UNIVERSITY OF CALIFORNIA

Los Angeles

Probability and Symmetry in Computational Linear Algebra

A dissertation submitted in partial satisfaction of the

requirements for the degree Doctor of Philosophy

in Mathematics

by

Man-Chung Yeung

1997

© Copyright by

Man-Chung Yeung

1997

The dissertation of Man-Chung Yeung is approved.

Bjorn Engquist

Stott Parker

Eitan Tadmor

Tony F. Chan, Committee Chair

University of California, Los Angeles

1997

TABLE OF CONTENTS

1 Introduction **1**

1.1 Gaussian Elimination 2

1.2 Krylov subspace methods 6

1.3 Contributions of this thesis 10

1.3.1 Probabilistic analysis of GE without pivoting 10

1.3.2 Symmetry in preconditioned Krylov subspace methods . . . 12

1.3.3 ML(k)BiCGSTAB 14

2 Probabilistic Analysis of Gaussian Elimination Without Pivoting and Its Application **19**

2.1 Properties of a Gaussian Matrix 19

2.2 Real Gaussian Elimination 21

2.2.1 Density Functions of u_{pq} and l_{pq} 21

2.2.2 Probability of Small Pivot 29

2.2.3 Probability of Large Growth Factor 30

2.2.4 Numerical Experiments 34

2.3 Complex Gaussian Elimination 37

2.3.1 Density Functions of u_{pq} 38

2.3.2 Density Functions of l_{pq} 43

2.4 Application 45

3 Preserving Symmetry in Preconditioned Krylov Subspace Methods 54

3.1 Symmetric preconditioning in GMRES 55

3.2 Truncated iterative methods 60

3.3 Symmetric preconditioning in Bi-CG 65

3.4 Numerical Experiments 67

3.4.1 Truncated iterative methods 68

3.4.2 Breakdown behavior of Bi-CG 72

3.5 Conclusions 76

4 ML(k)BiCGSTAB: A BiCGSTAB Variant Based on Multiple Lanczos Starting Vectors 77

4.1 A Lanczos method for k Left Starting Linear Independent Vectors 78

4.2 ML(k)BiCG: A BiCG Variant Based on Multiple Starting Vectors Lanczos 82

4.3 ML(k)BiCGSTAB: A BiCGSTAB Variant Based on Multiple Starting Vectors Lanczos 87

4.4 Numerical Experiments 107

4.5 Appendix 115

Bibliography **120**

LIST OF FIGURES

2.1 (a) Distribution of $u_{12,12}$: observed(+), predicted(-). (b) Distribution of $u_{31,31}$: observed(+), predicted(-). 35

2.2 (a) Distribution of $l_{13,12}$: observed(+), predicted(-). (b) Distribution of $l_{30,29}$: observed(+), predicted(-). 36

2.3 (a) Overlap of Figures 1(a) and 1(b). (b) Percentage frequency distributions of ρ_L (dashed) and ρ_U (solid). $n = 25$. 36

2.4 Percentage frequency distributions of ρ_L (dashed) and ρ_U (solid). (a) $n = 50$. (b) $n = 100$. 37

2.5 (a) Distribution of $u^R_{12,12}$: observed(+), predicted(-). (b) Distribution of $|u_{12,12}|$: observed(+), predicted(-). 45

2.6 (a) Distribution of $u^R_{31,31}$: observed(+), predicted(-). (b) Distribution of $|u_{31,31}|$: observed(+), predicted(-). 46

2.7 (a) Distribution of $l^R_{13,12}$: observed(+), predicted(-). (b) Distribution of $|l_{13,12}|$: observed(+), predicted(-). 46

2.8 Observed distribution (+) of the elements of U_{22} vs. the $N(0, 1.77)$ distribution (-). $k = 20, n = 51$. 49

2.9 (a) Observed distribution (+) of the $(12,12)$-th entry of $\tilde{U}_{22}$ vs. the predicted distribution $0.7 f_{u_{12,12}}(0.7t)$ (-). (b) Observed distribution (+) of the $(31,31)$-th entry of $\tilde{U}_{22}$ vs. the predicted distribution $0.8 f_{u_{31,31}}(0.8t)$ (-). The functions $f_{u_{12,12}}(t)$ and $f_{u_{31,31}}(t)$ are defined in Theorem 2.1. 50

2.10 (a) Observed distribution (+) of the $(13,12)$-th entry of $\tilde{L}_{22}$ vs. the predicted distribution $f_{l_{13,12}}(t)$ (-). (b) Observed distribution (+) of the $(30,29)$-th entry of $\tilde{L}_{22}$ vs. the predicted distribution $f_{l_{30,29}}(t)$ (-). The functions $f_{l_{13,12}}(t)$ and $f_{l_{30,29}}(t)$ are defined in Theorem 2.2. 50

2.11 Percentage frequency distributions of $\rho_{\bar{L}_{22}}$ (dashed) and $\rho_{\bar{U}_{22}}$ (solid). (a) $k = 20, n = 45$. (b) $k = 20, n = 70$. See Figures 2.3 and 2.4 for the meaning of α. 51

2.12 Percentage frequency distributions of $\rho_{\bar{L}_{22}}$ (dashed) and $\rho_{\bar{U}_{22}}$ (solid). (a) $k = 20, n = 120$. (b) $k = 20, n = 220$. See Figures 2.3 and 2.4 for the meaning of α. 51

3.1 Minimum cosines in right-preconditioned Bi-CG (solid line) and symmetrically-preconditioned Bi-CG (dashed line) for the $Re = 1$ problem. 76

4.1 Example 1: with no preconditioning 111

4.2 Example 2: with no preconditioning 111

4.3 Example 2: with ILU(0) preconditioning 112

4.4 Example 3: with ILU(0) preconditioning 112

4.5 Number of matvec's / $10n$ vs. k for the matrices HOR131 (a) and ORSIRR1 (b). "o" denotes no convergence within $10n$ matvec's. For $k = 1$, we plotted the number of matvec's for BiCGSTAB. Note that there is a dramatic improvement in performance as k increases from 1, but that for $k \geq 30$, the performance is not sensitive to k. . 114

LIST OF TABLES

2.1 Probabilities of small pivot. 35

2.2 12500 matrices from Class 1 were selected in this experiment. B: the number of matrices which make GE without pivoting performed on U_{22} breakdown; EP: the empirical probability of breakdown when GE without pivoting performed on U_{22} 52

3.1 Mat-Vec's for convergence for right-preconditioned methods. 70

3.2 Mat-Vec's for convergence for symmetric right-preconditioned methods. 71

3.3 Frequencies of minimum cosines for right-preconditioned (first row of each pair of rows) and symmetrically-preconditioned (second row of each pair of rows) Bi-CG. 74

3.4 Frequencies of minimum cosines when r_0^* is chosen randomly. . . . 75

3.5 Steps and minimum cosines for the driven cavity problem. 75

4.1 Average cost per step of the preconditioned ML(k)BiCGSTAB. . . 107

4.2 Comparison of Methods on a Representative Group of Matrices from the Harwell-Boeing Collection. ML(25), ML(50) and ML(100) stand for ML(25)BiCGSTAB, ML(50)BiCGSTAB and ML(100)BiCGSTAB respectively. The numbers in the table are number of matvec's. "-" means no convergence within $20n$ matvec's for BiCG and $10n$ for the other methods, "b" denotes breakdown, "o" denotes overflow. . 110

ACKNOWLEDGMENTS

I wish to express my sincere gratitude to:

My thesis advisor, Prof. Tony Chan, for his patience, excellent guidance and encouragement throughout the completion of the thesis. His careful reading of this thesis and his constructive suggestions are also very much appreciated.

My committee members, Profs Bjorn Engquist, Stott Parker and Eitan Tadmor for their involvement in this project.

Prof. Alan Edelman for providing me with several important references and suggestions on an earlier draft of Chapter 2 which improves the derivations of equations (2.1) and (2.5). In addition, I would like to thank Profs Gene Golub and Nick Trefethen for their valuable comments and suggestions on the earlier draft of Chapter 2.

Dr. Edmond Chow and Prof. Youcef Saad for their kindly help and collaboration in the research of Chapter 3. In addition, sincere thanks go to Prof. H. A. van der Vorst for his valuable comments on an earlier draft of Chapter 4.

The faculty, staff and students of the Department of Mathematics for their help and supports.

The author was supported by the National Science Foundation under contract ASC-92-01266, ONR under contract ONR-N00014-92-J-1890 and by the Army Research Office under contract DAAL-03-9-C-0047 (Univ. of Tenn. subcontract ORA 4466.04, Amendment 1 and 2).

PUBILCATIONS

R. Chan, X. Jin and M. Yeung, *The circulant operator in the Banach algebra of matrices,* Linear Algebra Appls., Vol. 149 (1991), pp. 41-53.

——, *The spectra of super-optimal circulant preconditioned Toeplitz systems,* SIAM J. Numer. Anal., Vol. 28 (1991), pp. 871-879.

R. Chan and M. Yeung, *Circulant preconditioners for Toeplitz matrices with positive continuous generating functions,* Math. Comp., Vol. 58 (1992), pp. 233-240.

——, *Circulant preconditioners constructed from kernels,* SIAM J. Numer. Anal., V29(1992), pp. 1093-1103.

——, *Jackson's theorem and circulant preconditioned Toeplitz systems,* J. Approx. Theory, 70 (1992), 191-205.

——, *Circulant preconditioners for complex Toeplitz matrices,* SIAM J. Numer. Anal., 30 (1993), 1193-1207.

——, *A note on the Besov space* $B_2^{1/2}$, Southeast Asian Bulletin of Mathematics, Volume 19, Number 1, February 1995, 75-80.

T. Chan, E. Chow, Y. Saad and M. Yeung, *Preserving symmetry in preconditioned Krylov subspace methods,* submitted to SIAM J., Sci. Comp.

S. Chung and M. Yeung, *On a subspace related to the Korovkin closure,* Southeast Asian Bulletin of Mathematics, Volume 18, Number 1, April 1994, 11-18.

M. Yeung and R. Chan, *Circulant preconditioners for Toeplitz matrices with piecewise continuous generating functions,* Math. Comp., 61 (1993), 701-718.

M. Yeung and T. Chan, *Probabilistic analysis of Gaussian elimination without pivoting,* SIAM J., Matrix Anal. Appl., Vol. 18, No. 2, pp. 499-517, April 1997.

——, *ML(k)BiCGSTAB: A BiCGSTAB variant based on multiple Lanczos starting vectors,* submitted to SIAM J., Sci. Comp.

ABSTRACT OF THE DISSERTATION

Probability and Symmetry in Computational Linear Algebra

by

Man-Chung Yeung

Doctor of Philosophy in Mathematics

University of California, Los Angeles, 1997

Professor Tony F. Chan, Chair

This thesis is divided into two parts. In the first part (Chapter 2), we study Gaussian elimination by means of probabilistic analysis and in the second one (Chapters 3 and 4), we study how to preserve symmetry in preconditioned Krylov subspace methods and develop a new Krylov subspace solver for the solution of a nonsymmetric linear system $Ax = b$.

The numerical instability of Gaussian elimination is proportional to the size of the L and U factors that it produces. The worst case bounds are well known. For the case without pivoting, breakdowns can occur and it is not possible to provide *a priori* bounds for L and U. For the partial pivoting case, the worst case bound is $O(2^n)$, where n is the size of the system. Yet, these worst case

bounds are seldom achieved, and in particular Gaussian elimination with partial pivoting is extremely stable in practice. Surprisingly, there has been relatively little theoretical study of the "average" case behaviour. The purpose of our study in Chapter 2 is to provide a probabilistic analysis of the case without pivoting. The distribution we use for the entries of A is the normal distribution with mean 0 and unit variance. We first derive the distributions of the entries of L and U. Based on this, we prove that the probability of the occurence of a pivot less than ϵ in magnitude is $O(\epsilon)$. We also prove that the probabilities $Prob(||U||_\infty / ||A||_\infty > n^{2.5})$ and $Prob(||L||_\infty > n^3)$ decay algebraically to zero as n tends to infinity. Numerical experiments are presented to support the theoretical results. By combining our theoretical results and the observations of Trefethen and Schreiber [57] on random matrices, we propose a new method of using GE which may be particularly suited to certain kinds of parallel computers, which greatly reduces data movements while avoiding breakdown and instability effectively.

In Chapter 3, we consider the problem of solving a linear system $Ax = b$ when A is nearly symmetric and when the system is preconditioned by a symmetric positive definite matrix M. In the symmetric case, one can recover symmetry by using M-inner products in the Conjugate Gradient Algorithm. Inner products can also be used in the nonsymmetric case, and near symmetry can be preserved similarly. We explore the implementation issues associated with this technique and compare a few different options.

We present in Chapter 4 a variant of the popular BiCGSTAB method for solving nonsymmetric linear systems. The method, which we denote by ML(k)BiCGSTAB, is derived from a variant of the BiCG method based on a Lanczos process using multiple ($k > 1$) starting left Lanczos vectors. Compared with the original BiCGSTAB method, our new method produces a residual polynomial which is of lower degree after the same number of steps but which also requires fewer matrix-vector products to generate, on average requiring only $1 + 1/k$ matvec's per step. Empirically, it also seems to be more stable and faster convergent. The new method can be implemented as a k-term recurrence and can be viewed as a bridge connecting the Arnoldi-based FOM/GMRES methods and the Lanczos-based BiCGSTAB methods.

CHAPTER 1

Introduction

One of the most frequently encountered tasks in numerical computation is solving a square and usually large linear system

$$Ax = b.$$

There are two types of approach for the solution of the system: direct methods, which are basically variations of Gaussian elimination (GE), and iterative ones, which come in many flavours. Direct methods are chosen when A is dense and unstructured. When A does not fall into this category, iterative methods are often of interest. Among iterative methods, Krylov subspace methods feature short term recurrences and therefore require low cost and low memory. In this Chapter, we give a description of GE and Krylov subspace methods in §1.1 and §1.2 respectively and summarize the contributions of this thesis in §1.3.

1.1 Gaussian Elimination

GE is the most common general method for solving an square, dense, unstructured linear system

$$Ax = b, \tag{1.1}$$

where A is an $n \times n$ real or complex matrix and b is a real or complex vector of length n. The basic idea of the method is to convert the system (1.1) to an equivalent triangular system. The conversion is built on the following theorem [31].

Theorem 1.1 *If all the principal submatrices of A are nonsingular, then there is a unit lower triangular matrix L and an upper triangular matrix U with nonzero diagonal entries such that*

$$A = LU\,.$$

If A is nonsingular, then there is a permutation matrix P, a unit lower triangular matrix L with no entry bigger than 1 in absolute value and an upper triangular matrix U with nonzero diagonal entries such that

$$PA = LU\,.$$

If A is nonsingular, then there are two permutation matrices P and Q, a unit lower triangular matrix L and an upper triangular matrix U, where no entry in L or U is bigger than 1 in absolute value and all the diagonal entries of U are

nonzero, such that

$$PAQ = LU\,.$$

Corresponding to the different cases in Theorem 1.1, there are three fundamental versions of GE:

- GE without pivoting: Compute $A = LU \rightarrow$ Solving $Ly = b \rightarrow$ Solving $Ux = y$;
- GE with partial pivoting: Compute $PA = LU \rightarrow$ Solving $PLy = b \rightarrow$ Solving $Ux = y$;
- GE with complete pivoting: Compute $PAQ = LU \rightarrow$ Solving $PLy = b \rightarrow$ Solving $UQx = y$;

For the details of the implementations of these GE versions, one is referred to [31].

The conditions in Theorem 1.1 are only sufficient. In some cases, the LU factorization of A need not exist. For example, one can not find l_{ij} and u_{ij} such that

$$\begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 7 \\ 1 & 6 & 3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ l_{21} & 1 & 0 \\ l_{31} & l_{32} & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} & u_{13} \\ 0 & u_{22} & u_{23} \\ 0 & 0 & u_{33} \end{bmatrix}.$$

To see this, one can equate entries to get

$$u_{11} = 1;$$

$$l_{21}u_{11} = 2 \rightarrow l_{21} = 2/u_{11} = 2;$$

$$l_{31}u_{11} = 1 \rightarrow l_{31} = 3/u_{11} = 1;$$

$$u_{12} = 2;$$

$$l_{21}u_{12} + u_{22} = 4 \rightarrow u_{22} = 0.$$

But when one then looks at the $(3,2)$-th entry one obtains the contradictory equation $6 = l_{31}u_{12} + l_{32}u_{22} = 4$.

In practice, if some diagonal entry $u_{i_0 i_0}$ of U is zero or close to zero, say, $u_{i_0 i_0}$ is less than the machine precision in absolute value, then the LU factorization of A can not be computed, or in other words, GE breaks down. This is because $u_{i_0 i_0}$ will be used as a denominator in the next step of the computations. Among the three versions of GE above, it is always true that GE with partial or complete pivoting has less chance to break down than GE without pivoting. This can be understood in part from Theorem 1.1 since the condition that all the principal submatrices of A are nonsingular implies that A is nonsingular but not true conversely. In the literature of GE, the diagonal elements of U are called pivot elements.

When GE is performed on the system (1.1) in floating point arithmetic, the computed LU factors $\hat{L}$ and $\hat{U}$ are produced. Then, by solving two corresponding triangular systems, we obtain the solution $\hat{x}$ to (1.1). The computed solution $\hat{x}$ satisfies

$$(A + E)\,\hat{x} = b$$

with

$$|E| \leq n\boldsymbol{u}\left(3\,|A| + 5\,|\hat{L}|\,|\hat{U}|\right) + O(\boldsymbol{u}^2)$$

in the case of GE without pivoting and

$$|E| \leq n\boldsymbol{u}\left(3\,|A| + 5\,\hat{P}^T|\hat{L}|\,|\hat{U}|\right) + O(\boldsymbol{u}^2)$$

in the case of GE with partial pivoting, where $\hat{P}$ is the computed analogs of P. Here $\boldsymbol{u}$ is the unit roundoff and for any matrix M, we use $|M|$ to denote the matrix obtained by taking the absolute value of the elements of M [31, P.106, P.115]. From this, it follows that

$$\|E\|_\infty \leq n\boldsymbol{u}\|A\|_\infty \left(3 + 5\|\hat{L}\|_\infty \frac{\|\hat{U}\|_\infty}{\|A\|_\infty}\right) + O(\boldsymbol{u}^2)\,.$$

We define

$$\rho_L = \|L\|_\infty\,, \qquad \rho_U = \|U\|_\infty / \|A\|_\infty \tag{1.2}$$

and call ρ_L and ρ_U the growth factors. It is obvious that the smaller the growth factors are, the more accurate the computed solution $\hat{x}$ is.

GE with partial pivoting is extremely stable in practice. However, this stability can not be guaranteed. The worst case examples are well known: the "growth factor" can be as large as $O(2^n)$ (and can occur in practical applications [24]). Theoretical studies on the numerical stability of GE have been made since 1940s by a great number of authors, for example, Turing [59], von Neumann and Goldstine [61], [62], Wilkinson [63], [64], and so on. Recently, Trefethen and Schreiber [57] considered the topic from a statistical viewpoint.

1.2 Krylov subspace methods

A Krylov subspace method is an iterative method for the solution of (1.1) when A is sparse, or probably structured. At the l-th iteration of the method, the approximate solution x_l is defined by the Petrov-Galerkin condition

$$x_l \in x_0 + K_l(A, r_0) \equiv x_0 + span\{r_0, Ar_0, \cdots, A^{l-1}r_0\}$$

and

$$b - Ax_l \perp \mathcal{L}_l,$$

where $\mathcal{L}_l$ is a subspace in $\mathcal{R}^n$ of dimension l and where $r_0 = b - Ax_0$ is the initial residual for the initial guess x_0. The subspace $K_l(A, r_0)$ is called a Krylov subspace. The existing Krylov subspace methods in the literature are defined with special choices of $\mathcal{L}_l$. For example, we obtain the BiCG method [22, 39, 50] when we choose

$$\mathcal{L}_l = K_l(A^T, r_0^*),$$

where r_0^* is an arbitrary vector, and the FOM method [47, 48, 50] when

$$\mathcal{L}_l = K_l(A, r_0).$$

The classical Conjugate Gradient method (CG) [36] is a special version of BiCG where A is symmetric.

Let v be a vector and $\nu_{(A,v)}$ be the grade of v with respect to A, i.e., the degree of the minimal polynomial of v with respect to A. Given two vectors v_0 and v_0^*,

we define

$$W_l = \begin{bmatrix} v_0^{*T} v_0 & v_0^{*T} A v_0 & \cdots & v_0^{*T} A^{l-1} v_0 \\ v_0^{*T} A v_0 & v_0^{*T} A^2 v_0 & \cdots & v_0^{*T} A^l v_0 \\ \vdots & \vdots & & \vdots \\ v_0^{*T} A^{l-1} v_0 & v_0^{*T} A^l v_0 & \cdots & v_0^{*T} A^{2(l-1)} v_0 \end{bmatrix}, \qquad l = 1, 2, \cdots, \nu,$$

where $\nu = \min\{\nu_{(A,v_0)}, \nu_{(A^T,v_0^*)}\}$. The derivation of BiCG is based on the Lanczos process [38] which is figured out by the following theorem.

Theorem 1.2 *Given v_0 and v_0^*. If the principal submatrices of W_ν are nonsingular, there exist $n \times \nu$ matrices $V_\nu = [v_0, v_1, \cdots, v_{\nu-1}]$ and $V_\nu^* = [v_0^*, v_1^*, \cdots, v_{\nu-1}^*]$ of rank ν, whose first columns are v_0 and v_0^* respectively, and a $\nu \times \nu$ tridiagonal matrix T_ν with all the entries $t_{j+1,j} = 1$ in its lower subdiagonal, such that*

$$(v_l, v_{l'}^*) = \delta_{l,l'}, \qquad l, l' = 0, 1, \cdots, \nu - 1,$$

where $\delta_{l,l'} = 0$ if $l \neq l'$ and $\delta_{l,l'} \neq 0$ if $l = l'$, and

$$A V_\nu = V_\nu T_\nu$$

and

$$A^T V_\nu^* = V_\nu^* T_\nu.$$

Such V_ν, V_ν^* and T_ν are unique.

