom rows; yet, unlike for tridiagonal matrix 2, GMR does not converge for any α . CGGMR also fails to converge for any α .

GMR and CGGMR successfully converge on tridiagonal matrix 4, which is a diagonally dominant, for almost all choices of α except for $\alpha = 1$.

When $\alpha = 1$, almost always GMR fails to converge except for the diagonally dominant matrix and tridiagonal matrix 2. GMR seemingly converges well for diagonal dominant matrices. The convergence of GMR is very sensitive to the initial guess. There may exist other choices for α , but GMR seems to be less efficient than Newton's iterations.

Table 3.40 is the summary of CGGMR for each α . The symbols represent as follows:

○ : CG converges; ■ : CG does not converge; × : CG diverges.

Matrices	$1/(\ A\ _1\ A\ _\infty)$	$1/\max(\ A\ _1, \ A\ _\infty)$	1	0.5	0.05
I Hilbert	○	○	○	○	○
II Diagonally Dominant	○	○	■	○	○
III Tridiagonal 1	×	■	■	■	×
IV Tridiagonal 2	○	○	○	○	○
V Tridiagonal 3	×	■	■	■	×
VI Tridiagonal 4	○	○	■	○	○

Table 3.40 CGGMR Summary

CGGMR converges on the Hilbert matrix and tridiagonal matrix 2, which are symmetric, although GMR shows poor results. For the diagonally dominant matrix and

tridiagonal matrix 4, which is also diagonally dominant, CGGMR converges with fewer iterations than for the Hilbert matrix and tridiagonal matrix 2. GMR seems to be favorable to diagonally dominant matrices.

If GMR is favorable to a matrix, CGGMR converges with almost any α although the total flops may show a large difference. However, CGGMR is less stable than CG with Newton's iterations.

Since constants such as 0.5 and 0.05 do not require any computation, they will be used for α to experiment with larger matrices in the next section.

3.4. Preconditioned Conjugate Gradient Method on Large Matrices

This section will solve a large system $\mathbf{Ax} = \mathbf{b}$. The coefficient matrices are the Hilbert matrix and three sparse matrices. The size of all matrices is 128×128 . The right-hand side vector $\mathbf{b}$ is set so that the exact solution is the vector of 1's. The zero vector is used as the initial guess so that $\mathbf{r}_0 = \mathbf{b} - \mathbf{Ax}_0 = \mathbf{b}$. Methods applied are NPCG, CGNN, CGN1, CGN2, CGGMR with $\alpha = 0.05$, CGGMR with $\alpha = 0.5$, the MATLAB function `pcg`, and CGNE.

The first column of the tables represents the following comparison items:

- 1) Iterations for preconditioner;
- 2) The one norm $\|\mathbf{I} - \mathbf{A}\tilde{\mathbf{A}}^{-1}\|_1$;
- 3) The Euclidean norm $\|\mathbf{I} - \mathbf{A}\tilde{\mathbf{A}}^{-1}\|_2$;
- 4) The trace $(\mathbf{A}\tilde{\mathbf{A}}^{-1})$;
- 5) The spectral condition number κ_2 before preconditioned;
- 6) The spectral condition number κ_2 after preconditioned;
- 7) $\sigma = (\sqrt{\kappa_2} - 1) / (\sqrt{\kappa_2} + 1)$ as in (1.19);
- 8) CG iterations
- 9) The error $\|\mathbf{x} - \tilde{\mathbf{x}}\|_2$;
- 10) The energy norm $\|\mathbf{x} - \tilde{\mathbf{x}}\|_{\mathbf{A}}$;
- 11) The residual $\|\mathbf{A}\tilde{\mathbf{x}} - \mathbf{b}\|_2$;
- 12) Total flops.

Also, for CGN1, CGN2, CGGMR with $\alpha = 0.5$, and CGGMR with $\alpha = 0.05$, the graphs of the eigenvalues of each original and its preconditioned matrices are presented. In the graphs, '*' represents the eigenvalues of the original matrices while 'o' represents the eigenvalues of the preconditioned matrices.

3.4.1. Hilbert Matrix

	Non-precond. CG NPCG	Non-scaled Newton CGNN	Scaled Newton 1 CGN1	Scaled Newton 2 CGN2
Precond Iteration		1	34	70
$\ I - A\tilde{A}^{-1}\ _1$		1.7945	92.8999	1.4146
$\ I - A\tilde{A}^{-1}\ _2$		1.0000	18.1259	1.0000
$\text{Trace}(A\tilde{A}^{-1})$		0.3603	12.4576	14.6972
κ_2 before precondition	1.4921e+16	1.4921e+16	1.4921e+16	1.4921e+16
κ_2 after precondition		9.9145+18	3.9028e+09	1.3319e+16
σ	1.0000	1.0000	1.0000	1.0000
CG Iterations	14	200	6	3
$\ x - \tilde{x}\ _2$	0.0126	0.0148	0.1167	12.2639
$\ x - \tilde{x}\ _A$	1.8267e-05	2.1918e-05	2.0605e-06	5.7490e-10
$\ A\tilde{x} - b\ _2$	3.0484e-08	3.5834e-08	6.2120e-08	8.5049e-08
Total flops	976795	26377777	832415342	1.7308e+09

	CGGMR $\alpha = 0.5$	CGGMR $\alpha = 0.05$	pcg	CGNE
Precond Iteration	200	200		
$\ I - A\tilde{A}^{-1}\ _1$	2.4800	2.4542		
$\ I - A\tilde{A}^{-1}\ _2$	1.0000	1.0000		
$\text{Trace}(A\tilde{A}^{-1})$	5.5054	5.3556		
κ_2 before precondition	1.4921e+16	1.4921e+16	1.4921e+16	1.4921e+16
κ_2 after precondition	8.2694e+15	6.6589e+15		
σ	1.000	1.000		1.0000
CG Iterations	8	8		12
$\ x - \tilde{x}\ _2$	0.0126	0.0126	4.9917e+03	0.0862
$\ x - \tilde{x}\ _A$	1.8216e-05	1.8222e-05	1.4749e-11	5.6802e-07
$\ A\tilde{x} - b\ _2$	2.8836e-08	2.8846e-08	1.1337e-13	1.4519e-09
Total flops	6.6230e+09	6.5201e+09	1626032	2170391

Table 3.41 CG on 128×128 Hilbert Matrix

The blank cells are items that are not available or not interesting to compute.