In practice, the Lanczos process could face breakdown - divisions by 0 - called *Lanczos breakdown*, when the Lanczos vectors v_l and v_l^* are exactly or nearly linear

dependent. Techniques called look-ahead have been developed to cure this kind of breakdown, see, for instance, [10, 11, 26, 27, 32, 34, 44, 45, 56] and the references given therein. Moreover, in the actual derivation of the BiCG algorithm, the matrix T_ν is decomposed into two triangular matrices and as a result, BiCG is a two-term recurrence method. Since such a decomposition of T_ν need not exist, BiCG has another potential breakdown called *pivot breakdown* in addition to the Lanczos one. Several methods have been produced to remedy pivot breakdowns, e.g., CSBCG [6, 7] and QMR [29].

FOM is derived from the Arnoldi process [3] which is charactered in the theorem below.

Theorem 1.3 *Given v_0 with $\|v_0\|_2 = 1$. There exist an $n \times \nu$ orthonormal matrix $V_\nu = [v_0, v_1, \cdots, v_{\nu-1}]$ of rank ν, whose first column is v_0, and a $\nu \times \nu$ Hessenberg matrix H_ν such that*

$$AV_\nu = V_\nu H_\nu,$$

where $\nu = \nu_{(A,v_0)}$. Such V_ν and H_ν are unique.

Mathematically, no breakdown occurres in the Arnoldi process. Similarly to the case of BiCG, the matrix H_ν is factorized into two triangular matrices in deriving the FOM algorithm and therefore, pivot breakdown is possible in FOM. GMRES [47, 48, 50] is an improved version of FOM which not only remedies pivot breakdowns in FOM but minimizes the residual at each iteration. FOM and GMRES are long-term recurrence methods involving all previously computed

iterates and residuals, unless truncated or restarted, otherwise they are practically impossible. However, both truncated and restarted treatments lose the convergence history.

BiCG and QMR require to perform a matrix-by-vector product with the transpose matrix A^T at each iteration. Since A^T may not be available in some applications [50, p.214], methods avoiding transpose product have been developed. Examples of these include TFiQMR [14], CGS [55], BiCGSTAB [60], BiCGSTAB2 [33], TFQMR [25] and BiCGSTAB(k) [53]. In principle, these methods compute a residual characterized by a product of the BiCG residual with a polynomial of the matrix A. Typical examples are CGS and BiCGSTAB. The residuals r_l^{CGS} and $r_l^{BiCGSTAB}$ of CGS and BiCGSTAB respectively at the l-iteration are defined by

$$r_l^{CGS} = \psi_l(A) r_l^{BiCG}$$

and

$$r_l^{BiCGSTAB} = \phi_l(A) r_l^{BiCG},$$

where $\psi_l(\lambda)$ is the polynomial of degree l such that $r_l^{BiCG} = \psi_l(A) r_0$ and where $\phi_l(\lambda) \equiv (\rho_l \lambda + 1)\phi_{l-1}(\lambda)$ and $\phi_0(\lambda) \equiv 1$. Transpose-free methods which cure breakdown have also been developed, e.g., [12, 30].

1.3 Contributions of this thesis

In this thesis, we study GE without pivoting from the point of view of probability (Chapter 2), how to preserve symmetry in preconditioned Krylov subspace methods (Chapter 3) and also develop a new Krylov subspace solver to the system (1.1) (Chapter 4).

1.3.1 Probabilistic analysis of GE without pivoting

Recently, Trefethen and Schreiber(TS) [57] studied the numerical stability of GE with pivoting from the viewpoint of statistics. Among their many results, they observed that for many distributions of matrices, the matrix elements after the first few steps of Gaussian elimination with (partial or complete) pivoting are approximately normally distributed. They also found that, for $n \leq 1024$, the average growth factor (normalized by the standard deviation of the initial matrix elements) is within a few percent of $n^{2/3}$ for the partial pivoting case and approximately $n^{1/2}$ for the complete pivoting case. After having performed more extensive experiments, Edelman and Mascarenhas [20] further suggest that the growth factor in the partial pivoting case may grow more like $n^{1/2}$ than $n^{2/3}$.

Following TS, we study the probability of small pivots and large growth factors in Chapter 2. Howeover, we will only consider the case without pivoting. We are doing so for three reasons. The first is quite obvious: the non-pivoting case is far

easier to analyze than the pivoting case. In particular, we are able to derive in closed form the density functions of the elements of the LU factors and probabilistic bounds for the occurence of small pivots and the growth factors. The second reason is that, with the advent of parallel computing, there is more incentive to trade off the stability of partial pivoting for the higher performance of simpler but possibly less stable forms of GE, including no pivoting, see, for instance, [42, 43]. Finally, we are hoping that our results for GE without pivoting will be useful in the analysis of, as well as providing a basis of comparison for, the partial pivoting case.

The class of random matrices we chose to study is that of (complex) Gaussian matrices. By definition, a (complex) Gaussian matrix is a random matrix with independent and identically distributed elements which are $(\tilde{N}(0,1))$ $N(0,1)$, the normal distribution with mean 0 and variance 1. This choice is motivated by the empirical results of TS mentioned earlier. Matrices of this type have also been studied by Edelman [17], [19], who derived the expected singular values.

On some kinds of parallel computers, say, systolic systems [37], avoiding data movements is necessary [40]. Based on our theoretical results and TS's observations, we suggest that we may use GE in such a way that, in the first few steps, we use GE with partial (or complete) pivoting and then use the version without pivoting (or with pairwise, neighbor pivoting). Our experimental results illustrate that this idea may minimize data movements while avoiding breakdown and instability of GE.

1.3.2 Symmetry in preconditioned Krylov subspace methods

Consider the solution of the linear system (1.1) by a preconditioned Krylov subspace method. Assume at first that A is symmetric positive definite (SPD) and let M be an SPD matrix that is a preconditioner for the matrix A. Then one possibility is to solve either the left-preconditioned system

$$M^{-1}Ax = M^{-1}b \tag{1.3}$$

or the right-preconditioned system

$$AM^{-1}u = b, \quad x = M^{-1}u \ . \tag{1.4}$$

Both of these systems have lost their symmetry. A remedy exists when the preconditioner M is available in factored form, e.g., as an incomplete Choleski factorization,

$$M = LL^T,$$

in which case a simple way to preserve symmetry is to *split* the preconditioner between left and right, i.e., solve

$$L^{-1}AL^{-T}u = L^{-1}b, \quad x = L^{-T}u.$$

This can also be done when M can be factored as $M = M^{1/2} \times M^{1/2}$. Unfortunately, the requirement that M be available in factored form is often too stringent.

However, this remedy is not required. As is well-known, we can preserve symmetry by using a different inner product. Specifically, we observe that $M^{-1}A$ is self-adjoint for the M-inner product,

$$(x, y)_M = (Mx, y) = (x, My)$$

since we have

$$(M^{-1}Ax, y)_M = (Ax, y) = (x, Ay) = (x, M(M^{-1}A)y) = (x, M^{-1}Ay)_M.$$

Indeed, this is how the preconditioned conjugate gradient (PCG) and symmetrically preconditioned conjugate residual algorithms may be derived [31, 35, 50].

The question we now raise is the following. When A is only nearly symmetric, it is often the case that there exists a preconditioner M which is SPD. In this situation, the use of either of the forms (1.3) or (1.4) is just as unsatisfactory as in the fully symmetric case. Indeed, whatever degree of symmetry was available in A is now entirely lost. Although the above remedy based on M-inner products is always used in the symmetric case, it is rather surprising that this problem is seldom ever mentioned in the literature for the nearly symmetric case. In the nonsymmetric case, when M exists in factored form, some form of balancing can also be achieved by splitting the preconditioner from left and right. However, there does not seem to have been much work done in exploiting M-inner products even when M is not available in factored form. This dichotomy between the treatment of the symmetric and the nonsymmetric cases is what motivated the study in Chapter

3, where we suppose A is nearly symmetric and show how alternative inner products may be used to preserve the degree of symmetry of A in preconditioned Krylov subspace methods.

Ashby, Manteuffel, and Saylor [4] have fully considered the case of using alternate inner products when the matrix A is symmetric. Work of which we are aware that consider the use of alternate inner products when A is near-symmetric are Young and Jea [65] and Meyer [41]. In the latter, the $A^T M^{-1} A$ inner product (with M SPD) is used with ORTHOMIN and ORTHODIR.

1.3.3 ML(k)BiCGSTAB

BiCGSTAB is a popular Krylov subspace method for the iterative solution of nonsymmetric linear systems. Its main features are that it is transpose-free, makes more efficient use of matrix-vector products when compared to BiCG and is more stable than CGS, but it faces potential breakdown. In Chapter 4, we introduce a new variant of BiCGSTAB which inherits all of these nice features and reduces the probability of breakdown to zero. In addition, the key new ingredient of our method is the use of multiple starting left Lanczos vectors which has the desirable effect of lowering the cost per step and increasing the robustness.

BiCGSTAB is derived from BiCG which is a Lanczos based Krylov subspace method. In BiCG, the residual vector r_l at the l-th step lies in a Krylov subspace

$K_{l+1}(A, r_0)$ and is chosen to be

$$r_l \perp \mathcal{L}_l = K_l(A^T, r_0^*).$$

In our variant of the BiCG method, which we denote by ML(k)BiCG, r_l is still in $K_{l+1}(r_0, A)$ but is now chosen to be

$$r_l \perp \mathcal{L}_l = span\{p_1, p_2, \cdots, p_l\},$$

where $p_{jk+i} \equiv A^{Tj} q_i, i = 1, 2, \cdots, k; j = 0, 1, 2, \cdots$, and q_i's are k arbitrary vectors. In the context of Krylov subspaces, $\mathcal{L}_l$ is the union of k Krylov subspaces $K_{j+1}(A^T, q_s)$ and $K_j(A^T, q_{s'})$, where $1 \leq s \leq i < s' \leq k$ and $jk + i = l$, generated from k linearly independent starting vectors q_i.

Our motivation for using multiple starting Lanczos vectors is to mitigate somewhat the ill-conditioning of $K_l(A^T, r_0^*)$ for large l by replacing a high degree Krylov polynomial corresponding to one starting vector by a union of lower degree Krylov polynomials generated from different, independent starting vectors. We think this leads to better stability and robustness of the resulting iterative method and, by choosing the starting vectors randomly, we may avoid breakdown in practice. We derive an efficient implementation of this idea, requiring only memory of the previous k iterates (i.e. a k-term recurrence).

But we consider the major contribution in Chapter 4 to be an extension of BiCGSTAB, which we denote by ML(k)BiCGSTAB, using multiple starting Lanczos vectors. The derivation is similar to, but rather more complicated than,

that of deriving BiCGSTAB from BiCG. Not only does our method inherits from ML(k)BiCG the increased stability of using multiple starting Lanczos vectors, it also inherits from BiCGSTAB the advantages of being transpose-free and also more efficient in using matvec per step than ML(k)BiCG. Specifically, after $l = jk + i$ steps, where $i = 1, \cdots, k$ and $j = 0, 1, \cdots$, the residual vector $\tilde{r}_l$ can be written as $\tilde{r}_l = \psi_{j+1}(A)\phi_l(A)r_0$, where ϕ_l is the degree l polynomial corresponding to the residual vector r_l of ML(k)BiCG, and ψ_{j+1} is a degree $j+1$ smoothing polynomial. Thus, the "degree" of $\tilde{r}_l$ is $l+j+1$. To compute $\tilde{r}_l$, exactly $l+j+1$ matvec's with A (no A^T) are required. Thus, the *cost per step on the average* is $1+1/k$ matvec's. Both ML(k)BiCG and ML(k)BiCGSTAB can be implemented efficiently as k-term recurrences.

A way to view our method is through the conditioning of the collection of vectors to which the residual vector is required to be orthogonal, which we believe controls the stability of the method. For FOM/GMRES, these vectors are mutually orthogonal and thus is perfectly conditioned. For BiCG, these vectors are the Lanczos vectors and they can be ill-conditioned. For our new ML(k)BiCG, these vectors consist of a union of k sets of Lanczos vectors, generated by (arbitrary) initial vectors which can be made orthogonal, thus making them better conditioned than in the BiCG case. Our new ML(k)BiCGSTAB method, being a product method derived from ML(k)BiCG, inherits this increased stability. Thus, ML(k)BiCGSTAB can be viewed as an attempt to merge the advantages of

FOM/GMRES (stability) and BiCGSTAB (short recurrence, transpose-free and efficient use of matvec's) while avoiding their disadvantages: FOM/GMRES (long recurrence) and BiCGSTAB (imperfect stability).

Our ML(k)BiCGSTAB method has a close relationship to the *Lanczos-type method for multiple starting vectors* proposed recently by Aliaga, Boley, Freund and Hernández (ABFH) [1]. In fact, even though we have developed our methods independently of the ABFH framework, our BiCG extension ML(k)BiCG can be easily derived from the ABFH framework, using one right Lanczos vector and k left Lanczos vectors. Even so, ML(k)BiCG deserves some interest on its own right, because we believe it is the first attempt in using multiple starting vector Lanczos-type methods for solving a *single* linear system.[1] The main application of nonequal left and right starting vectors cited in [1] is for computing transfer functions in multi-input multi-output time invariant linear dynamical systems. A more significant difference between our present results and those in [1] is that we have derived a BiCGSTAB variant based on multiple starting vectors, with the advantages stated earlier. We believe this is the first *product* Krylov method based on multiple starting vectors.

The origin of the ideas behind the new methods presented here can be traced to Ruhe's vectorwise implementation of the block Lanczos method [46], in the same way that [1] can be considered an extension of Ruhe's method to non-Hermitian

[1] Freund and Malhotra [28] did consider a QMR-type method based on the ideas in [1] for linear systems with multiple right-hand-sides, but there the number of right and left Lanczos vectors are the same.

matrices and non-equal left and right starting vectors plus look-ahead. As such, we believe our methods inherit the advantage of faster convergence of the underlying block Lanczos method.

Gutknecht [33], Sleijpen and Fokkema [53] recently generalized BiCGSTAB to versions called BiCGSTAB2 and BiCGSTAB(k) respectively, in which they replaced the GMRES(1) part in BiCGSTAB with GMRES(k). The purpose of doing so is to increase the robustness of BiCGSTAB. From our experimental results, ML(k)BiCGSTAB appears to be a good alternative to achieve this goal exploiting the robustness of GMRES(k)/FOM(k).

CHAPTER 2

Probabilistic Analysis of Gaussian Elimination Without Pivoting and Its Application

Let X be an $n \times n$ square (complex) Gaussian matrix and let $X = LU$ be the LU factorization of X. We derive the density functions of the entries of L and U respectively and prove that the probability of the occurence of a pivot less than ϵ in magnitude is $O(\epsilon)$. [1] We also derive bounds on the probabilities of large growth factors. In particular, we prove that the probabilities $Prob(||U||_\infty/||A||_\infty > n^{2.5})$ and $Prob(||L||_\infty > n^3)$ decay algebraically to zero as n tends to infinity. From our experimental results, we observe that the probabilities $Prob(n \leq ||L||_\infty < n^{1.5})$ and $Prob(n \leq ||U||_\infty/||A||_\infty < n^{1.5})$ tend to one as n goes to infinity. This indicates that our theoretical bounds are not the tightest possible but not too loose either.

2.1 Properties of a Gaussian Matrix

We present some properties involving Gaussian matrices which will be used in the following sections. These properties can be derived easily from classical facts

[1] We note that Foster [23] has studied the probability of large diagonal elements in the QR factorization of a rectangular matrix A.

that may be found in standard texts or papers, for example, [2, 9, 18, 52, 58].

Let $N(0,1)$ refer to the normal distribution with mean 0 and variance 1 and let $\tilde{N}(0,1)$ the complex distribution of $x+yi$ where x and y are independent and identically distributed (iid) $N(0,1)$. By definition, an (complex) Gaussian matrix is a random matrix with iid elements which are $(\tilde{N}(0,1))$ $N(0,1)$.

Proposition 2.1 *Let v and X be two independent (complex) Gaussian matrices of sizes $n \times 1$ and $n \times n$ respectively. Let H be the Householder matrix such that*

$$Hv = (s, 0, \cdots, 0)^T$$

where $s \geq 0$ and let

$$Y = X(PH)^T,$$

where P is a permutation matrix such that $PHv = (0, \cdots, 0, s)^T$. Then the entries $s, y_{ij}, i, j = 1, \cdots, n$, are independent and s^2 is (χ^2_{2n}) χ^2_n while all y_{ij} are $(\tilde{N}(0,1))$ $N(0,1)$. Moreover, Let

$$X = QR$$

be the QR factorization of X obtained by performing the standard Householder transformations and let

$$w = Q^{-1}v.$$

Then the entries $r_{ij}, w_i, i = 1, \cdots, n, j = i, \cdots, n$ are independent and r_{ii} is $(\chi^2_{2(n-i+1)})$ $\chi^2_{n-i+1}, i = 1, \cdots, n$ while all others are $(\tilde{N}(0,1))$ $N(0,1)$.

2.2 Real Gaussian Elimination

In this section, we suppose all the matrices appearing have real entries.

2.2.1 Density Functions of u_{pq} and l_{pq}

Let X be an $n \times n$ Gaussian matrix and let $X = LU$, where L is an unit lower triangular matrix and U is an upper triangular matrix, be the LU factorization of X [2]. The (p,q)-th $(p \leq q)$ entry u_{pq} of U and the entries of X have the following relation.

Lemma 2.1 *Let $X = LU$ be the LU factorization of X. Then*

$$u_{pq} = x_{pq} - x_{p*}^T X_{p-1}^{-1} x_{*q},$$

where

$$x_{p*} = (x_{p1}, \cdots, x_{pp-1})^T,$$

$$x_{*q} = (x_{1q}, \cdots, x_{p-1q})^T,$$

and X_{p-1} is the $(p-1) \times (p-1)$ leading principal submatrix of X.

Proof. Permuting the p-th and q-th columns of X and U simultaneously on both sides of $X = LU$ and then comparing the corresponding blocks, we find

$$\begin{bmatrix} X_{p-1} & x_{*q} \\ x_{p*}^T & x_{pq} \end{bmatrix} = \begin{bmatrix} L_{p-1} & 0 \\ l_{p*}^T & 1 \end{bmatrix} \begin{bmatrix} U_{p-1} & u_{*q} \\ 0 & u_{pq} \end{bmatrix}$$

[2] Since they just form a set of measure zero, we ignore matrices for which the Gaussian elimination fails.

where

$$l_{p*} = (l_{p1}, \cdots, l_{pp-1})^T ,$$

$$u_{*q} = (u_{1q}, \cdots, u_{p-1q})^T$$

and where L_{p-1} and U_{p-1} are the $(p-1) \times (p-1)$ leading principal submatrices of L and U respectively. It follows that

$$X_{p-1} = L_{p-1}U_{p-1} , \qquad l_{p*}^T = x_{p*}^T U_{p-1}^{-1} ,$$

$$u_{*q} = L_{p-1}^{-1} x_{*q} , \qquad u_{pq} = x_{pq} - l_{p*}^T u_{*q}$$

and these imply the desired equation. □

Let H be an $(p-1)\times(p-1)$ orthogonal matrix, e.g., the one PH in Proposition 2.1, such that

$$x_{p*}^T H = (0, \cdots, 0, s) \equiv \eta^T$$

with $s \geq 0$. Then

$$u_{pq} = x_{pq} - \eta^T \left(X_{p-1}H\right)^{-1} x_{*q}$$

$$\equiv x_{pq} - \eta^T Y^{-1} x_{*q} .$$

By Proposition 2.1, the entries s, x_{pq}, x_{iq} and y_{ij}, $i,j = 1, \cdots, p-1$, are mutually independent and all x_{pq}, x_{iq} and y_{ij}, $i,j = 1, \cdots, p-1$, are $N(0,1)$ while s^2 is χ^2_{p-1}. We now decompose Y as

$$Y = QR,$$

where Q is an $(p-1)\times(p-1)$ orthogonal matrix and R an $(p-1)\times(p-1)$ upper triangular matrix with positive diagonal elements. We then further have

$$\begin{aligned} u_{pq} &= x_{pq} - \eta^T R^{-1} Q^T x_{*q} \\ &\equiv x_{pq} - \eta^T R^{-1} w \\ &= x_{pq} - \frac{s w_{p-1}}{r_{p-1p-1}} . \end{aligned} \tag{2.1}$$

Again, by Proposition 2.1, the variables s, x_{pq}, w_i and r_{ij}, $i \le j$, $i, j = 1, \cdots, p-1$, are independent. s^2 is χ^2_{p-1} and r^2_{ii} is χ^2_{p-i}, $i = 1, \cdots, p-1$ and all others are $N(0,1)$.

Set

$$v = \frac{s^2}{(p-1) r^2_{p-1p-1}}$$

and

$$z = \frac{1}{\sqrt{1+(p-1)v}} x_{pq} - \frac{\sqrt{(p-1)v}}{\sqrt{1+(p-1)v}} w_{p-1} ,$$

then v is $F_{p-1,1}$, the F distribution with $p-1$ and 1 degrees of freedom, and z is $N(0,1)$ and (2.1) becomes

$$u_{pq} = z\sqrt{1+(p-1)v} . \tag{2.2}$$

Since the variables v and z are independent and their density functions are known, it is straightforward to determine the density function of u_{pq}.

Theorem 2.1 *Suppose X is an $n \times n$ Gaussian matrix and let $X = LU$ be the LU factorization of X. Then the density function of the (p,q)-th entry of U is*

$$f_{u_{pq}}(t) = \frac{\sqrt{2}}{\pi} \frac{\Gamma(p/2)}{\Gamma((p-1)/2)} \left(\sum_{i=0}^{\lfloor \frac{p-3}{2} \rfloor} \xi_{i,p}\, t^{-2i-2} + (-1)^{\lfloor \frac{p-1}{2} \rfloor}\, \zeta_p\, t^{-p+1} exp(-\frac{1}{2}t^2)\, \phi_p(t) \right) \tag{2.3}$$

where

$$\xi_{i,p} = \begin{cases} (-1)^i \prod_{j=0}^{i-1} (p-2j-3) & i > 0 \\ \\ 1 & i = 0\,, \end{cases}$$

$$\zeta_p = \begin{cases} (p-3)!! & p > 3 \\ \\ 1 & p = 2,\ 3\,, \end{cases}$$

$$\phi_p(t) = \left(\int_0^t exp\left(\frac{1}{2}x^2\right) dx \right)^{p-1-2\lfloor (p-1)/2 \rfloor}$$

and where $-\infty < t < \infty$, $2 \le p \le q$.

Proof. Since the variables v and z in (2.2) are $F_{p-1,1}$ and $N(0,1)$ respectively, the density functions of them are given as follows,

$$f_v(t) = \begin{cases} \frac{1}{\sqrt{\pi}} \frac{\Gamma(p/2)}{\Gamma((p-1)/2)} (p-1)^{(p-1)/2} t^{(p-3)/2} (1+(p-1)t)^{-p/2} & t > 0 \\ \\ 0 & t \le 0\,, \end{cases}$$

and

$$f_z(t) \;=\; \frac{1}{\sqrt{2\pi}}\, exp\,(-t^2/2)\;.$$

Since v and z are independent, their joint density function is given by

$$f(v,z) \;=\; f_v(v)\, f_z(z)$$

$$= \begin{cases} c\, v^{(p-3)/2}(1+(p-1)v)^{-p/2} exp\,(-z^2/2) & v>0 \\ \\ 0 & \text{otherwise}\;, \end{cases}$$

where $c = \dfrac{1}{\sqrt{2\pi}}\,\dfrac{\Gamma(p/2)}{\Gamma((p-1)/2)}(p-1)^{(p-1)/2}$. Thus, the distribution function $F_{u_{pq}}(\alpha)$ of u_{pq} is

$$F_{u_{pq}}(\alpha) \;=\; c\iint_{u_{pq}\le\alpha} v^{(p-3)/2}(1+(p-1)v)^{-p/2} exp\,(-z^2/2)\, dv\, dz$$

$$= \; c\iint_{z\sqrt{1+(p-1)v}\le\alpha} v^{(p-3)/2}(1+(p-1)v)^{-p/2} exp\,(-z^2/2)\, dv\, dz$$

$$= \; c\int_0^\infty dv \int_{-\infty}^{\alpha\Big/\sqrt{1+(p-1)v}} v^{(p-3)/2}(1+(p-1)v)^{-p/2} exp\,(-z^2/2)\, dz\;.$$

Letting $z = t\Big/\sqrt{1+(p-1)v}$ we find

$$F_{u_{pq}}(\alpha) \;=\; c\int_{-\infty}^{\alpha} dt \int_0^\infty v^{(p-3)/2}(1+(p-1)v)^{-(p+1)/2} exp\left(-\frac{1}{2}\frac{t^2}{1+(p-1)v}\right) dv\;.$$

Letting $s = 1\Big/(1+(p-1)v)$ this can then be written

$$F_{u_{pq}}(\alpha) \;=\; \frac{1}{\sqrt{2\pi}}\,\frac{\Gamma(p/2)}{\Gamma((p-1)/2)}\int_{-\infty}^{\alpha} dt \int_0^1 (1-s)^{(p-3)/2} exp\left(-\frac{1}{2}t^2 s\right) ds\;.$$

Thus

$$f_{u_{pq}}(t) = \frac{1}{\sqrt{2}\pi}\frac{\Gamma(p/2)}{\Gamma((p-1)/2)}\int_0^1 (1-s)^{(p-3)/2} exp\left(-\frac{1}{2}t^2 s\right) ds\,.$$

Finally, letting $w = |t|\sqrt{1-s}$ we have

$$f_{u_{pq}}(t) = \frac{\sqrt{2}}{\pi}\frac{\Gamma(p/2)}{\Gamma((p-1)/2)} exp\left(-\frac{1}{2}t^2\right) t^{-p+1}\int_0^t w^{p-2} exp\left(\frac{1}{2}w^2\right) dw\,. \quad (2.4)$$

Since

$$\int_0^t w^{p-2} exp\left(\frac{1}{2}w^2\right) dw = exp\left(\frac{1}{2}t^2\right)\sum_{i=0}^{\lfloor\frac{p-3}{2}\rfloor}(-1)^i \prod_{j=0}^{i-1}(p-2j-3)\, t^{p-2i-3} +$$

$$(-1)^{\lfloor(p-1)/2\rfloor}(p-3)!!\left(\int_0^t exp\left(\frac{1}{2}x^2\right) dx\right)^{p-1-2\lfloor(p-1)/2\rfloor}$$

by integration by parts, where we define $0!! = (-1)!! = 1$, the desired result follows.

□

Similar to the derivation of the density function of u_{pq}, we first establish a relation between l_{pq} and the entries of X and then simplify it. Let $X = LU$ and $X^T = \tilde{L}\tilde{U}$ be the LU factorizations of X and X^T respectively. Set $\tilde{D} = \text{diag}(\tilde{u}_{11}, \cdots, \tilde{u}_{nn})$. Thus, $X^T = \tilde{L}\tilde{D}\tilde{D}^{-1}\tilde{U}$. So $X = \left(\tilde{D}^{-1}\tilde{U}\right)^T\left(\tilde{L}\tilde{D}\right)^T$. Note that $\left(\tilde{D}^{-1}\tilde{U}\right)^T$ is unit lower triangular and $\left(\tilde{L}\tilde{D}\right)^T$ upper triangular. By the uniqueness of the LU factorization of X, we have

$$L = \left(\tilde{D}^{-1}\tilde{U}\right)^T.$$

Hence

$$l_{pq} = \tilde{u}_{qp}/\tilde{u}_{qq}$$

for $1 \leq q < p \leq n$. By Lemma 2.1,

$$\tilde{u}_{qp} = x_{pq} - x_{*q}^T X_{q-1}^{-T} x_{p*}$$

and

$$\tilde{u}_{qq} = x_{qq} - x_{*q}^T X_{q-1}^{-T} x_{q*},$$

where

$$x_{p*} = (x_{p1}, \cdots, x_{pq-1})^T,$$

$$x_{q*} = (x_{q1}, \cdots, x_{qq-1})^T,$$

$$x_{*q} = (x_{1q}, \cdots, x_{q-1q})^T$$

and X_{q-1} is the $(q-1) \times (q-1)$ leading principal submatrix of X. We now let H be an $(q-1) \times (q-1)$ orthogonal matrix such that

$$x_{*q}^T H = (0, \cdots, 0, s) \equiv \eta^T$$

with $s \geq 0$. Then

$$l_{pq} = \frac{x_{pq} - \eta^T \left(X_{q-1}^T H\right)^{-1} x_{p*}}{x_{qq} - \eta^T \left(X_{q-1}^T H\right)^{-1} x_{q*}}$$

$$\equiv \frac{x_{pq} - \eta^T Y^{-1} x_{p*}}{x_{qq} - \eta^T Y^{-1} x_{q*}}.$$

As in the case of u_{pq}, all the entries in the above expression are mutually independent and s^2 is χ_{q-1}^2 while others are $N(0,1)$. Let

$$Y = QR$$

be the QR factorization of Y where R has positive diagonal elements. Then the expression can be reduced to

$$\begin{aligned} l_{pq} &= \frac{x_{pq} - \eta^T R^{-1} Q^T x_{p*}}{x_{qq} - \eta^T R^{-1} Q^T x_{q*}} \\ &\equiv \frac{x_{pq} - \eta^T R^{-1} w}{x_{qq} - \eta^T R^{-1} \mu} \qquad (2.5) \\ &= \frac{r_{q-1q-1} x_{pq} - s w_{q-1}}{r_{q-1q-1} x_{qq} - s \mu_{q-1}} . \end{aligned}$$

The entries x_{pq}, x_{qq}, w_i, μ_i and $r_{ij} (i < j)$ are $N(0,1)$ while s^2 is χ^2_{q-1} and r^2_{ii} is χ^2_{q-i}, where $i = 1, \cdots, q-1$, $j = 2, \cdots, q-1$. They are all independent.

Set

$$v = \frac{r_{q-1q-1}}{\sqrt{r^2_{q-1q-1} + s^2}} x_{pq} - \frac{s}{\sqrt{r^2_{q-1q-1} + s^2}} w_{q-1}$$

and

$$z = \frac{r_{q-1q-1}}{\sqrt{r^2_{q-1q-1} + s^2}} x_{qq} - \frac{s}{\sqrt{r^2_{q-1q-1} + s^2}} \mu_{q-1} ,$$

then $l_{pq} = v/z$. Since both v and z are $N(0,1)$ and they are independent, l_{pq} has Cauchy distribution.

Theorem 2.2 *Suppose X is an $n \times n$ Gaussian matrix and let $X = LU$ be the LU factorization of X. Then the density function of the (p,q)-th entry of L is*

$$f_{l_{pq}}(t) = \frac{1}{\pi} \frac{1}{1+t^2} \qquad (2.6)$$

where $-\infty < t < \infty$ and $1 \leq q < p \leq n$.

2.2.2 Probability of Small Pivot

In practice, if one of the pivot elements u_{pp} is zero or smaller in magnitude than a preset tolerance ϵ, Gaussian elimination will fail. In this section, we describe the probability of the occurrence of such a situation.

To make the statements below neatly, we use a shorthand notation here. For given $\epsilon > 0$ and $1 \leq p \leq n$, we define

$$E_{p,\epsilon} = \{X \in R^{n \times n} \,|\, |u_{pp}| < \epsilon\} .$$

Then the event that at least one u_{pp} has $|u_{pp}| < \epsilon$ is naturally denoted by $\bigcup_{p=1}^{n} E_{p,\epsilon}$.

Lemma 2.2 *Suppose X is an $n \times n$ Gaussian matrix and let $X = LU$ be the LU factorization of X. Given $\epsilon > 0$ and $1 \leq p \leq n$. Then*

$$Prob(E_{p,\epsilon}) \leq \frac{\sqrt{2}}{\pi} \frac{\Gamma(p/2)}{\Gamma((p+1)/2)} \epsilon .$$

Proof. From (2.4), we have

$$f_{u_{pq}}(t) \;\leq\; \frac{\sqrt{2}}{\pi} \frac{\Gamma(p/2)}{\Gamma((p-1)/2)} t^{-p+1} \int_0^t w^{p-2} dw \;=\; \frac{1}{\sqrt{2}\,\pi} \frac{\Gamma(p/2)}{\Gamma((p+1)/2)} ,$$

where $p \geq 2$, and from which the desired result follows.

For the case where $p = 1$, it is sufficient to note that

$$Prob(E_{1,\epsilon}) = Prob(|x_{11}| < \epsilon) = \frac{1}{\sqrt{2\pi}} \int_{-\epsilon}^{\epsilon} exp\left(-\frac{1}{2}t^2\right) dt . \quad \square$$

Theorem 2.3 *Suppose X is an $n \times n$ Gaussian matrix and let $X = LU$ be the LU factorization of X. Then*

$$Prob(\bigcup_{p=1}^{n} E_{p,\epsilon}) \leq \frac{\sqrt{2}}{\pi} \sum_{p=1}^{n} \frac{\Gamma(p/2)}{\Gamma((p+1)/2)} \epsilon. \tag{2.7}$$

Proof. Since $Prob(\bigcup_{p=1}^{n} E_{p,\epsilon}) \leq \sum_{p=1}^{n} Prob(E_{p,\epsilon})$, (2.7) follows by Lemma 2.2. □

The coefficient of ϵ is a rather slow-growing function of n. In fact, it is about 1800 even when $n = 10^6$. So, if ϵ is small enough, (2.7) will certainly give a satisfying bound for the desirable probability. Moreover, the right hand side of (2.7) is linear with ϵ.

2.2.3 Probability of Large Growth Factor

It is possible that the growth factors ρ_L and ρ_U, which are defined by (1.2) in §1.1 of Chapter 1, can be very large because small pivots can appear. In this section, we study the probabilities of the occurrence of large ρ_L and ρ_U and give probabilistic bounds on the sizes of them in the following Theorem.

Theorem 2.4 *Suppose X is an $n \times n$ Gaussian matrix and let $X = LU$ be the LU factorization of X. Then there exist numbers $1 > b > 0$ and $c > 0$, independent of n, such that*

$$Prob(\rho_U > r) \leq \frac{c}{r} n^{5/2} + \min\left(\frac{c}{r} n^{7/2}, \frac{1}{n}\right) + b^n$$

and

$$Prob(\rho_L > r) \;\leq\; \frac{c}{r} n^3$$

for any $r \geq 1$.

Proof. We first claim that there exists a $c_1 > 0$, independent of n, such that

$$Prob(\|U\|_\infty > r) \;\leq\; \frac{c_1}{r} n^{7/2}\,. \tag{2.8}$$

In fact, by (2.4), we have

$$f_{u_{pq}}(t) \;\leq\; \frac{\sqrt{2}}{\pi} \frac{\Gamma(p/2)}{\Gamma((p-1)/2)} \exp\left(-\frac{1}{2}t^2\right) t^{-1} \int_0^t \exp\left(\frac{1}{2}w^2\right) dw\,.$$

$$= \frac{\sqrt{2}}{\pi\sqrt{p}} \frac{\Gamma(p/2)}{\Gamma((p-1)/2)} \exp\left(-\frac{1}{2}t^2\right) t \int_0^t \exp\left(\frac{1}{2}w^2\right) dw \frac{\sqrt{p}}{t^2}\,.$$

Since

$$\lim_{k\to+\infty} \frac{1}{\sqrt{k}} \frac{\Gamma(k/2)}{\Gamma((k-1)/2)}$$

exists by Stirling's formula

$$\lim_{x\to+\infty} \frac{\Gamma(x+1)}{x^x exp\,(-x)\,\sqrt{2\pi x}} \;=\; 1\,,$$

and since

$$\lim_{t\to+\infty} t \exp\left(-\frac{1}{2}t^2\right) \int_0^t \exp\left(\frac{1}{2}w^2\right) dw \;=\; 1\,,$$

we can find a c_2 such that

$$f_{u_{pq}}(t) \;\leq\; c_2\sqrt{p}/t^2\,.$$

Therefore

$$\begin{aligned} Prob(\|U\|_\infty > r) &\leq \sum_{p=1}^{n}\sum_{q=p}^{n} P(|u_{pq}| > r/n) \\ &= \sum_{p=1}^{n}\sum_{q=p}^{n} \int_{|t|>r/n} f_{pq}(t)\,dt \\ &\leq \sum_{p=1}^{n}\sum_{q=p}^{n} \int_{|t|>r/n} \frac{c_2\sqrt{p}}{t^2} dt \\ &\leq \frac{c_2 c_3}{r} n^{7/2} \end{aligned}$$

for some $c_3 > 0$, independent of n. The existence of c_3 is due to the existence of the limit

$$\lim_{k\to+\infty} \frac{1}{k^{5/2}} \sum_{p=1}^{k} (k-p+1)\sqrt{p} = \int_0^1 (1-t)\sqrt{t}\,dt\,.$$

We set $c_1 = c_2 c_3$ and then (2.8) is proven. For proving the first inequality in the theorem, we note that the expected value μ and the variance σ^2 of the variable $x_1 \equiv \sum_{q=1}^{n} |x_{1q}|$ are

$$\mu = n\sqrt{\frac{2}{\pi}} \quad , \quad \sigma^2 = \left(1 - \frac{2}{\pi}\right) n\,.$$

Setting $\varepsilon = n\sqrt{1 - \frac{2}{\pi}}$ in Chebyshev's inequality [15, p.183]

$$Prob\,(|x_1 - \mu| \geq \varepsilon) \leq \frac{\sigma^2}{\varepsilon^2}\,,$$

we have

$$Prob\,(x_1 < nc_4) \leq \frac{1}{n}\,, \tag{2.9}$$

where $c_4 = \sqrt{\frac{2}{\pi}} - \sqrt{1 - \frac{2}{\pi}}$. Combining (2.8) and (2.9) we find

$$\begin{aligned}
Prob\left(\rho_U > r\right) &= Prob\left(\|U\|_\infty > r\|A\|_\infty\right) \\
&\leq Prob\left(\|U\|_\infty > rx_1\right) \\
&= Prob\left(\|U\|_\infty > rx_1\,,\, x_1 \geq nc_4\right) + Prob\left(\|U\|_\infty > rx_1\,,\, nc_4 > x_1 > 1\right) + \\
&\quad Prob\left(\|U\|_\infty > rx_1\,,\, x_1 \leq 1\right) \\
&\leq Prob\left(\|U\|_\infty > nrc_4\right) + \min\left(Prob\left(\|U\|_\infty > r\right)\,,\, Prob\left(x_1 < nc_4\right)\right) + \\
&\quad Prob\left(|x_{1q}| \leq 1,\, \forall\, 1 \leq q \leq n\right) \\
&\leq \frac{c_1}{c_4 r} n^{5/2} + \min\left(\frac{c_1}{r} n^{7/2}\,,\, \frac{1}{n}\right) + \left(\frac{1}{\sqrt{2\pi}} \int_{-1}^{1} exp\left(-\frac{1}{2}t^2\right) dt\right)^n .
\end{aligned}$$

Finally,

$$\begin{aligned}
Prob\left(\rho_L > r\right) &\leq \sum_{p=2}^{n} \sum_{q=1}^{p-1} Prob\left(|l_{pq}| > \frac{r-1}{n-1}\right) \\
&= \frac{1}{\pi} \sum_{p=2}^{n} \sum_{q=1}^{p-1} \int_{|t| > \frac{r-1}{n-1}} \frac{1}{1+t^2} dt \\
&\leq \frac{c_5}{r} n^3
\end{aligned}$$

for some c_5. □

2.2.4 Numerical Experiments

In this subsection, we present numerical results to support Theorems 2.1 - 2.4. All our calculations have been carried out in MATLAB 4.2c on SUN workstations.

In our first experiment, 595000 Gaussian matrices of dimension $n = 31$ were selected at random. Then Gaussian elimination was applied to each of the matrices and then statistics on the elements $l_{13,12}$, $l_{30,29}$, $u_{12,12}$ and $u_{31,31}$ were accumulated. The data are plotted in Figures 2.1 and 2.2 together with the corresponding functions indicated in Theorems 2.1 and 2.2. In order to make clearer the difference between Figures 2.1(a) and 2.1(b), we present them together in Figure 2.3(a).

The purpose of our second experiment is to test formula (2.7). Gaussian matrices of several dimensions n were selected at random, with the sample size varying. A few tolerances ϵ were used. The results are outlined in Table 2.1. The frequency column of the table provides the numbers of matrices which, in their LU factors, have at least one u_{pp} less than ϵ in magnitude. By comparing with the empirical probabilities, we conclude that the bound given in (2.7) is a fairly tight one.

Finally, if we set $r = n^{\alpha}, \alpha > 2.5$ for ρ_U and $\alpha > 3$ for ρ_L, in Theorem 2.4, then we can see that the probabilities $Prob(\rho_L > n^{\alpha})$ and $Prob(\rho_U > n^{\alpha})$ decrease with n increasing. In fact, empirically this is true even for smaller α, say, $\alpha > 1.5$ for both ρ_L and ρ_U, as illustrated in Figures 2.3(b) and 2.4. In this experiment, we chose sample sizes to be $968500, 365500$ and 98000 for $n = 25, 50$ and 100

n	ϵ	*Sample Size*	*Frequency*	*Empirical probability*	*Theoretical bound*
25	10^{-5}	10^5	5	5×10^{-5}	5.8586×10^{-5}
50	10^{-3}	10^4	90	0.009	0.0085
50	10^{-4}	10^4	8	8×10^{-4}	8.4853×10^{-4}
50	10^{-5}	10^4	0	0	8.4952×10^{-5}
50	10^{-5}	10^5	9	9×10^{-5}	8.4952×10^{-5}
75	10^{-3}	10^4	89	0.0089	0.0105
75	10^{-4}	10^4	8	8×10^{-4}	0.0011
75	10^{-5}	10^4	0	0	1.0519×10^{-4}
100	10^{-3}	10^4	115	0.0115	0.0122

Table 2.1: Probabilities of small pivot.