In Table 3.41 on the previous page, it is observed that all of the methods yield small residuals, but larger errors are observed for CGN2 and pcg. Though the total flops are much larger, the errors and residuals for CGGMR with $\alpha = 0.5$ and CGGMR with $\alpha = 0.05$ are similar to those for NPCG. The results of CGNN in the above table are obtained by the stopping criterion $\|\mathbf{M}_n - \mathbf{M}_{n+1}\| < 0.1$. When the setting is changed to 0.01, unlike the earlier examples in Section 3.2, CGNN does not improve. Though it is not in the table, the error grows from 0.0148 to 31.3021. Figures 3.1 and 3.2 are the plots of the eigenvalues of the 128×128 Hilbert matrix before and after being processed by CGNN with 0.1 and 0.01, respectively. When $\|\mathbf{M}_n - \mathbf{M}_{n+1}\| < 0.01$, the eigenvalues of the preconditioned matrix are scattered. Figures 3.3, 3.4, 3.5, and 3.6 are the plots for CGN1 and CGN2, CGGMR with $\alpha = 0.5$, and CGGMR with $\alpha = 0.05$, respectively. CGN1 and CGN2 show a similar pattern; however, for the former, the eigenvalues scatter on the real axis, while they are clustered around 0 and 1 for the latter. Both types of CGGMR have an almost identical pattern for eigenvalues. Because the eigenvalues are clustered around 0, κ_2 is large for each case. NPCG, on the other hand, is quite efficient even without preconditioning.

Figure 3.1 CGNN (0.1) on 128×128 Hilbert Matrix

'' represents the eigenvalues of the original matrix while 'o' represents the eigenvalues of the preconditioned matrix. Note that the scattered eigenvalues are clustered about zero.*

Figure 3.2 CGNN(0.01) 128×128 Hilbert Matrix

The eigenvalues of the preconditioned eigenvalues are scattered within the radius of 1.5.

Figure 3.3 CGN1 on 128×128 Hilbert Matrix

'' represents the eigenvalues of the original matrix while 'o' represents the eigenvalues of the preconditioned matrix. The eigenvalues of the preconditioned matrix are clustered around zero.*

Figure 3.4 CGN2 on 128×128 Hilbert Matrix

The eigenvalues of the preconditioned matrix are clustered around the origin and 1, but mostly around zero.

Figure 3.5 CGGMR 0.5 on 128×128 Hilbert Matrix

'' represents the eigenvalues of the original matrix while 'o' represents the eigenvalues of the preconditioned matrix. The eigenvalues of the preconditioned matrix are clustered around zero and a few are around 1.*

Figure 3.6 CGGMR 0.05 on 128×128 Hilbert Matrix

The eigenvalues of the preconditioned matrix are clustered around zero and a few are around 1.

3.4.2. Example of Sparse Matrix 1

This matrix, **A** is from a one dimensional fluid dynamics problem of computing a flow through a converging-diverging nozzle [10]. The equation of this problem called Navier Stokes Equation, is as follows:

$$\frac{\partial \mathbf{Q}}{\partial x} + \frac{\partial \mathbf{F}}{\partial x} = \mathbf{R} ,$$

where **Q** is vector of conserved variables, **F** is a flux vector, and **R** contains the effect of viscosity. The matrix, whose original size is 300×300, has been reshaped by trimming the excess columns and rows so that a PC can handle it. This way, the effects of boundary conditions on the Navier Stokes Equation are lost, but the effect of discretization of the Partial Differential Operators is included in **A**. Figure 3.7 on the following page shows the sparsity pattern of the matrix. It is nonsymmetric, and the number of nonzero elements is 1129. The maximum and minimum elements are 122.9985 and -125.2665, respectively.

Table 3.42 presents the results of all the methods. Only CGN2 and **pcg** converge yielding small errors and residuals. For CGN2, κ_2 is 1.0000; hence, σ is very small, and the instant convergence is achieved. The number of total flops is quite large.

Figure 3.7 Sparsity Pattern of 128×128 Sparse Matrix 1 from Nozzle Flow

The dots represent non-zero elements. The number of non-zero elements is 1129. The matrix is non-symmetric.