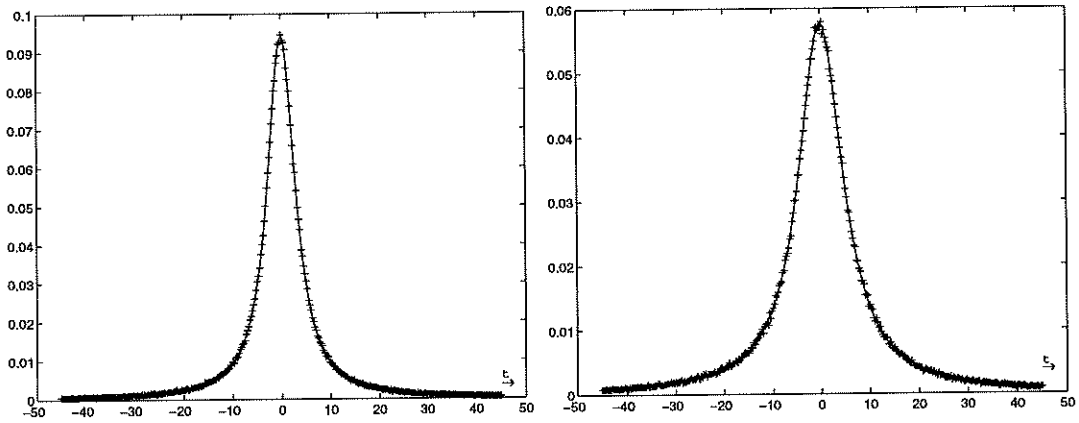

Figure 2.1: (a) Distribution of $u_{12,12}$: observed(+), predicted(-). (b) Distribution of $u_{31,31}$: observed(+), predicted(-).

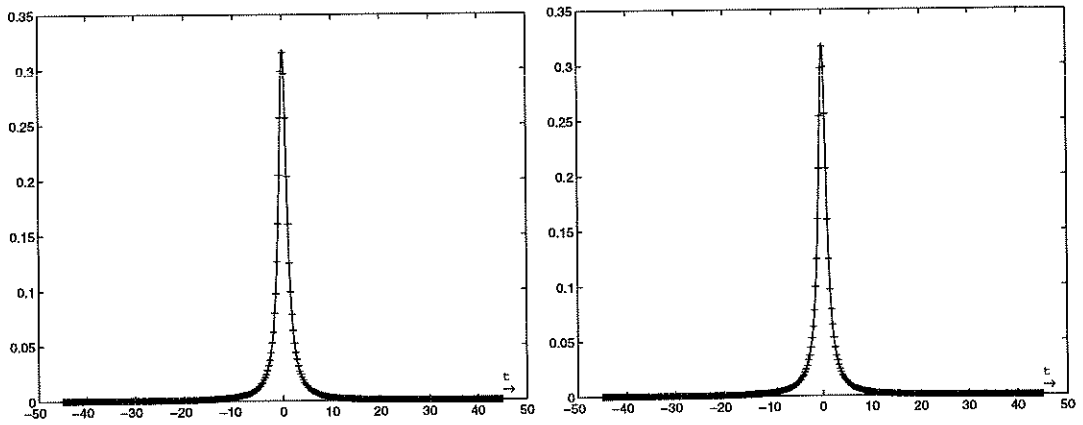

Figure 2.2: (a) Distribution of $l_{13,12}$: observed(+), predicted(-). (b) Distribution of $l_{30,29}$: observed(+), predicted(-).

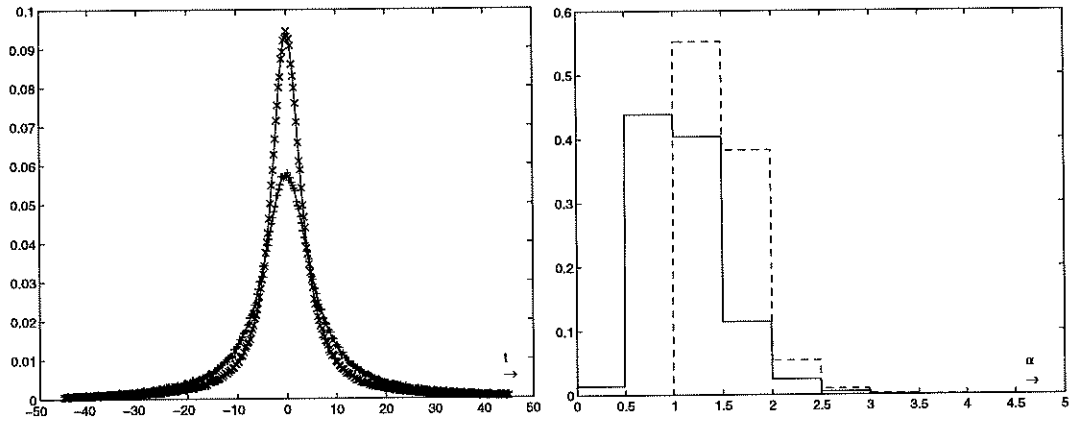

Figure 2.3: (a) Overlap of Figures 1(a) and 1(b). (b) Percentage frequency distributions of ρ_L (dashed) and ρ_U (solid). $n = 25$.

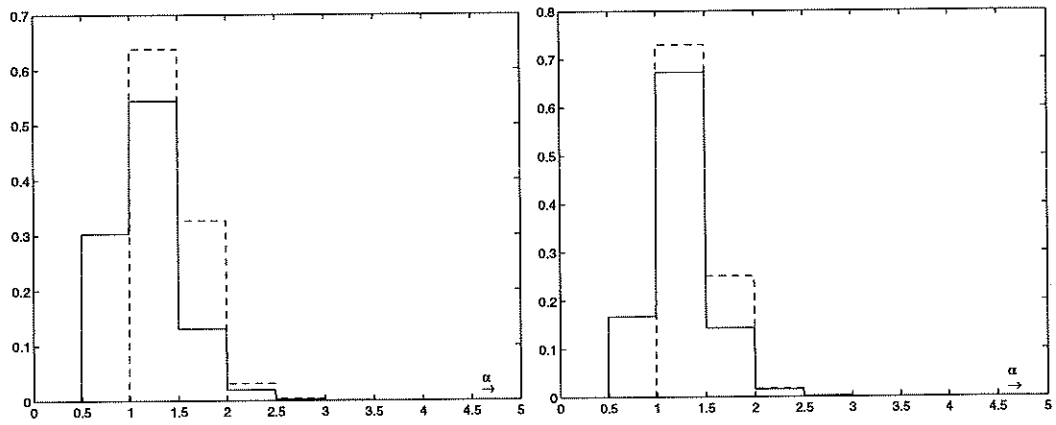

Figure 2.4: Percentage frequency distributions of ρ_L (dashed) and ρ_U (solid). (a) $n = 50$. (b) $n = 100$.

respectively. In each sample, we calculated ρ_L and ρ_U for each matrix X. Then the data of ρ_L and ρ_U were grouped into ten classes respectively. In the case of ρ_L, for example, the first class consists of matrices X with $n^0 \leq \rho_L < n^{0.5}$ and the second class with $n^{0.5} \leq \rho_L < n^1$, the third one with $n^1 \leq \rho_L < n^{1.5}$ and so on. The number of matrices in each class was then divided by the corresponding sample size to get the percentage frequency to the class. The distributions have been plotted in the form of histograms. Empirically, there is a tendency that $Prob(n \leq \rho_L < n^{1.5})$ and $Prob(n \leq \rho_U < n^{1.5})$ tend to one as n goes to infinity.

2.3 Complex Gaussian Elimination

All the results that were derived for the real case in §2.2.1 have straightforward analogues in the complex case.

2.3.1 Density Functions of u_{pq}

Let X be an $n \times n$ complex Gaussian matrix and let $X = LU$ be the complex LU factorization of X.

By applying the procedure of deriving (2.1) to the complex case now with unitary transforms replacing the corresponding orthogonal transforms, the following simple expression for the (p,q)-th $(2 \leq p \leq q)$ entry u_{pq} of U can be obtained,

$$u_{pq} = x_{pq} - \frac{sw_{p-1}}{r_{p-1p-1}},$$

where all the variables on the right-hand side are independent and $r^2_{p-1p-1}(r_{p-1p-1} \geq 0), s^2(s \geq 0), w_{p-1}$ and x_{pq} are $\chi^2_2, \chi^2_{2(p-1)}, \tilde{N}(0,1)$ and $\tilde{N}(0,1)$ respectively.

Set

$$v = \frac{1}{p-1}\frac{s^2}{r^2_{p-1p-1}} = \frac{s^2/2(p-1)}{r^2_{p-1p-1}/2}$$

and

$$z = \frac{1}{\sqrt{1+(p-1)v}}x_{pq} - \frac{\sqrt{(p-1)v}}{\sqrt{1+(p-1)v}}w_{p-1}\,,$$

then

$$u_{pq} = z\sqrt{1+(p-1)v}\,.$$

The variables v and z are again independent and they are $F_{2(p-1),2}$ and $\tilde{N}(0,1)$ respectively.

Theorem 2.5 *Suppose X is an $n \times n$ complex Gaussian matrix and let $X = LU$ be the LU factorization of X. Then the density functions of u^R_{pq}, u^I_{pq} and $|u_{pq}|$, the*

real part, imaginary part and absolute value of the (p,q)*-th* $(2 \le p \le q)$ *entry* u_{pq} *of* U *respectively, are*

$$f_{u_{pq}^R}(t) \;=\; f_{u_{pq}^I}(t) \;=\; \frac{p-1}{\sqrt{2\pi}} \sum_{i=0}^{\infty} \frac{(-1)^i}{2^i i!} B(p-1, i+3/2)\, t^{2i}\,,$$

where $-\infty < t < \infty$ *and*

$$f_{|u_{pq}|}(t) \;=\; (-1)^p 2^p (p-1)!\, t^{1-2p} \left(\exp\left(-\frac{t^2}{2}\right) \left(1 - p - \frac{t^2}{2}\right) - \sum_{i=0}^{p-1} (-1)^i \frac{1-p+i}{2^i i!} t^{2i} \right),$$

where $0 < t$.

Proof. Since v and z^R are independent and they are $F_{2(p-1),2}$ and $N(0,1)$ respectively, their joint density is

$$f(v,z) \;=\; f_v(v) f_{z^R}(z)$$

$$= \begin{cases} \dfrac{1}{\sqrt{2\pi}} (p-1)^p v^{p-2} (1+(p-1)v)^{-p} \exp(-z^2/2) & v > 0 \\ \\ 0 & \text{otherwise}\,, \end{cases}$$

where z^R is the real part of z. Thus the distribution function $F_{u_{pq}^R}(\alpha)$ of u_{pq}^R is given by

$$F_{u_{pq}^R}(\alpha) \;=\; \iint_{u_{pq}^R \le \alpha} f(v,z)\, dv dz$$

$$= \iint_{z\sqrt{1+(p-1)v} \le \alpha} \frac{1}{\sqrt{2\pi}} (p-1)^p v^{p-2} (1+(p-1)v)^{-p} \exp(-z^2/2)\, dv dz$$

$$= \frac{1}{\sqrt{2\pi}} (p-1)^p \int_0^{\infty} dv \int_{-\infty}^{\alpha \Big/ \sqrt{1+(p-1)v}} v^{p-2} (1+(p-1)v)^{-p} \exp(-z^2/2)\, dz\,.$$

Letting $z = t\Big/\sqrt{1+(p-1)v}$,

$$F_{u_{pq}^R}(\alpha) = \frac{1}{\sqrt{2\pi}}(p-1)^p \int_0^\infty dv \int_{-\infty}^{\alpha} v^{p-2}(1+(p-1)v)^{-p-1/2} \exp\left(-\frac{1}{2}\frac{t^2}{1+(p-1)v}\right) dt$$

$$= \frac{1}{\sqrt{2\pi}}(p-1)^p \int_{-\infty}^{\alpha} dt \int_0^\infty v^{p-2}(1+(p-1)v)^{-p-1/2} \exp\left(-\frac{1}{2}\frac{t^2}{1+(p-1)v}\right) dv\,.$$

Letting $s = \dfrac{1}{1+(p-1)v}$,

$$F_{u_{pq}^R}(\alpha) = \frac{1}{\sqrt{2\pi}}(p-1) \int_{-\infty}^{\alpha} dt \int_0^1 s^{1/2}(1-s)^{p-2} \exp\left(-\frac{1}{2}t^2 s\right) ds\,.$$

Thus

$$f_{u_{pq}^R}(t) = \frac{p-1}{\sqrt{2\pi}} \int_0^1 s^{1/2}(1-s)^{p-2} \exp\left(-\frac{1}{2}t^2 s\right) ds$$

$$= \frac{p-1}{\sqrt{2\pi}} \sum_{i=0}^{\infty} \frac{1}{i!}(-\frac{1}{2}t^2)^i \int_0^1 s^{i+1/2}(1-s)^{p-2} ds$$

$$= \frac{p-1}{\sqrt{2\pi}} \sum_{i=0}^{\infty} \frac{(-1)^i}{2^i i!} B(p-1, i+3/2)\, t^{2i}\,.$$

To obtain the density of $|u_{pq}|$, we notice that

$$|u_{pq}| = |z|\sqrt{1+(p-1)v}\,.$$

Set

$$w = z^{R2} + z^{I2}\,,$$

then

$$|u_{pq}| = \sqrt{w}\sqrt{1+(p-1)v}\,.$$

w is χ^2_2 and independent of v. Since the joint density of w and v is

$$f(w,v) = \begin{cases} \frac{1}{2}(p-1)^p v^{p-2}(1+(p-1)v)^{-p}\exp(-w/2) & v,w>0 \\ \\ 0 & \text{otherwise}, \end{cases}$$

the distribution function $F_{|u_{pq}|}(\alpha)$ of $|u_{pq}|$ is

$$\begin{aligned} F_{|u_{pq}|}(\alpha) &= \iint_{|u_{pq}|\le\alpha} f(w,v)dwdv \\ \\ &= \iint_{w(1+(p-1)v)\le\alpha^2} \frac{1}{2}(p-1)^p v^{p-2}(1+(p-1)v)^{-p}\exp(-w/2)dwdv \\ \\ &= \frac{1}{2}(p-1)^p \int_0^\infty dv \int_0^{\alpha^2/(1+(p-1)v)} v^{p-2}(1+(p-1)v)^{-p}\exp(-w/2)dw \\ \\ &= (p-1)^p \int_0^\infty v^{p-2}(1+(p-1)v)^{-p}\left(1-\exp\left(-\frac{1}{2}\frac{\alpha^2}{1+(p-1)v}\right)\right)dv\,. \end{aligned}$$

Letting $s = \dfrac{1}{1+(p-1)v}$,

$$F_{|u_{pq}|}(\alpha) = 1-(p-1)\int_0^1 (1-s)^{p-2}\exp\left(-\frac{\alpha^2}{2}s\right)ds\,.$$

Therefore,

$$\begin{aligned} f_{|u_{pq}|}(t) &= F'_{|u_{pq}|}(t) \\ \\ &= t(p-1)\int_0^1 (1-s)^{p-2}s\exp\left(-\frac{t^2}{2}s\right)ds \\ \\ &= t(p-1)\int_0^1 (1-s)^{p-2}s\sum_{i=0}^{\infty}(-1)^i\frac{t^{2i}}{2^i}\frac{1}{i!}s^i ds \end{aligned}$$

$$= t(p-1)\sum_{i=0}^{\infty}(-1)^i\frac{t^{2i}}{2^i}\frac{1}{i!}\int_0^1(1-s)^{p-2}s^{i+1}ds$$

$$= t(p-1)\sum_{i=0}^{\infty}(-1)^i\frac{t^{2i}}{2^i}\frac{1}{i!}B(p-1,i+2)$$

$$= t(p-1)\sum_{i=0}^{\infty}(-1)^i\frac{t^{2i}}{2^i}\frac{1}{i!}\frac{(p-2)!(i+1)!}{(i+p)!}$$

$$= t(p-1)!\sum_{i=0}^{\infty}(-1)^i\frac{t^{2i}}{2^i}\frac{i+1}{(i+p)!}.$$

Since

$$\sum_{i=0}^{\infty}x^i\frac{i+1}{(i+p)!} = \frac{d}{dx}\left(\sum_{i=0}^{\infty}\frac{x^{i+1}}{(i+p)!}\right)$$

$$= \frac{d}{dx}\left(x^{1-p}\sum_{i=0}^{\infty}\frac{x^{i+p}}{(i+p)!}\right)$$

$$= \frac{d}{dx}\left(x^{1-p}\left(\exp(x)-\sum_{i=0}^{p-1}\frac{x^i}{i!}\right)\right)$$

$$= x^{-p}\exp(x)(1-p+x)-\sum_{i=0}^{p-1}\frac{i-p+1}{i!}x^{i-p},$$

we have

$$f_{|u_{pq}|}(t) = (-1)^p2^p(p-1)!\,t^{1-2p}\left(\exp\left(-\frac{t^2}{2}\right)\left(1-p-\frac{t^2}{2}\right)-\sum_{i=0}^{p-1}(-1)^i\frac{1-p+i}{2^i i!}t^{2i}\right). \quad \square$$

2.3.2 Density Functions of l_{pq}

The expressions for the density functions of l_{pq} are far simpler than those of u_{pq} as shown in the real case. The situation is the same in the complex case. A similar derivation of (2.5) with unitary transforms replacing orthogonal transforms gives

$$l_{pq} = \frac{r_{q-1q-1}x_{pq} - sw_{q-1}}{r_{q-1q-1}x_{qq} - s\mu_{q-1}},$$

where the variables on the right-hand side are independent and $r^2_{q-1q-1}(r_{q-1q-1} > 0), s^2(s > 0)$ are $\chi^2_2, \chi^2_{2(q-1)}$ respectively and others are $\tilde{N}(0,1)$.

Set

$$v = \frac{r_{q-1q-1}}{\sqrt{r^2_{q-1q-1} + s^2}} x_{pq} - \frac{s}{\sqrt{r^2_{q-1q-1} + s^2}} w_{q-1}$$

and

$$z = \frac{r_{q-1q-1}}{\sqrt{r^2_{q-1q-1} + s^2}} x_{qq} - \frac{s}{\sqrt{r^2_{q-1q-1} + s^2}} \mu_{q-1},$$

then $l_{pq} = v/z$. Both v and z are $\tilde{N}(0,1)$ and independent. Thus,

$$l_{pq} = \frac{v^R + iv^I}{z^R + iz^I}$$

$$= \frac{v^R z^R + v^I z^I + i\left(v^I z^R - v^R z^I\right)}{z^{R2} + z^{I2}}$$

and therefore,

$$l^R_{pq} = \frac{v^R z^R + v^I z^I}{z^{R2} + z^{I2}}$$

$$= \frac{v^R z^R + v^I z^I}{\sqrt{z^{R2} + z^{I2}}} \Big/ \sqrt{z^{R2} + z^{I2}}$$

$$\equiv \xi/\sqrt{\eta}$$

$$= \frac{1}{\sqrt{2}}\frac{\xi}{\sqrt{\eta/2}}$$

$$\equiv \frac{1}{\sqrt{2}}\zeta\,.$$

η and ξ are χ^2_2 and $N(0,1)$ respectively and independent. Hence ζ has the student's t distribution with 2 degrees of freedom. Thus,

$$f_{l^R_{pq}}(t) = \frac{1}{2}(1+t^2)^{-3/2}\,,$$

where $-\infty < t < \infty$. Moreover,

$$|l_{pq}| = |v|/|z| = \sqrt{\frac{|v|^2/2}{|z|^2/2}} \equiv \sqrt{\xi}\,,$$

where ξ has $F_{2,2}$ distribution. Hence

$$f_{|l_{pq}|}(t) = \begin{cases} \dfrac{2t}{(1+t^2)^2} & t > 0 \\ \\ 0 & \text{otherwise}\,. \end{cases}$$

We summerize the above results about l_{pq} in the following theorem.

Theorem 2.6 *Suppose X is an $n \times n$ complex Gaussian matrix and let $X = LU$ be the LU factorization of X. Then the density functions of l^R_{pq}, l^I_{pq} and $|l_{pq}|$, the real part, imaginary part and absolute value of the (p,q)-th $(1 \le q < p \le n)$ entry l_{pq} of L respectively, are*

$$f_{l^R_{pq}}(t) = f_{l^I_{pq}}(t) = \frac{1}{2}(1+t^2)^{-3/2}\,,$$

where $-\infty < t < \infty$ *and*

$$f_{|l_{pq}|}(t) = \frac{2t}{(1+t^2)^2},$$

where $0 < t$.

Figures 2.5-2.7 plot the empirical densities of some entries of L and U and their corresponding predicted densities in Theorems 2.5 and 2.6.

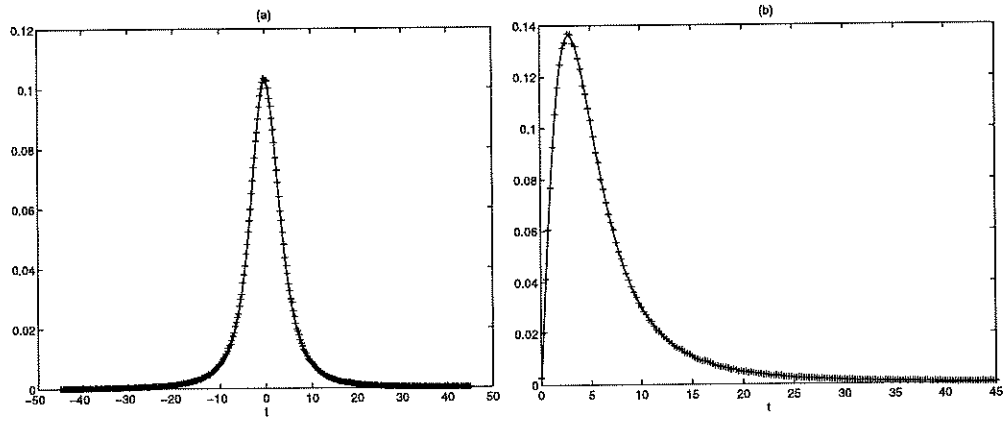

Figure 2.5: (a) Distribution of $u^R_{12,12}$: observed(+), predicted(-). (b) Distribution of $|u_{12,12}|$: observed(+), predicted(-).

2.4 Application

In some kinds of parallel computations, such as in a systolic system [37], Gaussian elimination without pivoting is preferred to GE with pivoting because the latter version is difficult to implement due to data movement which can not be determined a priori and, as a result, leads to high overhead both in time and

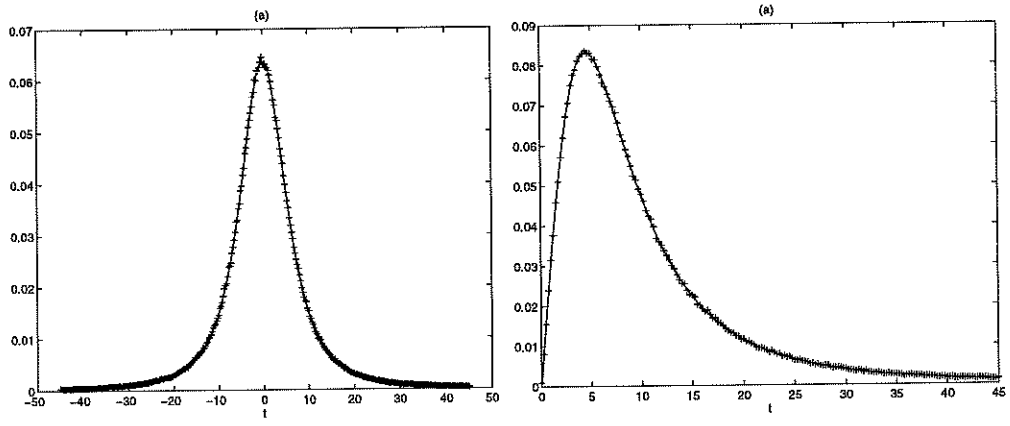

Figure 2.6: (a) Distribution of $u^R_{31,31}$: observed(+), predicted(-). (b) Distribution of $|u_{31,31}|$: observed(+), predicted(-).

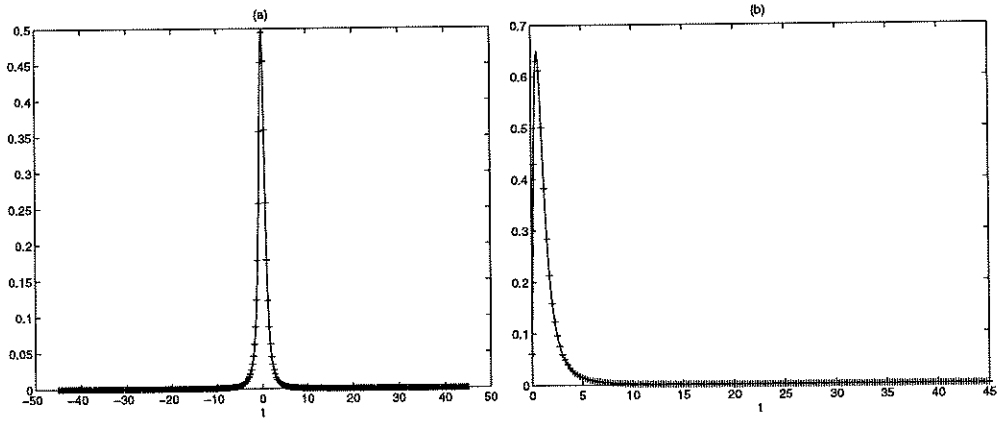

Figure 2.7: (a) Distribution of $l^R_{13,12}$: observed(+), predicted(-). (b) Distribution of $|l_{13,12}|$: observed(+), predicted(-).

memory [40]. However, as we mentioned earlier, there are two difficulties about the implementation of GE without pivoting, that is, breakdown and instability. Recently, Lê, Parker and Pierce (LPP) [40, 42, 43] suggested an effective method to overcome these difficulties in which they first randomized the original matrix and then applied GE without pivoting to the resulting matrix. More precisely, let A be an $n \times n$ matrix and V a random $n \times n$ matrix from some special class. Their basic idea is that they perform GE without pivoting on the product matrix VA instead of A. Experimental results have shown that this indead greatly reduces the probability of breakdown and improves the potential ill-conditioning of A.

In this section, we describe an alternative method to that of LPP's to cure breakdown and instability in GE without poviting. Based on eight random classes of matrices, Trefethen and Schreiber [57] observed by experiments that, after the first few steps of GE with partial (or complete) pivoting, the remaining matrix elements are approximately normally distributed, regardless of what class the matrix comes from. On the other hand, as we have seen in §2.2, if GE without pivoting is applied to a Gaussian matrix, then the probability of the occurence of a pivot less than ϵ in magnitude is $O(\epsilon)$ and $Prob(n \leq \rho_L < n^{1.5}) \approx 1$, $Prob(n \leq \rho_U < n^{1.5}) \approx 1$ when n is large. All of these observations suggest that we may use GE in such a way that, in the first few steps, we use GE with partial (or complete) pivoting and then use the version without pivoting (or with pairwise, neighbor pivoting). The goal of the use of GE with pivoting in the early steps is to make the elements ap-

proximately normally distributed so that the chances of breakdown and instability could be reduced when a simpler form of GE is performed later on. We have done some experiments on this idea based on two random classes of matrices:

- Class 1: independent elements from the discrete distribution with $P(-1) = P(1) = 0.5$
- Class 2: independent elements from the discrete distribution with $P(0) = P(1) = 0.5$

Since both the results from these two classes are analogous, we will only show those for Class 1 below. The scheme we adopted in our experiments is the following. We perform GE with complete pivoting on A in the first k steps to get the decomposition,

$$PAQ = \begin{bmatrix} L_{11} & 0 \\ L_{21} & I \end{bmatrix} \begin{bmatrix} U_{11} & U_{12} \\ 0 & U_{22} \end{bmatrix},$$

where P and Q are permutation matrices, U_{11} and L_{11} are $k \times k$ upper and unit lower triangular matrices respectively and I is the $(n-k)\times(n-k)$ identity matrix. Based on the observations of Trefethen and Schreiber, we may expect that the elements in U_{22} are approximately normally distributed. Then we perform GE without pivoting on U_{22} to get the LU decomposition of U_{22},

$$U_{22} = \tilde{L}_{22}\tilde{U}_{22}\,.$$

Thus we obtain the following form of LU decomposition for A,

$$PAQ = \begin{bmatrix} L_{11} & 0 \\ L_{21} & \tilde{L}_{22} \end{bmatrix} \begin{bmatrix} U_{11} & U_{12} \\ 0 & \tilde{U}_{22} \end{bmatrix}.$$

Although the entries of U_{22} are not independent, we may still expect, according to the theory we derived in earlier sections, that the probability of the occurrence of small pivots and large growth factors is very small in the LU factorization of U_{22}.

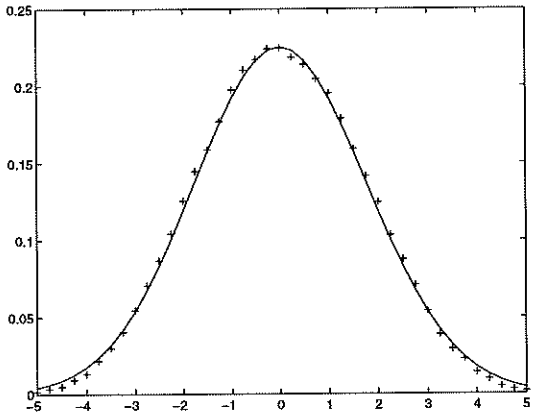

Figure 2.8: Observed distribution (+) of the elements of U_{22} vs. the $N(0, 1.77)$ distribution (-). $k = 20, n = 51$.

Figure 2.8 plots the empirical distribution of the elements of U_{22} together with the normal distribution $N(0, 1.77)$, where we chose $k = 20$ and $n = 51$. We can see that the elements of U_{22} are indeed almost normally distributed, even though it is obvious that they are correlated. Based on this observation, we can expect that the elements of $\tilde{L}_{22}$ and $\tilde{U}_{22}$ would obey the distributions described in Theorems 2.1 and 2.2. In fact, Figures 2.9 and 2.10 illustrate this point, where we plotted the empirical distributions of the elements $\tilde{l}_{13,12}, \tilde{l}_{30,29}, \tilde{u}_{12,12}$ and $\tilde{u}_{31,31}$ of $\tilde{L}_{22}$ and

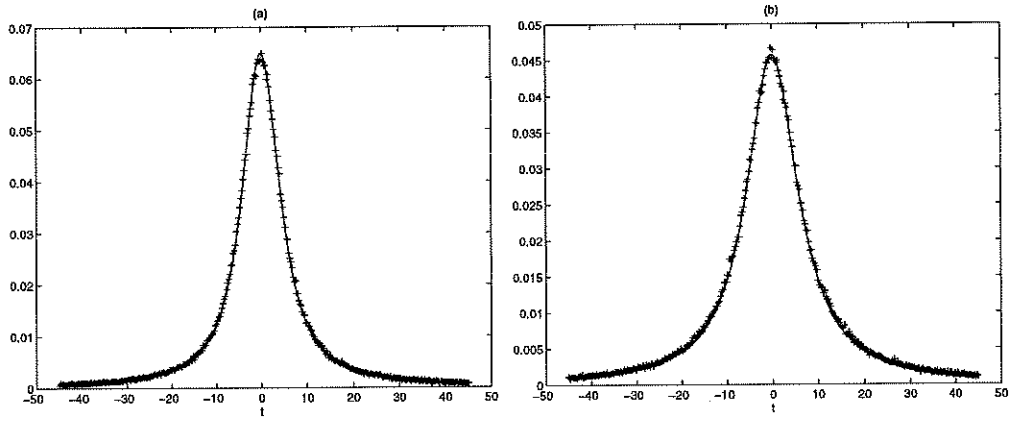

Figure 2.9: (a) Observed distribution (+) of the $(12, 12)$-th entry of $\tilde{U}_{22}$ vs. the predicted distribution $0.7 f_{u_{12,12}}(0.7t)$ (-). (b) Observed distribution (+) of the $(31, 31)$-th entry of $\tilde{U}_{22}$ vs. the predicted distribution $0.8 f_{u_{31,31}}(0.8t)$ (-). The functions $f_{u_{12,12}}(t)$ and $f_{u_{31,31}}(t)$ are defined in Theorem 2.1.

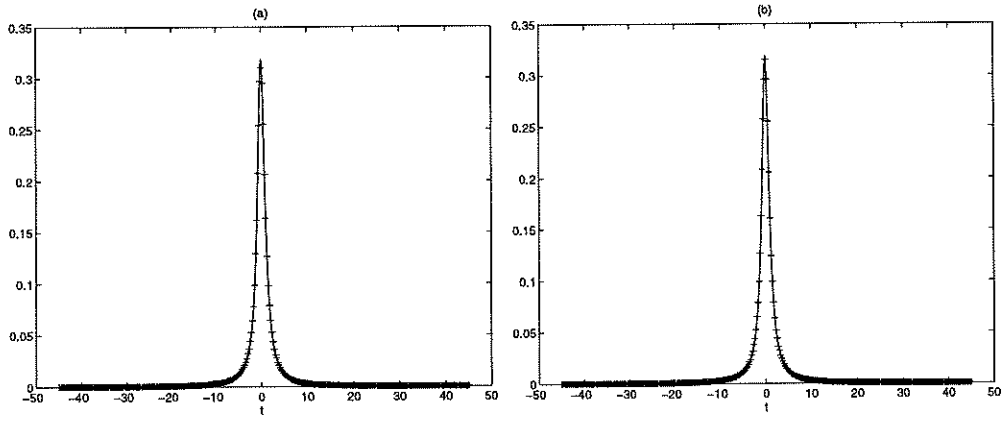

Figure 2.10: (a) Observed distribution (+) of the $(13, 12)$-th entry of $\tilde{L}_{22}$ vs. the predicted distribution $f_{l_{13,12}}(t)$ (-). (b) Observed distribution (+) of the $(30, 29)$-th entry of $\tilde{L}_{22}$ vs. the predicted distribution $f_{l_{30,29}}(t)$ (-). The functions $f_{l_{13,12}}(t)$ and $f_{l_{30,29}}(t)$ are defined in Theorem 2.2.

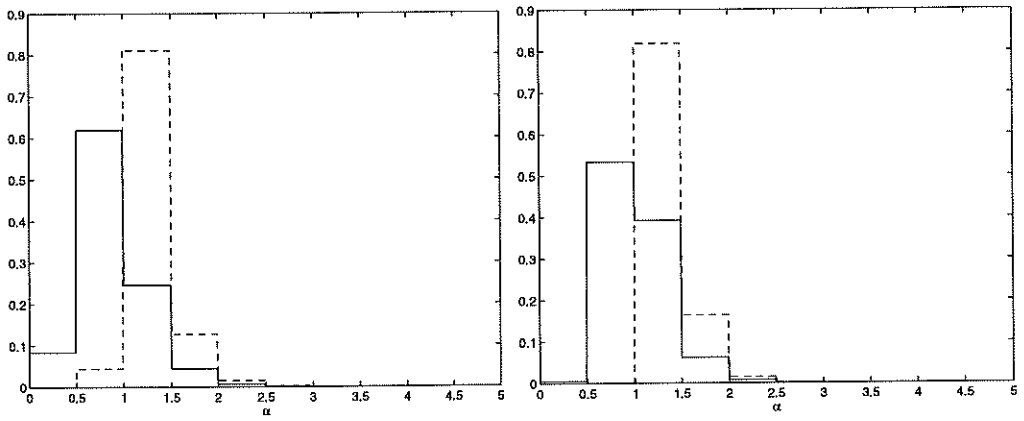

Figure 2.11: Percentage frequency distributions of $\rho_{\tilde{L}_{22}}$ (dashed) and $\rho_{\tilde{U}_{22}}$ (solid). (a) $k = 20, n = 45$. (b) $k = 20, n = 70$. See Figures 2.3 and 2.4 for the meaning of α.

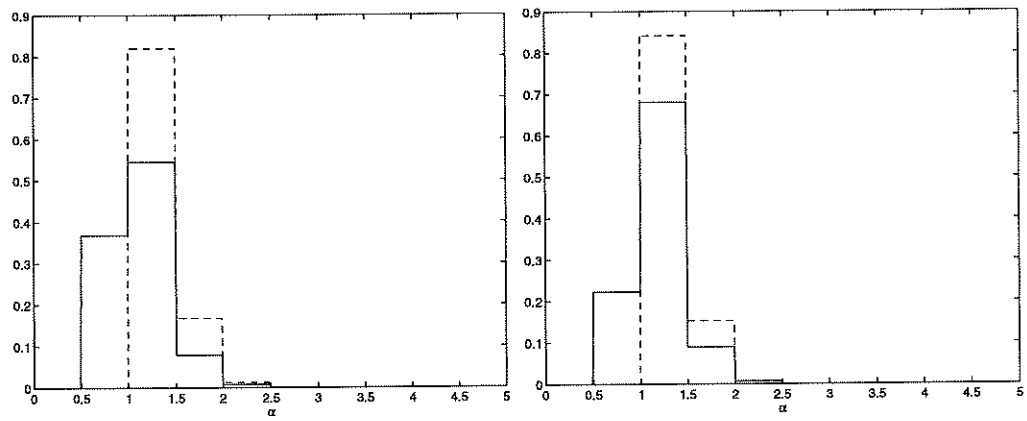

Figure 2.12: Percentage frequency distributions of $\rho_{\tilde{L}_{22}}$ (dashed) and $\rho_{\tilde{U}_{22}}$ (solid). (a) $k = 20, n = 120$. (b) $k = 20, n = 220$. See Figures 2.3 and 2.4 for the meaning of α.

	$n = 64$		$n = 128$		$n = 256$	
k	B	EP	B	EP	B	EP
0	12359	9.8872×10^{-1}	12378	9.9024×10^{-1}	12368	9.8944×10^{-1}
5	9273	7.4184×10^{-1}	9408	7.5264×10^{-1}	9324	7.4592×10^{-1}
10	926	7.408×10^{-2}	936	7.488×10^{-2}	935	7.48×10^{-2}
15	21	1.68×10^{-3}	14	1.12×10^{-3}	19	1.52×10^{-3}
20	1	8×10^{-5}	2	1.6×10^{-4}	0	0
25	0	0	0	0	0	0

Table 2.2: 12500 matrices from Class 1 were selected in this experiment. B: the number of matrices which make GE without pivoting performed on U_{22} breakdown; EP: the empirical probability of breakdown when GE without pivoting performed on U_{22}

$\tilde{U}_{22}$ respectively versus their corresponding (scaled) predicted distributions. Both the observed and predicted distributions match very well in these two figures.

Table 2.2 shows some information on the situation of breakdown. In this experiment, 12500 matrices were selected from Class 1 for each matrix order n. From the table, we can see that the probability of breakdown is almost the same as n varies for each fixed k while it decreases as k increases for n fixed. If this were true for the general case, then we would be able to avoid breakdown in GE without pivoting with little data movement by choosing small k, regardless of how large the matrix size is.

In the case of the Gaussian matrix, we have seen that the probabilities $Prob(m \leq \rho_L < n^{1.5})$ and $Prob(n \leq \rho_U < n^{1.5})$ tend to one as n goes to infinity. The situation seems to be the same for the Class 1 matrices as shown in Figures 2.11 and 2.12. More precisely, for k fixed, we may have

$$\lim_{n \to +\infty} Prob(n \leq \rho_{\tilde{L}_{22}} < n^{1.5}) = 1$$

and

$$\lim_{n \to +\infty} Prob(n \leq \rho_{\tilde{U}_{22}} < n^{1.5}) = 1 \,,$$

where $\rho_{\tilde{L}_{22}} \equiv \|\tilde{L}_{22}\|_\infty$ and $\rho_{\tilde{U}_{22}} \equiv \|\tilde{U}_{22}\|_\infty / \|U_{22}\|_\infty$.

We end this section with the conclusion that the method we discussed here may be a good choice in solving GE related problems to avoid breakdown, instability and data movements in large quantities. We can not say any more on this idea at this point because we only did a few experiments on some special classes. We mention it just for the references of the GE users.

CHAPTER 3

Preserving Symmetry in Preconditioned Krylov Subspace Methods

We consider the problem of solving a linear system

$$Ax = b$$

by a preconditioned Krylov subspace method when A is nearly symmetric and when the system is preconditioned by a symmetric positive definite matrix M. In the symmetric case, one can recover symmetry by using M-inner products in the conjugate gradient (CG) algorithm. This idea can also be used in the nonsymmetric case, and near symmetry can be preserved similarly. Like CG, the new algorithms are mathematically equivalent to split preconditioning, but do not require M to be factored. Better robustness in a specific sense can also be observed. When combined with truncated versions of iterative methods, tests show that this is more effective than the common practice of forfeiting near-symmetry altogether.

This chapter is organized as follows. In §1, it is shown how alternative inner products may be used to preserve symmetry in GMRES. §2 considers the use of truncated iterative methods when the preconditioned system is close to being symmetric. This has been hypothesized by many authors, for example, Axelsson [5] and Meyer [41]. In §3 we consider the symmetrically preconditioned Bi-CG

algorithm. §4 shows the results of tests on a Navier-Stokes problem. This problem is parameterized by the Reynolds number, and thus nearness to symmetry. We conclude this chapter in §5.

3.1 Symmetric preconditioning in GMRES

When A is nearly symmetric, split preconditioning may be used to preserve the original degree of symmetry. Alternatively, left-preconditioning with the M-inner product, or right-preconditioning with the M^{-1}-inner product may be used. We refer to these latter two preconditionings as *symmetric.*

Consider first the left-preconditioned version of the Arnoldi algorithm based on the classical Gram-Schmidt process, with preconditioner M, which we assume to be SPD.

ALGORITHM 2.1. **Left-Preconditioned Arnoldi-Classical Gram-Schmidt**

1. *Choose a vector* v_1 *of norm* 1.
2. *For* $j = 1, 2, \cdots, m$
3. $\quad w_j = Av_j;$
4. $\quad z = M^{-1}w_j;$
5. $\quad$ *Compute* $h_{ij} = (z, v_j)$ *for* $i = 1, 2, \cdots, j;$
6. $\quad z = z - \sum_{i=1}^{j} h_{ij}v_i;$
7. $\quad h_{j+1,j} = \|z\|_2;$
8. $\quad$ *If* $h_{j+1,j} = 0$*then stop;*
9. $\quad v_{j+1} = z/h_{j+1,j};$
10. *End*

To implement this procedure using M-inner products, we would first compute the scalars h_{ij} in Line 5,

$$h_{ij} = (z_j, v_i)_M = (Mz_j, v_i) = (w_j, v_i) \ , \quad i = 1, \ldots, j. \tag{3.1}$$

Then we would modify the vector z_j to obtain the next Arnoldi vector (before normalization),

$$\hat{z}_j = z_j - \sum_{i=1}^{j} h_{ij} v_i. \tag{3.2}$$

To complete the orthonormalization step we must normalize the final $\hat{z}_j$. Because of the M-orthogonality of $\hat{z}_j$ versus all previous v_i's we observe that

$$(\hat{z}_j, \hat{z}_j)_M = (z_j, \hat{z}_j)_M = (M^{-1} w_j, \hat{z}_j)_M = (w_j, \hat{z}_j). \tag{3.3}$$

Thus, the desired M-norm can be computed according to (3.3) and then computing

$$h_{j+1,j} = (\hat{z}_j, w_j)^{1/2} \quad \text{and} \quad v_{j+1} = \hat{z}_j / h_{j+1,j}. \tag{3.4}$$

One potentially serious difficulty with the above procedure is that the inner product $(\hat{z}_j, \hat{z}_j)_M$ as computed by (3.3) may be negative in the presence of round-off. Indeed, loss of M-orthogonality between z_j and the previous v_i's causes the first equality in (2.3) to be invalid in the presence of rounding, which may result in a negative inner product $(w_j, \hat{z}_j)$.