	Non-precond CG NPCG	Non-scaled Newton CGNN	Scaled Newton 1 CGN1	Scaled Newton 2 CGN2
Precond Iteration		1	1	29
$\ I - A\tilde{A}^{-1}\ _1$		1.4259	2.7782	0.9980
$\ I - A\tilde{A}^{-1}\ _2$		1.0000	1.0000	0.9980
$\text{Trace}(A\tilde{A}^{-1})$		15.7548	55.1306	128.0000
κ_2 before precondition	145.5680	145.5680	145.5680	145.5680
κ_2 after precondition		7.5643e+06	2.4080e+06	1.0000
σ	0.8469	0.9993	0.9987	9.9939e-12
CG Iterations	200	200	200	1
$\ x - \tilde{x}\ _2$	5.8789e+03	42.8906	26.5131	2.4636e-10
$\ x - \tilde{x}\ _A$	9.0669e+03	181.5604	114.1457	2.0650e-11
$\ A\tilde{x} - b\ _2$	4.7238e+05	2.8725e+03	2.1209e+03	2.5523e-11
Total flops	13524239	44527671	50181463	711885302

	CGGMR $\alpha = 0.5$	CGGMR $\alpha = 0.05$	pcg	CGNE
Precond Iteration	1	1		
$\ I - A\tilde{A}^{-1}\ _1$	8.0823e+04	8.0811e+03		
$\ I - A\tilde{A}^{-1}\ _2$	4.7413e+04	4.7404e+03		
$\text{Trace}(A\tilde{A}^{-1})$	9.8780e+05	9.8783e+04		
κ_2 before precondition	145.5680	145.5680	145.5680	145.5680
κ_2 after precondition	9.4384e+06	9.3068e+06		
σ	0.9993	0.9993		0.8469
CG Iterations	200	200	1	200
$\ x - \tilde{x}\ _2$	11.4409	78.8433	3.9106e-14	10.2837
$\ x - \tilde{x}\ _A$	14.3145	284.1832	2.2293e-28	0.4583
$\ A\tilde{x} - b\ _2$	471.2097	1.0095e+04	1.9951e-13	45.3059
Total flops	46927677	27048059	378331	36520464

Table 3.42 128×128 Sparse Matrix 1

The blank cells are items that are not available or not interesting to compute.

The following pages presents the plots of the eigenvalues. Figures 3.8, and 3.9 are the plots for NPCG and CGN1, respectively. The upper figures show the eigenvalues of the original and preconditioned matrices on the same coordinates. The lower figures show the eigenvalues of the preconditioned matrices on a larger scale. The eigenvalues of preconditioned matrices are clustered around 0 for each case. The eigenvalues of the preconditioned matrices are shown on the enlarged coordinates in the lower part of figures. As discussed in Chapter I, when the eigenvalues are close to 0, κ_2 becomes large and CG does not converge to the exact solution. This is evident in Table 3.42.

Figure 3.10 shows the eigenvalues for CGN2. Similar to the previous two figures, the upper part includes the plots of two sets of eigenvalues, and the lower part shows the eigenvalues of the preconditioned matrix. Unlike the previous two examples, the eigenvalues are clustered around 1, not 0. Hence, κ_2 is 1.0000 and thus, with small σ , CG converges in 1 iteration. Figures 3.11 and 3.12 show the cases when the preconditioning methods fail. The matrix is preconditioned by GMR with $\alpha = 0.5$ and $\alpha = 0.05$. The eigenvalues are scattered in a similar pattern, but when $\alpha = 0.05$, the eigenvalues are scattered further from the origin.

Figure 3.8 CGNN on 128×128 Sparse I from Nozzle Flow

'' represents the eigenvalues of the original matrix while 'o' represents the eigenvalues of the preconditioned matrix. Note that the scattered eigenvalues are clustered about zero.*

This figure shows the distribution of the eigenvalues of the preconditioned matrix on a larger scale. The eigenvalues are clustered around zero and some are scattered on the real axis.

Figure 3.9 CGN1 on 128×128 Sparse I from Nozzle Flow

'' represents the eigenvalues of the original matrix while 'o' represents the eigenvalues of the preconditioned matrix. Note that the scattered eigenvalues are clustered about zero.*

This figure shows the distribution of the eigenvalues of the preconditioned matrix on a larger scale. The eigenvalues are clustered around zero and some are scattered on the real axis.

Figure 3.10 CGN2 on 128×128 Sparse I from Nozzle Flow

'' represents the eigenvalues of the original matrix while 'o' represents the eigenvalues of the preconditioned matrix. Note that the scattered eigenvalues are clustered about 1.*

This figure shows the distribution of the eigenvalues of the preconditioned matrix on a larger scale. The eigenvalues are vertically clustered around 1.

Figure 3.11 CGGMR 0.5 on 128×128 Sparse I from Nozzle Flow

The eigenvalues of the preconditioned matrix are scattered. CGGMR fails to converge.

Figure 3.12 CGGMR 0.05 on 128×128 Sparse I from Nozzle Flow

The eigenvalues of the preconditioned matrix are scattered further from the origin. CGGMR fails to converge.

3.4.3. Example of Sparse Matrix 2

This sparse matrix is from a heat-conduction problem [26]. The problem approximates the steady temperature distribution in a winding with a rectangular cross-section. The equation for the temperature U is given by:

$$\frac{\partial^2 U}{\partial x^2} + \lambda \frac{\partial^2 U}{\partial y^2} = -\frac{q}{K}, \quad (3.2)$$

where K is the thermal conductivity, q is the amount of heat generated per unit area per second, and λ is the eigenvalue. For illustrative purposes, we take $\lambda = 3$ and $q/K = 100$, respectively. Then equation (3.2) becomes

$$\frac{\partial^2 U}{\partial x^2} + 3 \frac{\partial^2 U}{\partial y^2} = -100. \quad (3.3)$$

Let U be the solution of equation (3.3) inside the square $x = 0, 1, y = 0, 1$, satisfying the following boundary conditions:

$$U = 0 \text{ on } x = 1, \quad \frac{\partial U}{\partial y} = -U \text{ on } y = 1,$$

$$\frac{\partial U}{\partial x} = 0 \text{ on } y = 0, \quad \frac{\partial U}{\partial y} = 0 \text{ on } x = 0.$$

A finite difference approximation equation (3.3) is

$$\frac{1}{h^2} (u_{i+1,j} - 2u_{i,j} + u_{i-1,j}) + \frac{3}{h^2} (u_{i,j+1} - 2u_{i,j} + u_{i,j-1}) = -16.$$

With $h = 1/4$,

$$u_{i+1,j} + 3u_{i,j+1} + u_{i-1,j} + 3u_{i,j-1} - 8u_{i,j} = -1, \quad (3.4)$$

where $i = 1, \dots, 20$, and $j = 1, \dots, 20$.