There are two remedies. First, we can compute this M-norm explicitly at the expense of an additional matrix-vector multiplication with M, i.e., from

$$(\hat{z}_j, \hat{z}_j)_M \ = \ (M\hat{z}_j, \hat{z}_j).$$

As was pointed out earlier, this is undesirable, since the operator M is often not available explicitly. Indeed in many cases, only the preconditioning operation M^{-1} is available from a sequence of operations, as is the case for multigrid preconditioning. Another difficulty with computing h_{ij} with (3.1) is that it is not immediately amenable to a modified Gram-Schmidt implementation. Indeed, consider the first step of a hypothetical modified Gram-Schmidt step, which consists of M-orthonormalizing z against v_1,

$$h_{i1} = (z, v_1)_M \ , \qquad \hat{z} = z - h_{i1} v_1 .$$

As was observed, the inner product $(z, v_1)_M$ is equal to (w, v_1) which is computable. Now we need $M\hat{z}$ to compute $(z - h_{i1}v_1, v_i)_M$ in the modified Gram-Schmidt process. However, no such vector is available, and we can only compute the $(z, v_i)_M$ of classical Gram-Schmidt.

An alternative is to save the set of vectors Mv_i (again, not computed by multiplying by M) which would allow us to accumulate inexpensively both the vector $\hat{z}_j$ and the vector $\hat{w}_j$ via the relation

$$\hat{w}_j \equiv M\hat{z}_j = w_j - \sum_{i=1}^{j} h_{ij} M v_i,$$

which is obtained from (3.2). Now the inner product $(\hat{z}_j, \hat{z}_j)_M$ is given by

$$(\hat{z}_j, \hat{z}_j)_M = (M^{-1}\hat{w}_j, M^{-1}\hat{w}_j)_M = (M^{-1}\hat{w}_j, \hat{w}_j).$$

In this form, this inner product is guaranteed to be nonnegative as desired. This leads to the following algorithm.

ALGORITHM 2.2. **Arnoldi-Classical Gram-Schmidt and M-inner products**

1. *Choose a vector* w_1 *such that* $v_1 = M^{-1}w_1$ *has* M*-norm 1.*
2. *For* $j = 1, 2, \cdots, m$
3. $\quad w = Av_j$;
4. $\quad$ *Compute* $h_{ij} = (w, v_i)$ *for* $i = 1, 2, \cdots, j$;
5. $\quad \hat{w} = w - \sum_{i=1}^{j} h_{ij} w_i$;
6. $\quad z = M^{-1}\hat{w}$;
7. $\quad h_{j+1,j} = (z, \hat{w})^{1/2}$; *If* $h_{j+1,j} = 0$ *then stop*;
8. $\quad w_{j+1} = \hat{w}/h_{j+1,j}$;
9. $\quad v_{j+1} = z/h_{j+1,j}$;
10. *End*

As is noted, the above algorithm requires that we save two sets of vectors: the v_j's and the w_i's. The v_i's form the needed Arnoldi basis, and the w_i's are required when computing the vector $\hat{w}_j$ in Line 5. If we do save these two sets of vectors we can also easily formulate the algorithm with the modified Gram-Schmidt version of the Arnoldi procedure.

The situation for right-preconditioning with the M^{-1}-inner product is much simpler, mainly because $M^{-1}z$ is available when z needs to be normalized in the M^{-1} norm. It is only necessary to note that $M^{-1}z$ is normally computed at the next iteration in the standard Arnoldi algorithm. Again, both the v's and the w's need to be saved, where $w_j = M^{-1}v_j$ in this case.

The additional storage of the w's, however, makes the algorithm naturally *flexible*, i.e., it accommodates the situation where M varies at each step as when $M^{-1}v$ is the result of some unspecified computation. If M^{-1} is not a constant operator, then a basis for the right-preconditioned Krylov subspace cannot be constructed from the v's alone. However, the vectors $w_j = M_j^{-1}v_j$ do form a basis for this subspace, where M_j^{-1} denotes the preconditioning operation at the j-th step. The use of this extra set of vectors is exactly how the standard flexible variant of GMRES is implemented [49].

We can now write preconditioned GMRES algorithms using these implementations of alternative inner products in the Gram-Schmidt process. In the left-preconditioned case, we can minimize the M-norm of the preconditioned residual vector $M^{-1}(b - Ax)$ by simply minimizing the 2-norm of the projected system in the standard GMRES algorithm. To be formal, we state the following theorem without proof.

Theorem 3.1 *The approximate solution x_m obtained from the left-preconditioned GMRES algorithm with M-inner products minimizes the residual M-norm $\|M^{-1}(b - Ax)\|_M$ over all vectors of the affine subspace $x_0 + K_m$ in which*

$$K_m = span\{z_0, M^{-1}Az_0, \ldots, (M^{-1}A)^{m-1}z_0\} \tag{3.5}$$

where $z_0 = M^{-1}r_0$. Also, the approximate solution x_m obtained from the right-preconditioned GMRES algorithm with M^{-1}-inner products minimizes the residual M^{-1}-norm $\|b - Ax\|_{M^{-1}}$ over the same affine subspace.

We can show that both left and right symmetric preconditioning are mathematically equivalent to split preconditioning. All that must be noticed is that minimizations of

$$\|L^{-1}(b - Ax)\|_2 = \|b - Ax\|_{M^{-1}} = \|M^{-1}(b - Ax)\|_M$$

are the same, and that the minimizations are over the same subspace in each of the left, right, and split preconditioning options [50, §9.3.4]. We emphasize in particular that it is the split preconditioned residual that is minimized in all three algorithms.

3.2 Truncated iterative methods

Truncated iterative methods are an alternative to restarting, when the number of steps required for convergence is large and the computation and storage of the Krylov basis becomes excessive. When A is exactly symmetric, a three-term recurrence governs the vectors in the Arnoldi process, and it is only necessary to orthogonalize the current Arnoldi vector against the previous two vectors. If A is nearly symmetric, an incomplete orthogonalization against a small number of previous vectors may be advantageous over restarted methods. The advantage here may offset the cost of maintaining the extra set of vectors to maintain the initial degree of symmetry. The incomplete Arnoldi procedure outlined below stores only the previous k Arnoldi vectors, and orthogonalizes the new vectors against them.

It differs from the full Arnoldi procedure only in Line 4, which would normally be a loop from 1 to j. It can be considered to be the full Arnoldi procedure when k is set to infinity.

ALGORITHM 3.1. **Incomplete Arnoldi Procedure**

1. *Choose a vector* v_1 *of norm 1.*
2. *For* $j = 1, 2, \cdots, m$
3. $\quad w = Av_j$;
4. $\quad$ *For* $i = \max\{1, j-k+1\}, \cdots, j$
5. $\quad\quad h_{ij} = (w, v_i)$;
6. $\quad\quad w = w - h_{ij}v_i$;
7. $\quad$ *End*;
8. $\quad h_{j+1,j} = \|z\|_2$;
9. $\quad$ *If* $h_{j+1,j} = 0$ *then stop*;
10. $\quad v_{j+1} = w/h_{j+1,j}$;
11. *End*

The truncated version of GMRES uses this incomplete Arnoldi procedure and is called Quasi-GMRES [13]. The practical implementation of this algorithm allows the solution to be updated at each iteration, and is thus called a 'direct' version, or DQGMRES [51].

To suggest that truncated iterative methods may be effective in cases of near symmetry, we study the asymptotic behavior of the iterates of DQGMRES as the coefficient matrix A varies from nonsymmetry to (skew) symmetry. We first

decompose A as

$$A = S + B$$

in which S is symmetric or skew symmetric, and set

$$\varepsilon = \|B\|_2.$$

We will first establish asymptotic relations among the variables in the incomplete and full Arnoldi procedures. Then we will apply the incomplete procedure to A, and the full procedure to S, using the superscripts I and F to distinguish between the variables appearing in the two procedures. (Note that since S is (skew) symmetric, the full procedure on S is the same as the incomplete procedure with $k \geq 2$.)

Moreover, if we denote the degree of the minimal polynomial of v_1^F with respect to S by ν, then $h^F_{\nu+1,\nu} = 0$ and $h^F_{j+1,j} \neq 0$ for $1 \leq j < \nu$. In the proof of the following lemma, we also use $\hat{v}_j^I$ and $\hat{v}_j^F$ to denote the vectors w^I and w^F obtained at the end of Line 7 in the incomplete and complete Arnoldi procedures.

Lemma 3.1 *Assume the truncation parameter $k \geq 2$. If $v_1^I = v_1^F + O(\varepsilon)$, then*

$$h^I_{ij} = h^F_{ij} + O(\varepsilon)\,, \qquad v_j^I = v_j^F + O(\varepsilon)$$

where $1 \leq j \leq \nu$ and $\max\{1, j-k+1\} \leq i \leq j+1$.

Proof. The proof is by induction on the index j. By Lines 5 and 6 of the Arnoldi procedure,

$$h^I_{11} = (Av_1^I, v_1^I)\,, \quad \hat{v}_2^I = Av_1^I - h^I_{11}v_1^I\,, \quad h^I_{21} = \|\hat{v}_2^I\|_2\,,$$

we have

$$h_{11}^I = (Sv_1^F, v_1^F) + O(\varepsilon) = h_{11}^F + O(\varepsilon)\,,$$

$$\hat{v}_2^I = Sv_1^F - h_{11}^F v_1^F + O(\varepsilon) = \hat{v}_2^F + O(\varepsilon)\,,$$

$$h_{21}^I = \|\hat{v}_2^F\|_2 + O(\varepsilon) = h_{21}^F + O(\varepsilon)$$

and hence the lemma holds for $j = 1$. Assume that the lemma has been proved for $j < j_0 \leq \nu$. On that hypothesis, we prove it for $j = j_0$. By Line 10 of the Arnoldi procedure,

$$v_{j_0}^I = \hat{v}_{j_0}^I / h_{j_0,j_0-1}^I$$

which yields that

$$v_{j_0}^I = \frac{\hat{v}_{j_0}^F + O(\varepsilon)}{h_{j_0,j_0-1}^F + O(\varepsilon)} = \frac{\hat{v}_{j_0}^F}{h_{j_0,j_0-1}^F} + O(\varepsilon) = v_{j_0}^F + O(\varepsilon)$$

by the induction hypothesis. Therefore

$$w^I = Av_{j_0}^I = Sv_{j_0}^F + O(\varepsilon) = w^F + O(\varepsilon)$$

for the w^I and w^F in Line 3 of the Arnoldi procedures. Using another induction on the index i in Lines 5 and 6, and the induction hypothesis on j and, in the mean time, noting that $h_{ij_0}^F = 0$ for $1 \leq i \leq j_0 - 2$, we have

$$h_{ij_0}^I = h_{ij_0}^F + O(\varepsilon)$$

for $\max\{1, j_0 - k + 1\} \leq i \leq j_0$ and

$$\hat{v}_{j_0+1}^I = \hat{v}_{j_0+1}^F + O(\varepsilon)\,.$$

From the last equation,

$$h^I_{j_0+1,j_0} = \|\hat{v}^I_{j_0+1}\|_2 = \|\hat{v}^F_{j_0+1}\|_2 + O(\varepsilon) = h^F_{j_0+1,j_0} + O(\varepsilon)$$

and then the induction step is complete. □

We now turn to the DQGMRES algorithm. Consider the linear system

$$Sx = b$$

and denote by x^G_m and x^Q_m the approximate solutions by the GMRES and DQGMRES algorithms, respectively. Let μ be the degree of the minimal polynomial of the vector $b - Sx_0$ with respect to S. A result of the lemma can be stated as follows.

Theorem 3.2 *Given the same initial guess x_0 to GMRES and DQGMRES with $k \geq 2$, then at any given step m with $1 \leq m \leq \mu$,*

$$x^Q_m = x^G_m + O(\varepsilon).$$

Proof. By the definitions of DQGMRES and GMRES, we have

$$x^Q_m = x_0 + \beta^Q V^I_m \left(\left(\bar{H}^I_m \right)^T \bar{H}^I_m \right)^{-1} \left(\bar{H}^I_m \right)^T e_1$$

and

$$x^G_m = x_0 + \beta^G V^F_m \left(\left(\bar{H}^F_m \right)^T \bar{H}^F_m \right)^{-1} \left(\bar{H}^F_m \right)^T e_1$$

where $\beta^Q = \|b - Ax_0\|_2$ and $\beta^G = \|b - Sx_0\|_2$. Since

$$v^I_1 = \frac{1}{\beta^Q}(b - Ax_0) = \frac{1}{\beta^G}(b - Sx_0) + O(\varepsilon) = v^F_1 + O(\varepsilon)\,,$$

we have by the lemma,

$$\bar{H}_m^I = \bar{H}_m^F + O(\varepsilon), \qquad V_m^I = V_m^F + O(\varepsilon)$$

and therefore the desired equation holds. □

If we let x_A be the exact solution to $Ax = b$ and x_S be the exact solution to $Sx = b$, then it is obvious that $x_A = x_S + O(\varepsilon)$. Since, on the other hand, $x_\mu^G = x_S$ we immediately have the following corollary.

Corollary 3.1 *For any initial guess x_0 and any $k \geq 2$,*

$$x_\mu^Q = x_A + O(\varepsilon).$$

The corollary suggests that we may use DQGMRES with small k when A is nearly symmetric or nearly skew symmetric. It may be possible to generalize the above theory to the class of CG(s) matrices of Faber and Manteuffel [21], i.e., the class of matrices for which it is possible to define Krylov subspace algorithms using s-term recurrences.

3.3 Symmetric preconditioning in Bi-CG

Both left-symmetric and right-symmetric preconditioning of the Bi-CG algorithm are relatively straightforward, and no extra vectors are required. In the right-preconditioned Bi-CG algorithm with coefficient matrix A and preconditioner M, the preconditioned coefficient matrix of the dual system is $(AM^{-1})^T = M^{-T}A^T$,

i.e., the dual residual r^* is the residual of

$$M^{-T}A^T x^* = b^*,$$

which is a *left*-preconditioned version of some linear system with A^T. To develop the symmetric right-preconditioned Bi-CG algorithm using M^{-1}-inner products, the preconditioned coefficient matrix of the dual system must be the adjoint of AM^{-1} in the M^{-1} inner product. This is $A^T M^{-1}$ as shown by

$$(AM^{-1}x, y)_{M^{-1}} = (M^{-1}AM^{-1}x, y) = (x, M^{-1}A^T M^{-1}y) = (x, A^T M^{-1}y)_{M^{-1}}.$$

The dual system thus involves the coefficient matrix $A^T M^{-1}$.

Like GMRES, both left and right symmetric preconditioned versions of Bi-CG are equivalent to the split preconditioned version, and this can be shown by a change of variables. However, in both left and right symmetric preconditioned versions, the exact, rather than the split preconditioned residual is available.

The unpreconditioned Bi-CG algorithm cannot have a *serious breakdown* if A is SPD and r_0^* is chosen to be r_0. This is because $r_j^* = r_j$ and $p_j^* = p_j$ for all j and the vectors Ap_j, p_j^* never become orthogonal. In fact, the cosine

$$\frac{(Ap_j, p_j^*)}{\|Ap_j\|\|p_j^*\|} = \frac{(Ap_j, p_j^*)}{\|p_j\|\|p_j^*\|}\frac{\|p_j\|}{\|Ap_j\|}$$

can be bounded below by the reciprocal of the condition number of A.

Similarly, in the symmetric right-preconditioned version of Bi-CG, if both A

and M are SPD, and $r_0^* = r_0$, then $r_j^* = r_j$ and $p_j^* = p_j$ for all j, and

$$\begin{aligned} \frac{(Ap_j, p_j^*)}{\|Ap_j\|\|p_j^*\|} &\geq cond^{-1}(A) \\ \frac{(M^{-1}r_j, r_j^*)}{\|M^{-1}r_j\|\|r_j^*\|} &\geq cond^{-1}(M). \end{aligned}$$

We measure the cosines rather than the quantities (Ap_j, p_j^*) and $(M^{-1}r_j, r_j^*)$ because the p and r vectors have magnitudes going to 0 as the algorithms progress. Recall that in the case when $(M^{-1}r_j, r_j^*) = 0$ and $r_j = 0$, we have a *lucky breakdown.*

For the case of regular right- or left-preconditioning, or if $r_0^* \neq r_0$ in the symmetrically preconditioned cases, then no such lower bounds as the above exist, and the algorithms are liable to break down.

When A is near-symmetric, it is our hypothesis that the probability of breakdown is lower in the symmetrically preconditioned cases, and this will be shown by experiment in the next section.

3.4 Numerical Experiments

§5.1 tests the idea of using symmetric preconditionings with truncated iterative methods. §5.2 tests the breakdown behavior of symmetrically preconditioned Bi-CG.

3.4.1 Truncated iterative methods

To test the idea of using symmetric preconditionings with truncated iterative methods for nearly symmetric systems, we selected a standard fluid flow problem where the degree of symmetry in the matrices is parameterized by the Reynolds number. The flow problem is the two-dimensional square-lid driven cavity, and was discretized by the Galerkin Finite Element method. Rectangular elements were used, with biquadratic basis functions for velocities, and linear discontinuous basis functions for pressure. We considered a segregated solution method for the Navier-Stokes equations, where the velocity and pressure variables are solved separately; the matrices arising from a fully-coupled solution method are otherwise indefinite. In particular, we considered the expression of the conservation of momentum,

$$Re(u \cdot \nabla u) = -\nabla p + \nabla^2 u$$

where u denotes the vector of velocity variables, p denotes the pressure variable, and Re is the Reynolds number. The boundary conditions for the driven cavity problem over the unit square are $u = (1, 0)^T$ on the top edge of the square, and $u = (0, 0)^T$ on the other three sides and the corners. The reference pressure specified at the bottom-left corner is 0.

The matrices are the initial Jacobians at each Newton iteration, assuming a zero pressure distribution. For convenience, however, we chose the right-hand sides of the linear systems to be the vector of all ones. A mesh of 20 by 20 elements was used, leading to momentum equation matrices of order 3042 and having 91204

nonzero entries. The nodes corresponding to the boundaries were not assembled into the matrix, and the degrees of freedom were numbered element by element. For Reynolds number 0., the matrix is SPD, and is equal to the symmetric part of the matrices with nonzero Reynolds number.

We generated matrices with Reynolds number less than 10, which gives rise to the nearly symmetric case. For Reynolds number 1., the degree of symmetry measured by

$$\frac{\|A - A^T\|_F}{\|A + A^T\|_F}$$

has value 7.5102×10^{-4} and this measure increases linearly with the Reynolds number (at least up to $Re = 10$).

In the numerical experiments below, we show the number of matrix-vector products consumed by GMRES(k) and DQGMRES(k) to reduce the *actual* residual norm to less than 10^{-6} of the original residual norm, with a zero initial guess. Several values of k are used. A dagger (†) in the tables indicates that there was no convergence within 500 matrix-vector products. The incomplete Choleski factorization IC(0) of the $Re = 0$ problem was used as the preconditioner in all the problems.

For comparison, we first show in Table 3.1 the results using standard right-preconditioning. Table 3.2 shows the results using right-preconditioning with M^{-1} inner products, or equivalently, split preconditioning. The right-preconditioned methods have a slight advantage in this comparison (by as many as 20 Mat-Vec's),

since they directly minimize the actual residual norm, whereas the symmetrically preconditioned methods minimize a preconditioned residual norm.

When split preconditioning is used, the *actual*, or non-preconditioned residual norm is not available. In this case, for GMRES, we use the preconditioned residual norm in the stopping criterion, and check that the actual residual norm is within twice the tolerance at the end of the iterations. For DQGMRES, since an accurate residual norm estimate is not available within the algorithm, the actual residual norm was computed and used for the stopping criterion for the purpose of this comparison. (This slight mismatch in the stopping criterion accounts for the difference between GMRES(∞) and DQGMRES(2) for the $Re = 0$ case.) In practice, the stopping criterion should use the preconditioned residual norm that is available, not necessarily the actual one.

Table 3.1: Mat-Vec's for convergence for right-preconditioned methods.

	GMRES(k)			DQGMRES(k)								
Re.	5	10	∞	2	3	4	5	6	7	8	9	10
0	232	129	59	76	220	105	72	92	†	160	83	75
1	218	126	69	76	276	131	81	94	†	97	90	79
2	233	126	70	78	391	258	82	95	†	98	94	78
3	208	126	71	87	346	232	85	95	†	99	97	79
4	214	128	72	94	345	†	88	95	477	108	98	86
5	210	128	72	192	394	†	91	95	326	128	99	90
6	214	128	73	446	361	†	94	97	258	197	100	94
7	215	129	73	†	345	†	97	99	239	229	101	96

The results in Table 3.1 show the irregular performance of DQGMRES(k) for these small values of k when the preconditioned system is not symmetric. The performance is entirely regular in Table 3.2, where the preconditioned system is near

Table 3.2: Mat-Vec's for convergence for symmetric right-preconditioned methods.

	GMRES(k)			DQGMRES(k)								
Re.	5	10	∞	2	3	4	5	6	7	8	9	10
0	243	119	57	58	58	58	58	58	58	58	58	58
1	243	119	67	75	74	74	75	74	74	74	75	75
2	244	120	68	78	78	78	79	78	78	78	78	78
3	244	121	69	88	87	87	87	87	86	86	87	87
4	244	122	70	108	95	95	95	93	91	91	93	95
5	244	126	70	†	105	103	105	100	97	96	101	104
6	244	127	71	†	118	111	119	108	101	101	110	117
7	243	128	71	†	131	121	139	117	104	105	121	139

symmetric. For Reynolds numbers up to 3, the systems are sufficiently symmetric so that DQGMRES(2) behaves the same as DQGMRES with much larger k. The performance remains regular until beyond Reynolds number 7, when the number of steps to convergence begins to become irregular, like in the right-preconditioned case.

GMRES with either right or symmetric preconditioning does not show any marked difference in performance; apparently the symmetry of the preconditioned system is not as essential here for this problem. However, the results do show that DQGMRES(k) with small values of k may perform as well, in terms of number of steps, as GMRES(k) with large values of k, particularly if near-symmetry is preserved. Since the former is much more efficient, the combination of preserving symmetry and truncated iterative methods may result in a much more economical method, as well as the more regular behavior shown above.

We also performed the same experiments with orthogonal projection methods, namely the Full Orthogonalization Method (FOM) and its truncated variant, the

Direct Incomplete Orthogonalization Method (DIOM) [50]. The results were very similar to the results above, and are not shown here. Indeed, the development of the algorithms and the theory above is identical for these methods.

For interest, we also performed tests where an ILU(0) preconditioner was constructed for each matrix and compared right and split preconditioning. For the near-symmetric systems here, there was very little difference in these results compared to using IC(0) constructed from the $Re = 0$ case for all the matrices. Thus the deterioration in performance as the Reynolds number increases is not entirely due to a relatively less accurate preconditioner, but is more due to the increased nonsymmetry and non-normality of the matrices. Although the eigenvalues of the preconditioned matrices are identical, their eigenvectors and hence their degree of non-normality may change completely. Unfortunately, it is difficult to quantitatively relate non-normality and convergence.

3.4.2 Breakdown behavior of Bi-CG

To test the breakdown behavior of Bi-CG, MATLAB was used to generate random matrices of order 300 with approximately 50 percent normally distributed nonzero entries. The matrices were adjusted so that

$$A \leftarrow \frac{A + A^T}{2} - (\sigma_{min} + 10^{-5})I + \varepsilon \frac{A - A^T}{2},$$

i.e., the symmetric part was shifted so that the lowest eigenvalue was 10^{-5} and then ε times the skew-symmetric part was added back. The parameter ε was altered to get varying degrees of nonsymmetry.

For each ε that we tested, 100 matrices were generated, and the smallest value of the cosines corresponding to the denominators in the algorithms were recorded. In the right-preconditioned case, we recorded the minimum of

$$\frac{(AM^{-1}p_j, p_j^*)}{\|AM^{-1}p_j\|\|p_j^*\|} \quad and \quad \frac{(r_j, r_j^*)}{\|r_j\|\|r_j^*\|}$$

for all j, and for the symmetric right-preconditioned case, we recorded the minimum of

$$\frac{(Ap_j, p_j^*)}{\|Ap_j\|\|p_j^*\|} \quad and \quad \frac{(M^{-1}r_j, r_j^*)}{\|M^{-1}r_j\|\|r_j^*\|}$$

for all j. The relative residual norm reduction was 10^{-9} when the iterations were stopped. The initial guesses were 0, and r_0^* was set to r_0. IC(0) of the symmetric part was used as the preconditioner.

Table 3.3 shows the frequencies of the size of minimum cosines for the right-preconditioned (first row of each pair of rows) and the symmetrically-preconditioned cases (second row of each pair of rows). For example, all 100 minimum cosines were between 10^{-3} and 3×10^{-3} in the symmetrically-preconditioned case. The average number of Bi-CG steps and the average minimum cosine is also shown. The last column, labeled 'better', shows the number of times that the minimum cosine was higher in the improved algorithm.

The Table shows that the right-preconditioned algorithm can produce much

smaller cosines, indicating a greater probability for breakdown. The difference between the algorithms is less as the degree of nonsymmetry is increased. For $\varepsilon = 0.1$, there is almost no difference in the breakdown behavior of the algorithms. The Table shows that the number of Bi-CG steps is not significantly reduced in the new algorithm, nor is the *average* minimum cosine of the modified algorithm significantly increased. It is the probability that a small cosine is not encountered that is better.

Table 3.3: Frequencies of minimum cosines for right-preconditioned (first row of each pair of rows) and symmetrically-preconditioned (second row of each pair of rows) Bi-CG.

ε	steps	3e-6 to 1e-5	1e-5 to 3e-5	3e-5 to 1e-4	1e-4 to 3e-4	3e-4 to 1e-3	1e-3 to 3e-3	3e-3 to 1e-2	1e-2 to 3e-2	3e-2 to 1e-1	average $\times 10^{-3}$	better
0.000	32.51	1	4	9	19	26	30	11			1.35	
	31.77						100				1.87	74
0.005	30.57		1	2	8	16	34	34	5		3.51	
	29.97					2	1	82	15		8.54	92
0.010	29.27	1	0	3	2	10	29	32	20	3	7.25	
	28.94				1	2	1	6	88	2	15.79	77
0.050	27.53		1	3	8	13	36	31	8		4.15	
	27.32			2	3	7	18	39	26	5	9.47	69
0.100	26.38		1	11	18	39	27	4			0.88	
	26.42		3	4	15	40	27	11			1.26	57

It is important to note that this behavior only applies when r_0^* is set to r_0. When r_0^* is chosen randomly, there is no gain in the symmetrically-preconditioned algorithm, as shown in Table 3.4.

Table 3.4: Frequencies of minimum cosines when r_0^* is chosen randomly.

ε	steps	3e-6 to 1e-5	1e-5 to 3e-5	3e-5 to 1e-4	1e-4 to 3e-4	3e-4 to 1e-3	1e-3 to 3e-3	3e-3 to 1e-2	1e-2 to 3e-2	3e-2 to 1e-1	average $\times 10^{-3}$	better
0.000	33.05	1	4	11	24	26	30	4			0.92	
	32.54	1	1	3	11	24	56	4			1.31	63

Table 3.5: Steps and minimum cosines for the driven cavity problem.

	Bi-CG steps		min cosines	
Re.	right	symm	right	symm
0	70	62	1.52e-4	1.45e-1
1	71	68	1.08e-4	6.73e-3
2	74	72	2.44e-4	5.12e-4
3	73	72	2.02e-4	9.07e-3
4	77	72	1.93e-5	6.52e-3
5	80	75	5.54e-5	5.19e-4
6	80	78	1.91e-4	4.30e-5
7	80	80	1.87e-4	1.02e-3

Table 3.5 shows the number of steps and the minimum cosines for the two algorithms applied to the driven cavity problem described in Section 5.1 above. Figure 3.1 shows a plot of the minimum cosines as the two algorithms progress for the $Re = 1$ problem. Note that the minimum cosines are higher and much smoother in the symmetrically-preconditioned case. In the $Re = 7$ problem, the cosines are still higher, but the smoothness is lost.

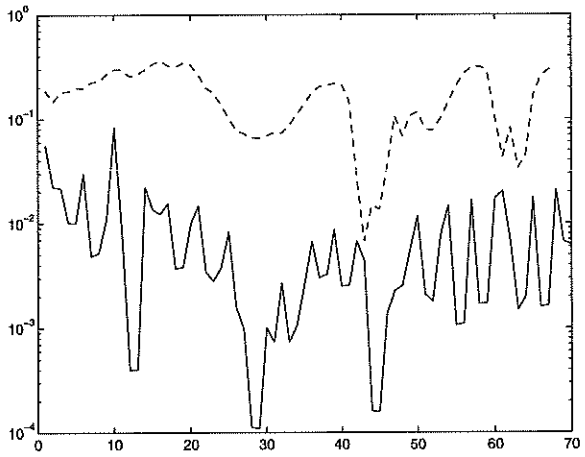

Figure 3.1: Minimum cosines in right-preconditioned Bi-CG (solid line) and symmetrically-preconditioned Bi-CG (dashed line) for the $Re = 1$ problem.

3.5 Conclusions

When solving linear systems with matrices that are very close to being symmetric, this paper has shown that it is possible to improve upon the standard practice of using a (nonsymmetric) preconditioner for that matrix along with a left- or right-preconditioned iterative method. The original degree of symmetry may be maintained by using a symmetric preconditioner and an alternate inner product (or split preconditioning, if appropriate). By combining this idea with truncated iterative methods, solution procedures that converge more quickly and require less storage are developed. The truncated methods also seem to become more robust with the truncation parameter k when near-symmetry is maintained. The Bi-CG algorithm also seems to be more robust with respect to serious breakdown when near-symmetry is maintained.

CHAPTER 4

ML(k)BiCGSTAB: A BiCGSTAB Variant Based on Multiple Lanczos Starting Vectors

We propose a new Krylov subspace solver, which we name ML(k)BiCGSTAB, for solving the nonsymmetric linear system

$$Ax = b$$

in this chapter. The derivation of the new method is similar to that of BiCGSTAB but based on multiple Lanczos process. ML(k)BiCGSTAB can be regarded as a bridge between BiCGSTAB and FOM. In fact, both BiCGSTAB and FOM are special cases of ML(k)BiCGSTAB where $k = 1$ and 2 respectively. In §1, we give our version of the multiple Lanczos process with one right starting vector and k left starting vectors. In §2, a BiCG-like method is derived from the Lanczos method in §1. §3 contains the main derivation for the ML(k)BiCGSTAB method. Numerical results are given in §4.

4.1 A Lanczos method for k Left Starting Linear Independent Vectors

Let A be an $n \times n$ real matrix and given $k+1$ real vectors v_0 and $q_1, q_2, \cdots, q_k$. We define

$$p_{jk+i} = A^{Tj} q_i \tag{4.1}$$

for $i = 1, 2, \cdots, k; j = 0, 1, 2, \cdots$, and suppose the matrices

$$W_l = \begin{bmatrix} p_1^T v_0 & p_1^T A v_0 & \cdots & p_1^T A^{l-1} v_0 \\ p_2^T v_0 & p_2^T A v_0 & \cdots & p_2^T A^{l-1} v_0 \\ \vdots & \vdots & & \vdots \\ p_l^T v_0 & p_l^T A v_0 & \cdots & p_l^T A^{l-1} v_0 \end{bmatrix}, \qquad l = 1, 2, \cdots, \nu$$

are nonsingular, where ν is the grade of v_0 with respect to A, i.e., the degree of the minimal polynomial of v_0 with respect to A.

We now consider a sequence $\{v_l\}_{l=0,1,\cdots,\nu}$ of vectors from the Krylov space

$$K(v_0, A) = span\{v_0, Av_0, A^2 v_0, \cdots\}$$

with the properties

$$v_l \in A^l v_0 + K_l(v_0, A) \equiv A^l v_0 + span\{v_0, Av_0, \cdots, A^{l-1} v_0\} \tag{4.2}$$

and

$$v_l \perp span\{p_1, p_2, \cdots, p_l\}. \tag{4.3}$$

The existence and uniqueness of such a sequence is guaranteed by the nonsingularity of the W_l's. In fact, if we express v_l as

$$v_l = A^l v_0 + \gamma_0^{(l)} v_0 + \gamma_1^{(l)} A v_0 + \cdots + \gamma_{l-1}^{(l)} A^{l-1} v_0 \,, \tag{4.4}$$

then (4.3) is equivalent to $W_l \gamma^{(l)} = -f$ where $\gamma^{(l)} = [\gamma_0^{(l)}, \gamma_1^{(l)}, \cdots, \gamma_{l-1}^{(l)}]^T$ and $f = [p_1^T A^l v_0, \cdots, p_l^T A^l v_0]^T$.

Two simple facts can be derived for the sequence $\{v_l\}_{l=0,1,\cdots,\nu}$: (a) $v_\nu = 0$ and (b) $v_l \not\perp p_{l+1}$ whenever $l < \nu$. To see (a), we note that ν is the grade of v_0 and hence $A^\nu v_0 \in K_\nu(v_0, A)$. Property (4.2) is now reduced to $v_\nu \in K_\nu(v_0, A)$ and thus (a) follows by the uniqueness of v_ν. To prove (b), we assume that $v_l \perp p_{l+1}$. Then combining with property (4.3), we have $v_l \perp span\{p_1, \cdots, p_{l+1}\}$, and hence $W_{l+1}\tilde{\gamma}^{(l)} = 0$, where $\tilde{\gamma}^{(l)} = [\gamma_0^{(l)}, \cdots, \gamma_{l-1}^{(l)}, 1]^T$ and $\gamma_i^{(l)}$ are defined in (4.4), in contradiction with the nonsingularity of W_{l+1}.

Applying (4.4) to itself recursively, we can represent v_l in terms of its previous $v_0, \cdots, v_{l-1}$ as follows,

$$v_l = A v_{l-1} + h_{l-1}^{(l-1)} v_{l-1} + h_{l-2}^{(l-1)} v_{l-2} + \cdots + h_0^{(l-1)} v_0 \,.$$

Noting that $v_i \perp span\{p_1, \cdots, p_i\}$, $v_i \not\perp p_{i+1}$ and $A^T p_i = p_{k+i}$, and examining in turn

$$p_i^T v_l = p_i^T A v_{l-1} + h_{l-1}^{(l-1)} p_i^T v_{l-1} + h_{l-2}^{(l-1)} p_i^T v_{l-2} + \cdots + h_0^{(l-1)} p_i^T v_0$$

for $i = 1, 2, \cdots, l-k-1$, we find all the coefficients zero except $h_{l-1}^{(l-1)}, h_{l-2}^{(l-1)}, \cdots, h_{m_l}^{(l-1)}$ where $m_l = \max(l-k-1, 0)$. Thus we obtain a $k+2$ term recursion relationship

for $\{v_l\}_{l=1,\cdots,\nu}$ of the form,

$$v_l = Av_{l-1} + h_{l-1}^{(l-1)}v_{l-1} + h_{l-2}^{(l-1)}v_{l-2} + \cdots + h_{m_l}^{(l-1)}v_{m_l}\,. \tag{4.5}$$

If we set $V_\nu = [v_0, v_1, \cdots, v_{\nu-1}]$ and $H_\nu = (h_{ij})_{i,j=1,\cdots,\nu}$, the $\nu \times \nu$ Hessenberg matrix with $h_{j+1,j} = 1; h_{ij} = -h_{i-1}^{(j-1)}$ for $m_j + 1 \le i \le j; h_{ij} = 0$ otherwise, and if we recall that $v_\nu = 0$, then the recurrence relations (4.5) can be written in matrix form as

$$AV_\nu = V_\nu H_\nu\,. \tag{4.6}$$

Moreover, set $P_\nu = [p_1, p_2, \cdots, p_\nu]$ and apply P_ν^T to (4.6) from the left,

$$P_\nu^T AV_\nu = P_\nu^T V_\nu H_\nu\,. \tag{4.7}$$

Because of (4.3) and the fact that $v_l \not\perp p_{l+1}$, $P_\nu^T V_\nu$ is a lower triangular matrix with all the entries nonzero in its diagonal. Comparing the corresponding principal blocks of both sides of (4.7), we obtain a condition, which guarantees the nonsingularity of H_ν, that if the matrices

$$S_l = \begin{bmatrix} p_1^T Av_0 & p_1^T A^2 v_0 & \cdots & p_1^T A^l v_0 \\ p_2^T Av_0 & p_2^T A^2 v_0 & \cdots & p_2^T A^l v_0 \\ \vdots & \vdots & & \vdots \\ p_l^T Av_0 & p_l^T A^2 v_0 & \cdots & p_l^T A^l v_0 \end{bmatrix}, \qquad l = 1, 2, \cdots, \nu$$

are nonsingular, so are the principal blocks of H_ν.

Finally, we can see from (4.2) that

$$K_{l+1}(v_0, A) = span\{v_0, v_1, \cdots, v_l\}\,, \qquad l = 0, 1, \cdots, \nu - 1\,. \tag{4.8}$$

Since the dimension of $K_\nu(v_0, A)$ is ν, the vectors $\{v_l\}_{l=0,1,\cdots,\nu-1}$ are linear independent.

Summing up the above discussions, we conclude that

Theorem 4.1 *Given $v_0, q_1, q_2, \cdots, q_k$ and ν, p_l's, W_ν and S_ν as defined above. If the principal submatrices of W_ν are nonsingular, there exist an $n \times \nu$ matrix $V_\nu = [v_0, v_1, \cdots, v_{\nu-1}]$ of rank ν, whose first column is v_0, and an $\nu \times \nu$ Hessenberg matrix H_ν, which has upper bandwidth k and all the entries $h_{j+1,j} = 1$ in its lower subdiagonal, such that*

$$v_l \perp span\{p_1, p_2, \cdots, p_l\}, \quad v_l \not\perp p_{l+1}, \qquad l = 0, 1, \cdots, \nu - 1,$$

and

$$AV_\nu = V_\nu H_\nu . \tag{4.9}$$

Such V_ν and H_ν are unique. Furthermore, if the principal submatrices of S_ν are nonsingular, so are the principal submatrices of H_ν.

In most of cases, the conditions of Theorem 4.1 are satisfied when the q_i's are chosen randomly from a continuous distribution.

Our procedure for generating the vectors v_i's and p_i's are closely related to the multiple starting vector Lanczos method in [1]. In fact, the columns of V_ν are exactly the same as the right Lanczos vectors in [1]. However, the p_i's are different from the left Lanczos vectors in [1]; in fact they are not orthogonal to the

v_i's. It turns out we do not need the full bi-orthogonality property in deriving our extensions of BiCG and BiCGSTAB.

4.2 ML(k)BiCG: A BiCG Variant Based on Multiple Starting Vectors Lanczos

We now turn to the linear system [1]

$$Ax = b \tag{4.10}$$

and we shall derive, based on Theorem 1, a projection method by borrowing the techniques used in the derivation of the Conjugate Gradient algorithm from the symmetric Lanczos procedure [p.176, 12]. Even though this is a well known procedure, in our case there is some difference from the standard case and therefore we include the derivation here for completeness and clarity.

Suppose an initial guess x_0 to (4.10) is given. We set in Theorem 1 $v_0 = b - Ax_0$. At the lth step of our projection method, we seek an approximate solution x_l with

$$x_l \in x_0 + span\{v_0, v_1, \cdots, v_{l-1}\} \tag{4.11}$$

and

$$r_l \equiv b - Ax_l \perp span\{p_1, p_2, \cdots, p_l\}, \tag{4.12}$$

[1] We do not assume the matrix A is nonsingular throughout the paper unless where specified. We just require that the assumptions in Theorem 4.1 hold.

where the v_i's and p_i's are as defined in Theorem 4.1.

Since $P_l^T V_l$, a lower triangular matrix with all diagonal entries nonzero, is nonsingular, it can be shown in the same manner as in deriving CG from the Lanczos basis that x_l is uniquely determined by conditions (4.11) and (4.12) and has the following expression,

$$x_l = x_0 + V_l H_l^{-1} e_1, \tag{4.13}$$

with the corresponding residual r_l is in the same direction as v_l, where $P_l \equiv [p_1, \cdots, p_l]$, $V_l \equiv [v_0, \cdots, v_{l-1}]$, H_l is the $l \times l$ principal submatrix of H_ν, and e_1 is the first column of the $l \times l$ identity matrix. From (4.11) and (4.8), we have $r_l \in v_0 - span\{Av_0, A^2 v_0, \cdots, A^l v_0\}$ and hence $r_l \neq 0$ if $l < \nu$ since ν is the grade of v_0. Moreover, $r_\nu = 0$ from (4.13) and (4.9).

Letting $r_i = \xi_i v_i$ for some scalar ξ_i which is not zero whenever $i < \nu$ and setting $\Lambda_l = diag\{\xi_0, \xi_1, \cdots, \xi_{l-1}\}$, then (4.13) can be rewritten as

$$x_l = x_0 + R_l \Lambda_l^{-1} H_l^{-1} e_1,$$

where $R_l \equiv [r_0, r_1, \cdots, r_{l-1}]$. Write the LDU decomposition of $H_l \Lambda_l$ as

$$H_l \Lambda_l = L_l D_l U_l,$$

which exists due to the nonsingularities of the principal submatrices of $H_l \Lambda_l$, and define

$$G_l \equiv [g_0, g_1, \cdots, g_{l-1}] = R_l U_l^{-1}, \qquad z_l = D_l^{-1} L_l^{-1} e_1.$$

Following again the derivation for CG, we have

$$z_l = \begin{bmatrix} z_{l-1} \\ \\ \alpha_l \end{bmatrix}$$

for some α_l. As a result, x_l can be updated as

$$x_l = x_0 + G_l z_l = x_0 + G_{l-1} z_{l-1} + \alpha_l g_{l-1} = x_{l-1} + \alpha_l g_{l-1} \tag{4.14}$$

and hence

$$r_l = r_{l-1} - \alpha_l A g_{l-1} \,. \tag{4.15}$$

On the other hand, since $G_{l+1} = R_{l+1} U_{l+1}^{-1}$, g_l can be computed from the previous g_i's and r_l by the update

$$g_l = r_l + \beta_{l-1}^{(l)} g_{l-1} + \beta_{l-2}^{(l)} g_{l-2} + \cdots + \beta_{\tilde{m}_l}^{(l)} g_{\tilde{m}_l} \tag{4.16}$$

where $\tilde{m}_l = \max(l-k, 0)$ and where $-\beta_i^{(l)}$ are the nonzero entries of the last column of U_{l+1}.