20 mesh points including those on the boundaries are created. Writing equation (3.4) at each mesh point, a system of equations, $\mathbf{Ax} = \mathbf{b}$, is obtained. Each matrix and vector represent the following:

$$\mathbf{A} = \begin{bmatrix} (\mathbf{B} - 1.5\mathbf{I}) & 6\mathbf{I} & & & \\ & 3\mathbf{I} & \mathbf{B} & 3\mathbf{I} & \\ & & 3\mathbf{I} & \mathbf{B} & 3\mathbf{I} \\ & & & \ddots & \ddots & \ddots \\ & & & & 3\mathbf{I} & \mathbf{B} & 3\mathbf{I} \\ & & & & & 6\mathbf{I} & \mathbf{B} \end{bmatrix}, \mathbf{b} = \begin{bmatrix} -1 \\ -1 \\ -1 \\ \vdots \\ -1 \\ -1 \end{bmatrix},$$

$$\mathbf{B} = \begin{bmatrix} -8 & 2 & 0 & 0 \\ 1 & -8 & 1 & 0 \\ 0 & 1 & -8 & 1 \\ 0 & 0 & 1 & -8 \end{bmatrix},$$

and $\mathbf{I}$ is the 4×4 identity matrix.

For a testing purpose, the system is expanded to 128 equations, maintaining the boundary conditions. Also, instead of the vector of -1 , $\mathbf{b}$ is chosen so that the exact solution $\mathbf{x}$ is the vector of 1 . Clearly $\mathbf{A}$ is a non-symmetric and ill-conditioned. Figure 3.13 shows the sparsity of this matrix. It is 128×128 , and the number of non-zeros elements is 568. The maximum and minimum elements are 6 and -9.5 , respectively.

Table 3.43 presents the results of all the methods on this sparse matrix. All the methods except CGGMR with $\alpha = 0.5$ converge to the exact solution. However, NPCG

yields rather large error norms and residuals compared to the others. Also, the errors by CGNE are small but the residual is large. For CGNN, CGN1, and CGN2 $\kappa_2 = 1.0000$; hence, σ is very small, and instant convergence is achieved. The stopping criterion for the preconditioner of CGNN is set to 0.01. The results are identical to those for CGN2.

Figure 3.13 Sparsity Pattern of Sparse Matrix 2

The number of non-zero elements is 568. The dots represent non-zero elements. The matrix is non-symmetric.

	Non-precond. CG NPCG	Non-scaled Newton CGNN	Scaled Newton 2 CGN1	Scaled Newton 2 CGN2
Precond Iteration		18	12	18
$\ I - A\tilde{A}^{-1}\ _1$		6.4222e-08	4.1601e-07	6.4222e-08
$\ I - A\tilde{A}^{-1}\ _2$		3.5729e-08	2.0780e-07	3.5729e-08
$\text{Trace}(A\tilde{A}^{-1})$		128.0000	128.0000	128.0000
κ_2 before precondition	0.0099	0.0099	0.0099	0.0099
κ_2 after precondition		1.0000	1.0000	1.0000
σ	0.8192	8.9323e-09	5.1950	8.9323e-09
CG Iterations	200	1	2	1
$\ x - \tilde{x}\ _2$	0.0020	2.2796e-07	2.6529e-15	2.2796e-07
$\ x - \tilde{x}\ _A$	0.0020	1.0755e-07	2.9227e-15	1.0755e-07
$\ A\tilde{x} - b\ _2$	0.0026	9.8478e-08	8.7918e-15	9.8478e-08
Total flops	13524239	441825012	302129839	441937315

	GMR $\alpha = 0.5$	GMR $\alpha = 0.05$	pcg	CGNE
Precond Iteration	1	3		
$\ I - A\tilde{A}^{-1}\ _1$	45.1690	1.5796		
$\ I - A\tilde{A}^{-1}\ _2$	26.2782	0.9808		
$\text{Trace}(A\tilde{A}^{-1})$	768.0367	114.2824		
κ_2 before precondition	0.0099	0.0099	0.0099	0.0099
κ_2 after precondition	1.0348e+03	28.1096		
σ	0.9387	0.6826		0.8192
CG Iterations	200	161	1	200
$\ x - \tilde{x}\ _2$	10.1553	2.2722e-07	4.0717e-15	7.0915e-15
$\ x - \tilde{x}\ _A$	4.4631	1.1755e-07	4.5722e-15	1.6657e-15
$\ A\tilde{x} - b\ _2$	6.0470	9.5883e-08	1.0805e-14	19.7916
Total flops	46957070	114261359	299875	36520464

Table 3.43 128×128 Sparse Matrix 2

The blank cells are items that are not available or not interesting to compute.

As discussed in Chapter I, when the eigenvalues are close to 1, $\kappa_2 \approx 1$; hence, σ in inequality 1.22 becomes very small, which implies the fast convergence. On the other hand, when the eigenvalues are close to 0 or very large, κ_2 becomes large and the convergence cannot be achieved. The following pages present figures of the eigenvalues distributions. Figures 3.14, and 3.15 are the plots for NPCG and CGN1, respectively. In both figures, the eigenvalues of the non-preconditioned matrix are scattered on the real axis, but the eigenvalues of the preconditioned matrix are clustered around 1. Figure 3.16 shows the eigenvalues of the matrix preconditioned by CGN2. The pattern is very similar to the previous two figures, but Figure 3.16 shows fewer 'o'. That suggests that there are multiple identical eigenvalues. Figure 3.17 is a failed example of CGGMR with $\alpha = 0.5$. The eigenvalues of the preconditioned matrix are scattered in both positive and negative directions on the real axis. Some are scattered in the complex plane. Figure 3.18 is the plot for CGGMR with $\alpha = 0.05$. The eigenvalues of the preconditioned matrix are clustered around 1, but some are close to 0.

Figure 3.14 CGNN on 128×128 Sparse Matrix 2

'' represents the eigenvalues of the original matrix while 'o' represents the eigenvalues of the preconditioned matrix. Note that the scattered eigenvalues are clustered about 1.*

Figure 3.15 CGN1 on 128×128 Sparse Matrix 2

'' represents the eigenvalues of the original matrix while 'o' represents the eigenvalues of the preconditioned matrix. Note that the scattered eigenvalues are clustered about 1.*

Figure 3.16 CGN2 on 128×128 Sparse 2

'' represents the eigenvalues of the original matrix while 'o' represents the eigenvalues of the preconditioned matrix. Note that the scattered eigenvalues are clustered about 1.*

Figure 3.17 CGGMR 0.5 on 128×128 Sparse 2

The eigenvalues of the preconditioned matrix are scattered on the real axis. Some are scattered in the complex plane. CGGMR fails to converge.

Figure 3.18 CGGMR 0.05 on Sparse 2

The eigenvalues of the preconditioned matrix are clustered around 1. Some are close to zero.

3.4.4. Example of Sparse Matrix 3

Slight changes on the boundaries are made in sparse matrix 2 of the last example in order to construct this sparse matrix. A negative sign is given to the element $A(1,2)$, and the element of the bottom right corner of A is changed to 100. This matrix is ill-conditioned with the conditioned number 629.5399, and will be a severe test for the method in this thesis.

Table 3.44 on the next page presents the results of all the methods. CGNN, CGN1, and CGGMR fail to converge to the exact solution. The errors and residual for NPCG are small compared to those of CGNN and CGN1, but NPCG takes 200 iterations. CGN2 and `pcg` converges to the exact solution yielding small energy norms and residuals, but the former takes more flops. The error norms for CGNE are small, but the residuals are large.

Figures of the eigenvalues follow the table. The pattern of the original eigenvalues is slightly different from that of sparse matrix 2; some eigenvalues are scattered on the complex plane and one eigenvalue $\lambda = 100$ is found on the real axis. In Figure 3.19 and 3.20, the eigenvalues of the preconditioned matrix are clustered around zero. In Figure 3.21, the eigenvalues of the preconditioned matrix are clustered around 1. In both Figures 3.22 and 3.23 for CGGMR, the eigenvalues of the preconditioned matrix are scattered on the real axis. When the eigenvalues are clustered around 1, which is seen for CGN2, the method converges. Table 3.44 shows $\kappa_2 = 1.0000$ and small σ . On the other hand, for the

methods that do not converge, the eigenvalues are clustered around 0 and large values of κ_2 are found in the table.

	Non-precond. CG NPCG	Non-scaled Newton CGNN	Scaled Newton 1 CGN1	Scaled Newton 2 CGN2
Precond Iteration		1	1	23
$\ I - A\tilde{A}^{-1}\ _1$		1.0261	1.0941	6.6025e-009
$\ I - A\tilde{A}^{-1}\ _2$		1.0000	1.0000	4.1995e-009
$\text{Trace}(A\tilde{A}^{-1})$		2.9405	8.5681	128.0000
κ_2 before precondition	600.3232	600.3232	600.3232	600.3232
κ_2 after precondition		2.1539e+005	9.8279e+003	1.0000
σ	0.9216	0.9957	0.9800	1.0499e-009
CG Iterations	200	200	200	2
$\ x - \tilde{x}\ _2$	0.0409	3.7530	18.2190	2.8283e-015
$\ x - \tilde{x}\ _A$	0.0251	52.1521	9.1414	2.2995e-015
$\ A\tilde{x} - b\ _2$	0.0390	1.6929	151.1556	5.8210e-015
Total flops	13524239	44562556	49903052	566394365

	GMR $\alpha = 0.50$	GMR $\alpha = 0.05$	pcg	CGNE
Precond Iteration	1	1		
$\ I - A\tilde{A}^{-1}\ _1$	168.2150	15.7433		
$\ I - A\tilde{A}^{-1}\ _2$	148.4192	13.7996		
$\text{Trace}(A\tilde{A}^{-1})$	6.1156e+003	603.0824		
κ_2 before precondition	600.3232	600.3232	600.3232	600.3232
κ_2 after precondition	1.3251e+004	5.9572e+004		
σ	0.9828	0.9918		0.9216
CG Iterations	200	200	1	200
$\ x - \tilde{x}\ _2$	12.1110	14.2889	1.3961e-014	4.6995e-014
$\ x - \tilde{x}\ _A$	9.4549	11.5947	5.5843e-015	1.4007e-013
$\ A\tilde{x} - b\ _2$	151.7513	255.1522	4.4655e-14	1.1017
Total flops	46923537	46933954	299569	36520464

Table 3.44 128×128 Sparse Matrix 3

The blank cells are items that are not available or not interesting to compute.

Figure 3.19 CGNN on 128×128 Sparse 3

'' represents the eigenvalues of the original matrix while 'o' represents the eigenvalues of the preconditioned matrix. Note that the scattered eigenvalues are clustered about zero.*

Figure 3.20 CGN1 on 128×128 Sparse 3

'' represents the eigenvalues of the original matrix while 'o' represents the eigenvalues of the preconditioned matrix. Note that the scattered eigenvalues are clustered about zero.*

Figure 3.21 CGN2 on 128×128 Sparse 3

'' represents the eigenvalues of the original matrix while 'o' represents the eigenvalues of the preconditioned matrix. Note that the scattered eigenvalues are clustered about 1.*

Figure 3.22 CGGMR 0.5 on 128×128 Sparse 3

The eigenvalues of the preconditioned matrix are scattered on the real axis. CGGMR fails to converge.

Figure 3.23 CGGMR 0.05 on 128×128 Sparse 3

The eigenvalues of the preconditioned matrix are scattered on the real axis. CGGMR fails to converge.

CHAPTER IV

Conclusions and Future Work

This thesis ends with conclusions and discussion of future possible work. The thesis has examined how preconditioning can improve the convergence of the Conjugate Gradient method when it is applied to non-symmetric and/or ill-conditioned matrices. The preconditioning methods used are based on an approximate inverse which was computed by Newton's method and the Global Minimal Residual method. These methods are categorized as explicit methods.

Besides the usual Newton's method, two accelerated Newton's methods have been introduced. Newton's method scaled by the singular values [23] is found to be effective, but CGN1 (CG with the Newton's method scaled by the singular values) fails to converge to the exact solution when the method is applied to ill-conditioned matrices whose eigenvalues include complex numbers. Newton's method scaled by the maximum of the 1-norm and the infinite norm is also found to be effective. CGN2 (CG with the Newton's method scaled by the norm) converges to the exact solution for almost all the cases except when the coefficient matrix is close to singular. The accuracy of Newton's method scaled by the norm is better than the non-scaled Newton's and the singular-value-scaled Newton's method. However, the speed of convergence was not particularly improved when compared with usual Newton's method. In order to increase the speed, another accelerating method such as scaling by singular values could be applied to the norm-scaled Newton's.

Also, the thesis has experimented with the Global Minimal Residual method [11] with the different initial guesses and varying the parameter of α . GMR is seemingly less efficient and numerically less stable than Newton's Method. CGGMR converges to the exact solution for diagonally dominant matrices, but it takes more flops than CG with any of Newton's methods. For ill-conditioned matrices, the results from CGGMR can be disastrous. There are more possibilities for the initial guess parameter which may yield better convergence, but it seems that the choice of α may not be so important if GMR converges, though the speed of convergence may be affected.

When preconditioning is successful, the eigenvalues of the preconditioned matrix are clustered together around 1, hence, κ_2 is close to 1 and σ is very small. This results in fast convergence of CG. When the eigenvalues are clustered together around 0, κ_2 is large and σ is close to 1, which implies slow convergence. Studying how the distribution of the eigenvalues of the preconditioned matrix $A\tilde{A}^{-1}$ changes in each iteration may be helpful to understand the nature of the convergence and to improve its speed.

As expected, all the preconditioning methods presented in this thesis are expensive. Since Newton's methods and GMR were developed for sparse matrices, these methods are too expensive for dense matrices. The thesis did not apply the numerical dropping in order to avoid the potential risk of losing the accuracy, but that is the possible approach to maintain the sparsity [5], [11].

We also experimented the MATLAB function `pcg`, which uses implicit preconditioning method. The numerical results are comparable to those from CGN2, but the flops are much less.

Another method experimented is CGNE, which is a popular iterative method for non-symmetric systems. This method was applied to large matrices, but it does not converge to the exact solution for any case. CGNE can be preconditioned by the methods discussed in Chapter 2. For instance, if A is non-symmetric, it is possible to apply an approximate inverse $(AA^T)^{-1}$ computed by CGN2.

Other preconditioning methods are listed in Chapter 2. Compared to them, approximating an inverse by Newton's method may not be so sophisticated. However, based on the results of the numerical experiments, it is possible to conclude that CGN2 has the potential to become a good tool to help the Conjugate Gradient method in solving non-symmetric and ill-conditioned systems. To make CGN2 a better preconditioning tool, a thorough understanding of the theoretical aspect of CGN2 is necessary.

This thesis applied the preconditioned CG to linear systems only. A possible future work is to apply PCG to a nonlinear system and study the convergence behavior. Also it will be interesting to study the distribution of the eigenvalues of any matrix preconditioned by other techniques such as column-oriented Minimal Residual method, triangular factorization, or an incomplete LU decomposition.

APPENDIX

The followings are MATLAB M-files for:

- I. The Conjugate Gradient method
- II. Non-scaled Newton's method
- III. Singular-value-scaled Newton's method
- IV. Norm-scaled Newton's method
- V. Global Minimal Residual
- VI. Conjugate Gradient Method on Normal Equations.

I. The Conjugate Gradient Method

```
function [x,residual]=cg(A,x,b,tol,steps)
%
%This function solves  $Ax=b$  by the Conjugate Gradient Method. A preconditioning
function is called before the iteration.
%Input:      A (coefficient matrix)
%            x (initial guess)
%            b (vector on right hand side of equation)
%            tol (tolerance)
%            steps (maximum iterations)
%Output:     x (solution)
%            residual (norm(b-Ax))
%
rnew=b-A*x;
wold=10;
[Ainv]=newton(A,steps); % Compute approximate inverse by preconditioner
for i=1:steps
    if wold~=0
        rold=rnew;
        z=Ainv*rold;
        z=rold;
        w=rold'*z;
        if i==1
            p=z;
        else
```

```

        B=w/wold;
        p=z+B*p;
    end
    q=A*p;
    s=p'*q;
    if s==0
        a=w/s;
        x=x+a*p;
        residual=norm(b-A*x);
        if residual<=tol
            break
        end
        rnew=rold-a*q;
        wold=w;
    end
end
end
end

```

II. Non-Scaled Newton's Method

```

function [Ainv]=nsnewton(A,steps)
%
%This function computes an approximate inverse of A by non-scaled Newton's method.
%Input:      A (matrix)
%            steps (maximum iterations)
%Output:     Ainv (approximate inverse of A)
%
alpha=1/(norm(A,1)*norm(A,inf));
Mold=alpha*A';
for i=1:steps
    Mnew=Mold*(2*eye(size(A))-A*Mold);
    if norm(eye(size(A))-A*Mnew)>=10
        break
    end
    diff=Mold-Mnew;
    if norm(diff)<=0.1
        break
    end
    Mold=Mnew;
end
Ainv=Xnew;

```

III. Singular-Value Scaled Newton's Method

```
function [Ainv]=snewton1(A,steps)
%
%This function computes an approximate inverse by singular-valued scaled Newton's
  method based on Pan & Schreiber's algorithm [23].
%Input:      A (matrix)
%            steps (maximum iterations)
%Output:     Ainv (approximate inverse of A)
%
B=svd(A);
alpha0=2/(max(B)^2+min(B)^2);
p=alpha0*min(B);
Mold=alpha0*A';
for i=1:steps
    alpha=2/(1+(2-p)*p);
    Mnew=alpha*(2*eye(size(A))-Mold*A)*Mold;
    norm1=norm(eye(size(A))-A*Mnew);
    if norm1>=10
        break
    end
    norm2=norm(Mold-Mnew);
    if norm2<=0.1
        break
    end
    Mold=Mnew;
    p=alpha*(2-p)*p;
end
Ainv=Mnew;
```

IV. Norm-Scaled Newton's Method

```
function [Ainv]=nnewton2(A,steps)
%
%This function computes an approximate inverse by norm-scaled Newton's method.
%Input:      A (matrix)
%            steps (maximum iterations)
%Output:     Ainv (approximate inverse of A)
%
m=max(norm(A,1),norm(A,inf));
AA=(1/m)*A;
alpha=1/(norm(AA,1)*norm(AA,inf));
```

```

Xold=alpha*AA';
for i=1:steps
    Xnew=Xold*(2*eye(size(A))-AA*Xold);
    if norm(eye(size(A))-Xnew*AA)>=10
        break
    end
    diff=Xold-Xnew;
    if norm(diff)<=0.1
        break
    end
    Xold=Xnew;
end
Ainv=(1/m)*Xnew;

```

V. Global Minimal Residual

```

function [Ainv]=GMR(A,steps)
%
%This function computes an approximate inverse by GMR based on Chow & Saad's
algorithm [11].
%Input:      A (matrix)
%            steps (maximum iterations)
%Output:     Ainv (approximate inverse of A)
%
%a=1/(norm(A,1)*norm(A,inf))           %parameter choice for initial guess
%a=1/max(norm(A,1),norm(A,inf))        %  "      "      "      "      "
%a=1                                    %  "      "      "      "      "
%a=0.5                                  %  "      "      "      "      "
a=0.05                                  %  "      "      "      "      "
Mold=a*A';
for i=1:steps
    R=eye(size(A))-A*Mold;
    RR=A*R;
    alph=trace(R'*RR)/trace(RR'*RR);
    Mnew=Mold+alph*R;
    norm1=norm(A*Mnew-eye(size(A)));
    if norm1>=10 %revise:3
        break
    end
    norm2=norm(Mold-Mnew);
    if norm2<=0.1
        break
    end
end

```

```

end
Mold=Mnew;
end
Ainv=Mnew;

```

V. The Conjugate Gradient Method on Normal Equation

```

function [x,residual]=cgne(A,x,b,tol,steps)
%
%This function solves  $Ax=b$  by the Method of Conjugate Gradient applied to the
normal equations [18].
%Input:      A (coefficient matrix)
%            x (initial guess)
%            b (vector on right-hand side of Eq.)
%            tol (tolerance)
%            steps (maximum iterations)
%Output:     x (solution)
%
rold=b-A*x;
d=A'*rold;
xold=x;
for i=1:steps
    s=A*d;
    wold=A'*rold;
    ss=s'*s;
    if (ss~=0)
        ww=wold'*wold;
        a=ww/ss;
        xnew=xold+a*d;
        residual=norm(b-A*x);
        if (residual<=tol)
            break
        end;
        rnew=rold-a*A*d;
        wnew=A'*rnew;
        if (ww~=0)
            b=(wnew'*wnew)/ww;
            d=wnew+b*d;
            xold=xnew;
            rold=rnew;
        else
            break
        end
    end
end

```

```
        end;  
    else  
        break  
    end;  
end;
```

REFERENCES

- [1] Axelsson, O., *Conjugate Gradient Type Methods for Unsymmetric and Inconsistent Systems of Linear Equations*, Linear Algebra and its Applications, 29:1-16, 1980.
- [2] Barrett, R., et al, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*, SIAM, Philadelphia, 1994.
- [3] Ben-Israel, A., and Cohen, D., *On Iterative Computation of Generalized Inverses and Associated Projections*, J. SIAM Numer. Anal., 3:410-419, 1966.
- [4] Benzi, M., Meyer, C.D., and Tuma, M., *A Sparse Approximate Inverse Preconditioner for the Conjugate Gradient Method*, SIAM, J. Sci. Comput., 17:1135-1149, 1996.
- [5] Benzi, M. and Tuma, M., *A Sparse Approximate Inverse Preconditioner for Nonsymmetric Linear Systems*, SIAM, J. Sci. Comput., 19:968-994, 1998.
- [6] Bjorck, A. and Elfving T., *Accelerated Projection Methods for Computing Pseudoinverse Solutions of Systems of Linear Equations*, BIT 19:145-163, 1979.
- [7] Bruaset, A.M., *A Survey of Preconditioned Iterative Methods*, Volume 328 of Pitman Research Notes, Longman Scientific & Technical, 1995.
- [8] Buchanan, J.L. and Turner, P.R., *Numerical Methods and Analysis*, pages 371-388, McGraw-Hill Book Company, 1992.
- [9] Chan, R.H. and Ng, M.K., *Conjugate Gradient Methods for Toeplitz Systems*, SIAM Review, 38:427-482, 1996.
- [10] Cheer, A. and Saleem, M., *Acceleration of Convergence by Shifting the Spectrum of Implicit Finite Difference of Gas Operators Associated with the Equations of Gas Dynamics*, International Journal for Numerical Methods in Fluids, 5:443-462, 1991.
- [11] Chow, E. and Saad Y., *Approximate Inverse Preconditioners for General Sparse Matrices*, Department of Computer Science, and Minnesota Supercomputer Institute University of Minnesota, May 1994.
- [12] Cosgrove, J.D.F., Diaz, J.C. and Griewank, A., *Approximate Inverse Preconditionings for Sparse Linear Systems*, Intern. J. Computer Math., 44:91-110, 1992.

- [13] Drako, N., *Iterative Methods*, PVI, sepwww.stanford.edu/sep/prof/pvi/toc_html, 1997.
- [14] Dubois, P.F., Greenbaum, A., and Rodrigue, G.H., *Approximating the Inverse of a Matrix for Use in Iterative Algorithms on Vector Processors*, *Computing*, 22:257-268, 1979.
- [15] Eicke, B., Louis, A.K., and Plato, R., *The instability of some gradient methods for ill-posed problems*, *Numer. Math.* 58:129-134, 1990.
- [16] Gerlach, J., *Accelerated Convergence in Newton's Method*, *SIAM Review*, 36:272-276, 1994.
- [17] Grote, M. J. and Huckle, T., *Parallel Preconditioning with Sparse Approximate Inverse*, *SIAM J. Sci. Comput.*, 18:838-853, 1997.
- [18] Hanke, M. *Conjugate Gradient Type Methods for ill-posed Problems*, Volume 327 of Pitman Research Notes, Longman Scientific & Technical, 1995.
- [19] Hanke, M. and Nagy, J.G., *Restoration of Atmospherically Blurred Images by Symmetric Indefinite Conjugate Gradient Techniques*, *Inverse Problems*, 12:157-173, 1996.
- [20] Jennings, A., *Influence of the Eigenvalue Spectrum on the Convergence Rate of the Conjugate Gradient Method*, *J. Inst. Math. Appl.*, 20:61-72, 1977.
- [21] Kolotilina, L.Y. and Yereimin, A.Y., *Factorized Sparse Approximate Inverse Preconditioning I. Theory*, *SIAM J. Matrix Anal. Appl.*, 14:45-58, 1993.
- [22] Mitchell, A.R. and Griffiths, D.F., *The Finite Difference Method in Partial Differential Equations*, pages 154-159, John Wiley & Sons, 1987.
- [23] Pan, V. and Schreiber, R., *An Improved Newton Iteration for the Generalized Inverse of a Matrix with Applications*, *SIAM J. Sci. Stat. Comput.*, 12:1109-1130, 1991.
- [24] Saad, Y., *Preconditioning Techniques for Nonsymmetric and Indefinite Linear Systems*, *J. of Comp. and Appl. Math.*, 24:89-105, 1988.
- [25] Shewchuk, J. R., *An Introduction to the Conjugate Gradient Method Without the Agonizing Pain*, CMU-CS-94-125, Carnegie Mellon University, Pittsburgh, PA, 1994.

- [26] Smith, G. D., *Numerical Solution of Partial Differential Equations: Finite Difference Methods*, Oxford Applied Mathematics and Computing Science Series, Oxford Press, 1993.
- [27] Strakos, Z., *On the Real Convergence Rate of the Conjugate Gradient Method*, *Linear Algebra and its Applications*, 154-156:535-549, 1991.
- [28] Van Der Sluis, A. and Van Der Vorst, H.A., *The Rate of Convergence of Conjugate Gradients*, *Numer. Math.*, 48:543-560, 1986.
- [29] Van Der Vorst, H.A., *Conjugate Gradient Type Methods for Nonsymmetric Linear Systems*, In R. Beauwens and P. de Groen, editors, *Iterative Methods in Linear Algebra*, pages 67-76, North-Holland, 1992.
- [30] Vassilenvski, P.S., *Preconditioning Nonsymmetric and Indefinite Finite Element Matrices*, *J. Numer. Linear Algebra*, 1:59-76, 1992.
- [31] Matlab Version 5 - Computer Software by MATHWORKS, 1996.

IMAGE EVALUATION TEST TARGET (QA-3)

APPLIED IMAGE, Inc.
1653 East Main Street
Rochester, NY 14609 USA
Phone: 716/482-0300
Fax: 716/288-5989

© 1993, Applied Image, Inc., All Rights Reserved