To compute the coefficients α_l and $\beta_i^{(l)}$ in (4.14), (4.15) and (4.16), we need the A-orthogonality of the vectors g_i's and p_i's. Since

$$\begin{aligned} P_l^T A G_l &= P_l^T A R_l U_l^{-1} = P_l^T A V_l \Lambda_l U_l^{-1} = P_l^T (V_l H_l + v_l e_l^T) \Lambda_l U_l^{-1} \\ &= P_l^T V_l H_l \Lambda_l U_l^{-1} = P_l^T V_l L_l D_l U_l U_l^{-1} = P_l^T V_l L_l D_l \,, \end{aligned}$$

where e_l denotes the last column of the $l \times l$ identity matrix, and since $P_l^T V_l$ is nonsingular and lower triangular, we have

$$p_i^T A g_j = 0 \,, \qquad i \le j \tag{4.17}$$

and

$$p_i^T A g_{i-1} \neq 0\,.$$

Thus, utilizing this information, we examine the following equations derived from (4.15),

$$p_l^T r_l = p_l^T r_{l-1} - \alpha_l p_l^T A g_{l-1}$$

and

$$p_i^T A g_l = p_i^T A r_l + \beta_{l-1}^{(l)} p_i^T A g_{l-1} + \beta_{l-2}^{(l)} p_i^T A g_{l-2} + \cdots + \beta_{\tilde{m}_l}^{(l)} p_i^T A g_{\tilde{m}_l}$$

for i going from $\tilde{m}_l + 1$ to l. Then we get

$$\alpha_l = \frac{p_l^T r_{l-1}}{p_l^T A g_{l-1}} \tag{4.18}$$

and

$$\beta_{i-1}^{(l)} = -\frac{p_i^T A r_l + p_i^T \sum_{j=\tilde{m}_l}^{i-2} \beta_j^{(l)} A g_j}{p_i^T A g_{i-1}}\,, \qquad i = \tilde{m}_l + 1, \cdots, l\,. \tag{4.19}$$

Now, putting the relations (4.14), (4.15), (4.16), (4.18) and (4.19) together, we have

ALGORITHM 4.1. **ML(k)BiCG**

1. *Choose an initial guess* x_0 *and* k *vectors* $q_1, q_2, \cdots, q_k$.
2. *Compute* $r_0 = b - Ax_0$ *and set* $p_1 = q_1$, $g_0 = r_0$.
3. *For* $l = 1, 2, \cdots$, *until convergence:*
4. $\quad \alpha_l = p_l^T r_{l-1} / p_l^T A g_{l-1}$;
5. $\quad x_l = x_{l-1} + \alpha_l g_{l-1}$;
6. $\quad r_l = r_{l-1} - \alpha_l A g_{l-1}$;
7. $\quad$ *For* $s = \max(l-k, 0), \cdots, l-1$

8. $\beta_s^{(l)} = -p_{s+1}^T A \left(r_l + \sum_{t=\max(l-k,0)}^{s-1} \beta_t^{(l)} g_t \right) / p_{s+1}^T A g_s ;$

9. *End*

10. $g_l = r_l + \sum_{s=\max(l-k,0)}^{l-1} \beta_s^{(l)} g_s ;$

11. *Compute* p_{l+1} *according to* (4.1)

12. *End*

It is worthwhile to remark on two special cases where $k = 1$ and $k \geq \nu$. If $k = 1$, then $p_l = A^{T^{l-1}} q_1$ and conditions (4.11), (4.12) become

$$x_l \in x_0 + span\{v_0, v_1, \cdots, v_{l-1}\}, \qquad r_l \perp K_l(q_1, A^T),$$

which are exactly what the BiCG approximate solution x_l^{BiCG} needs to satisfy. As a result, Algorithm 4.1 is equivalent to the BiCG algorithm mathematically. On the other hand, when $k \geq \nu$, (4.11) and (4.12) reduce to

$$x_l \in x_0 + span\{v_0, v_1, \cdots, v_{l-1}\}, \qquad r_l \perp span\{q_1, q_2, \cdots, q_l\}$$

for $1 \leq l \leq \nu$. If, at the lth step of the computations, we choose $(p_{l+1} =) q_{l+1} = r_l$ while setting $(p_1 =) q_1 = r_0$ beforehand, then Algorithm 4.1 is mathematically equivalent to the FOM algorithm.

From its derivation, we can state the following result about Algorithm 4.1.

Theorem 4.2 *Under the assumptions of Theorem 4.1, ML(k)BiCG does not break down by zero division before step* ν *and the approximate solution* x_ν *at step* ν *is exact to the system (4.10).*

4.3 ML(k)BiCGSTAB: A BiCGSTAB Variant Based on Multiple Starting Vectors Lanczos

The implementation of Algorithm 4.1 requires the use of A^T to compute p_{l+1} in Line 11. In practice, however, the transpose of A is not always available, for instance if the matrix is not formed explicitly and the matvec product is only given as an operator. But this difficulty can be overcome by adopting the techniques in the derivations of CGS and BiCGSTAB. In this section, we give a transpose-free version of Algorithm 4.1 which we call ML(k)BiCGSTAB.

We first rearrange the outer *for* loop of Algorithm 4.1 into a form more convenient for our development. Let $l = jk + i$ and let the index i vary from 1 to k and j starting with 0. Then we convert the loop in l into doubly nested loops in j and i respectively. By moving the case where $i = 1$ outside the i-loop, we rewrite the l-loop (omitting Lines 5 and 11) of Algorithm 4.1 as,

1. *For* $j = 0, 1, 2, \cdots$

2. $\alpha_{jk+1} = p_{jk+1}^T r_{(j-1)k+k} / p_{jk+1}^T A g_{(j-1)k+k}$;

3. $r_{jk+1} = r_{(j-1)k+k} - \alpha_{jk+1} A g_{(j-1)k+k}$;

4. $For\ i = 1, 2, \cdots, k$

5. $For\ s = \max((j-1)k + i, 0), \cdots, jk + i - 1$

6. $\beta_s^{(jk+i)} = -p_{s+1}^T A \left(r_{jk+i} + \sum_{t=\max((j-1)k+i,0)}^{s-1} \beta_t^{(jk+i)} g_t \right) \Big/ p_{s+1}^T A g_s$;

7. *End*

8. $g_{jk+i} = r_{jk+i} + \sum_{s=\max((j-1)k+i,0)}^{jk+i-1} \beta_s^{(jk+i)} g_s;$

9. *If* $i < k$

10. $\alpha_{jk+i+1} = p_{jk+i+1}^T r_{jk+i} / p_{jk+i+1}^T A g_{jk+i};$

11. $r_{jk+i+1} = r_{jk+i} - \alpha_{jk+i+1} A g_{jk+i};$

12. *End*

13. *End*

14. *End*

in which Lines 5-8 can be again expanded into:

1. *For* $s = \max((j-1)k+i, 0), \cdots, (j-1)k+k-1$

2. $\beta_s^{(jk+i)} = -p_{s+1}^T A \left(r_{jk+i} + \sum_{t=\max((j-1)k+i,0)}^{s-1} \beta_t^{(jk+i)} g_t \right) \Big/ p_{s+1}^T A g_s;$

3. *End*

4. $\beta_{(j-1)k+k}^{(jk+i)} = -p_{jk+1}^T A \left(r_{jk+i} + \sum_{t=\max((j-1)k+i,0)}^{(j-1)k+k-1} \beta_t^{(jk+i)} g_t \right) \Big/ p_{jk+1}^T A g_{(j-1)k+k};$

5. *For* $s = jk+1, \cdots, jk+i-1$

6. $\beta_s^{(jk+i)} = -p_{s+1}^T A \left(r_{jk+i} + \sum_{t=\max((j-1)k+i,0)}^{(j-1)k+k} \beta_t^{(jk+i)} g_t + \sum_{t=jk+1}^{s-1} \beta_t^{(jk+i)} g_t \right) \Big/ p_{s+1}^T A g_s;$

7. *End*

8. $g_{jk+i} = r_{jk+i} + \sum_{s=\max((j-1)k+i,0)}^{(j-1)k+k} \beta_s^{(jk+i)} g_s + \sum_{s=jk+1}^{jk+i-1} \beta_s^{(jk+i)} g_s;$

and further into:

1. $If\ j = 0$

2. $\beta_0^{(i)} = -p_1^T A r_i / p_1^T A g_0;$

3. $For\ s = 1, \cdots, i-1$

4. $\beta_s^{(i)} = -p_{s+1}^T A \left(r_i + \beta_0^{(i)} g_0 + \sum_{t=1}^{s-1} \beta_t^{(i)} g_t \right) \Big/ p_{s+1}^T A g_s;$

5. *End*

6. $g_i = r_i + \sum_{s=0}^{i-1} \beta_s^{(i)} g_s;$

7. *Else*

8. $For\ s = i, \cdots, k-1$

9. $$\beta_{(j-1)k+s}^{(jk+i)} = -p_{(j-1)k+s+1}^T A \Big(r_{jk+i} + \sum_{t=i}^{s-1} \beta_{(j-1)k+t}^{(jk+i)} g_{(j-1)k+t} \Big) \Big/ p_{(j-1)k+s+1}^T A g_{(j-1)k+s};$$

10. *End*

11. $\beta_{(j-1)k+k}^{(jk+i)} = -p_{jk+1}^T A \left(r_{jk+i} + \sum_{t=i}^{k-1} \beta_{(j-1)k+t}^{(jk+i)} g_{(j-1)k+t} \right) \Big/ p_{jk+1}^T A g_{(j-1)k+k};$

12. $For\ s = 1, \cdots, i-1$

13. $$\beta_{jk+s}^{(jk+i)} = -p_{jk+s+1}^T A \left(r_{jk+i} + \sum_{t=i}^{k} \beta_{(j-1)k+t}^{(jk+i)} g_{(j-1)k+t} + \sum_{t=1}^{s-1} \beta_{jk+t}^{(jk+i)} g_{jk+t} \right) \Big/ p_{jk+s+1}^T A g_{jk+s};$$

14. End

15. $$g_{jk+i} = r_{jk+i} + \sum_{s=i}^{k} \beta_{(j-1)k+s}^{(jk+i)} g_{(j-1)k+s} + \sum_{s=1}^{i-1} \beta_{jk+s}^{(jk+i)} g_{jk+s};$$

16. End

We now introduce an auxiliary polynomial $\psi_l(\lambda)$ defined by the recurrence relation,

$$\psi_0(\lambda) = 1\,,$$

$$\psi_l(\lambda) = (\rho_l \lambda + 1)\psi_{l-1}(\lambda)\,, \qquad l = 1, 2, \cdots,$$

where ρ_l is a free parameter. This polynomial $\psi_l(\lambda)$ was first used by van der Vorst in the derivation of BiCGSTAB [60]. If we express $\psi_l(\lambda)$ in terms of the power basis,

$$\psi_l(\lambda) = \eta_l^{(l)} \lambda^l + \cdots + \eta_1^{(l)} \lambda + \eta_0^{(l)}\,,$$

then it is clear that $\eta_l^{(l)} = \rho_1 \rho_2 \cdots \rho_l$ and $\eta_0^{(l)} = 1$.

Next, we define the following vectors, analogous to those defined in BiCGSTAB,

$$\tilde{\pi}_{jk+i} = \psi_j(A) r_{jk+i}\,, \qquad \pi_{jk+i} = \psi_{j+1}(A) r_{jk+i}\,,$$

$$\tilde{\omega}_{jk+i} = \psi_j(A)g_{jk+i}\,, \qquad \omega_{jk+i} = \psi_{j+1}(A)g_{jk+i}\,,$$

for $i = 1, 2, \cdots, k; j = 0, 1, \cdots$, and set $\pi_0 = \omega_0 = r_0\,(= g_0)$. Our goal is to define π_{jk+i} to be the residual of our new method. The other three vectors are needed in deriving a recurrence for π_{jk+i}. By recalling (4.1), (4.12) and (4.17), we find that the scalars α_l's and β_l's in Algorithm 4.1 can be computed via these new vectors. The derivations are quite complicated and therefore we give the details in the Appendix while summarizing only the results here. In fact, we have

$$\alpha_{jk+1} = \frac{q_1^T \pi_{(j-1)k+k}}{q_1^T A\omega_{(j-1)k+k}}$$

for $0 \le j, i = 1$;

$$\alpha_{jk+i+1} = \frac{\rho_{j+1} q_{i+1}^T \tilde{\pi}_{jk+i}}{q_{i+1}^T(\omega_{jk+i} - \tilde{\omega}_{jk+i})}$$

for $0 \le j, 1 \le i < k$;

$$\beta_0^{(i)} = -\frac{q_1^T \pi_i}{\rho_1 q_1^T A\omega_0}$$

for $1 \le i \le k, j = 0$;

$$\beta_s^{(i)} = -\frac{q_{s+1}^T \left(\pi_i + \beta_0^{(i)} \rho_1 A\omega_0 + \sum_{t=1}^{s-1} \beta_t^{(i)} (\omega_t - \tilde{\omega}_t)\right)}{q_{s+1}^T(\omega_s - \tilde{\omega}_s)}$$

for $1 \le s < i \le k, j = 0$;

$$\beta_{(j-1)k+s}^{(jk+i)} = -\frac{q_{s+1}^T \left(\tilde{\pi}_{jk+i} + \sum_{t=i}^{s-1} \beta_{(j-1)k+t}^{(jk+i)} (\omega_{(j-1)k+t} - \tilde{\omega}_{(j-1)k+t})\right)}{q_{s+1}^T(\omega_{(j-1)k+s} - \tilde{\omega}_{(j-1)k+s})}$$

for $1 \le j, 1 \le i \le s \le k-1$;

$$\beta_{(j-1)k+k}^{(jk+i)} = -\frac{q_1^T \left(\pi_{jk+i} + \rho_{j+1} \sum_{t=i}^{k-1} \beta_{(j-1)k+t}^{(jk+i)} A\omega_{(j-1)k+t}\right)}{\rho_{j+1} q_1^T A\omega_{(j-1)k+k}}$$

for $1 \leq j, 1 \leq i \leq k$;

$$\beta_{jk+s}^{(jk+i)} = -\frac{q_{s+1}^T \left(\pi_{jk+i} + \rho_{j+1} \sum_{t=i}^{k} \beta_{(j-1)k+t}^{(jk+i)} A\omega_{(j-1)k+t} + \sum_{t=1}^{s-1} \beta_{jk+t}^{(jk+i)} (\omega_{jk+t} - \tilde{\omega}_{jk+t})\right)}{q_{s+1}^T (\omega_{jk+s} - \tilde{\omega}_{jk+s})}$$

for $1 \leq j, 1 \leq s < i \leq k$.

The vectors $\tilde{\pi}_{jk+i}$, π_{jk+i}, $\tilde{\omega}_{jk+i}$ and ω_{jk+i} themselves can be updated according to Algorithm 4.1 as follows,

$$\begin{aligned}
\tilde{\pi}_{jk+1} &= \psi_j(A) r_{jk+1} \\
&= \psi_j(A)(r_{(j-1)k+k} - \alpha_{jk+1} A g_{(j-1)k+k}) \\
&= \pi_{(j-1)k+k} - \alpha_{jk+1} A\omega_{(j-1)k+k}
\end{aligned}$$

for $0 \leq j, i = 1$;

$$\begin{aligned}
\pi_{jk+1} &= \psi_{j+1}(A) r_{jk+1} \\
&= (\rho_{j+1} A\psi_j(A) + \psi_j(A)) r_{jk+1} \\
&= \rho_{j+1} A\tilde{\pi}_{jk+1} + \tilde{\pi}_{jk+1}
\end{aligned}$$

for $0 \leq j, i = 1$;

$$\begin{aligned}
\tilde{\pi}_{jk+i+1} &= \psi_j(A) r_{jk+i+1} \\
&= \psi_j(A)(r_{jk+i} - \alpha_{jk+i+1} A g_{jk+i})
\end{aligned}$$

$$= \psi_j(A)r_{jk+i} - \frac{\alpha_{jk+i+1}}{\rho_{j+1}}(\psi_{j+1}(A) - \psi_j(A))g_{jk+i}$$

$$= \tilde{\pi}_{jk+i} - \frac{\alpha_{jk+i+1}}{\rho_{j+1}}(\omega_{jk+i} - \tilde{\omega}_{jk+i})$$

for $1 \leq i < k, 0 \leq j$;

$$\pi_{jk+i+1} = \psi_{j+1}(A)r_{jk+i+1}$$

$$= \psi_{j+1}(A)(r_{jk+i} - \alpha_{jk+i+1}Ag_{jk+i})$$

$$= \pi_{jk+i} - \alpha_{jk+i+1}A\omega_{jk+i}$$

for $1 \leq i < k, 0 \leq j$;

$$\tilde{\omega}_i = \psi_0(A)g_i$$

$$= \psi_0(A)\left(r_i + \textstyle\sum_{s=0}^{i-1}\beta_s^{(i)}g_s\right)$$

$$= \tilde{\pi}_i + \beta_0^{(i)}\omega_0 + \textstyle\sum_{s=1}^{i-1}\beta_s^{(i)}\tilde{\omega}_s$$

for $1 \leq i \leq k, j = 0$;

$$\omega_i = \psi_1(A)g_i$$

$$= \psi_1(A)\left(r_i + \textstyle\sum_{s=0}^{i-1}\beta_s^{(i)}g_s\right)$$

$$= \psi_1(A)r_i + \beta_0^{(i)}\psi_1(A)g_0 + \textstyle\sum_{s=1}^{i-1}\beta_s^{(i)}\psi_1(A)g_s$$

$$= \pi_i + \beta_0^{(i)}(\rho_1 A\omega_0 + \omega_0) + \sum_{s=1}^{i-1} \beta_s^{(i)}\omega_s$$

for $1 \leq i \leq k, j = 0$;

$$\tilde{\omega}_{jk+i} = \psi_j(A)g_{jk+i}$$

$$= \psi_j(A)\left(r_{jk+i} + \sum_{s=i}^{k} \beta_{(j-1)k+s}^{(jk+i)} g_{(j-1)k+s} + \sum_{s=1}^{i-1} \beta_{jk+s}^{(jk+i)} g_{jk+s}\right)$$

$$= \tilde{\pi}_{jk+i} + \sum_{s=i}^{k} \beta_{(j-1)k+s}^{(jk+i)} \omega_{(j-1)k+s} + \sum_{s=1}^{i-1} \beta_{jk+s}^{(jk+i)} \tilde{\omega}_{jk+s}$$

for $1 \leq j, 1 \leq i \leq k$;

$$\omega_{jk+i} = \psi_{j+1}(A)g_{jk+i}$$

$$= \psi_{j+1}(A)\left(r_{jk+i} + \sum_{s=i}^{k} \beta_{(j-1)k+s}^{(jk+i)} g_{(j-1)k+s} + \sum_{s=1}^{i-1} \beta_{jk+s}^{(jk+i)} g_{jk+s}\right)$$

$$= \psi_{j+1}(A)r_{jk+i} + \sum_{s=i}^{k} \beta_{(j-1)k+s}^{(jk+i)} (\rho_{j+1}A + I)\psi_j(A)g_{(j-1)k+s} +$$

$$\sum_{s=1}^{i-1} \beta_{jk+s}^{(jk+i)} \psi_{j+1}(A)g_{jk+s}$$

$$= \pi_{jk+i} + \sum_{s=i}^{k} \beta_{(j-1)k+s}^{(jk+i)} (\rho_{j+1}A\omega_{(j-1)k+s} + \omega_{(j-1)k+s}) + \sum_{s=1}^{i-1} \beta_{jk+s}^{(jk+i)} \omega_{jk+s}$$

for $1 \leq j, 1 \leq i \leq k$, where I is the $n \times n$ identity matrix.

The formulas we have just derived constitute the main operations of the ML(k)BiCGSTAB algorithm, which we summarize as follows.

1. *Set* $\pi_0 = \omega_0 = r_0$.

2. *For* $j = 0, 1, 2, \cdots$

3. $\alpha_{jk+1} = \dfrac{q_1^T \pi_{(j-1)k+k}}{q_1^T A \omega_{(j-1)k+k}}$;

4. $\tilde{\pi}_{jk+1} = \pi_{(j-1)k+k} - \alpha_{jk+1} A \omega_{(j-1)k+k}$;

5. *Choose* $\rho_{j+1} \neq 0$

6. $\pi_{jk+1} = \rho_{j+1} A \tilde{\pi}_{jk+1} + \tilde{\pi}_{jk+1}$;

7. $For\ i = 1, 2, \cdots, k$

8. $If\ j = 0$

9. $\beta_0^{(i)} = -\dfrac{q_1^T \pi_i}{\rho_1 q_1^T A \omega_0}$;

10. $For\ s = 1, \cdots, i-1$

11. $\beta_s^{(i)} = -\dfrac{q_{s+1}^T \left(\pi_i + \beta_0^{(i)} \rho_1 A \omega_0 + \sum_{t=1}^{s-1} \beta_t^{(i)} (\omega_t - \tilde{\omega}_t) \right)}{q_{s+1}^T (\omega_s - \tilde{\omega}_s)}$;

12. *End*

13. $\tilde{\omega}_i = \tilde{\pi}_i + \beta_0^{(i)} \omega_0 + \sum_{s=1}^{i-1} \beta_s^{(i)} \tilde{\omega}_s$;

14. $\omega_i = \pi_i + \beta_0^{(i)} (\rho_1 A \omega_0 + \omega_0) + \sum_{s=1}^{i-1} \beta_s^{(i)} \omega_s$;

15. *Else*

16. $For\ s = i, \cdots, k-1$

17. $\beta_{(j-1)k+s}^{(jk+i)} = -\dfrac{q_{s+1}^T\left(\tilde{\pi}_{jk+i} + \sum_{t=i}^{s-1}\beta_{(j-1)k+t}^{(jk+i)}(\omega_{(j-1)k+t} - \tilde{\omega}_{(j-1)k+t})\right)}{q_{s+1}^T(\omega_{(j-1)k+s} - \tilde{\omega}_{(j-1)k+s})}$;

18. *End*

19. $\beta_{(j-1)k+k}^{(jk+i)} = -\dfrac{q_1^T\left(\pi_{jk+i} + \rho_{j+1}\sum_{t=i}^{k-1}\beta_{(j-1)k+t}^{(jk+i)}A\omega_{(j-1)k+t}\right)}{\rho_{j+1}q_1^T A\omega_{(j-1)k+k}}$;

20. *For* $s = 1, \cdots, i-1$

21. $\beta_{jk+s}^{(jk+i)} = -q_{s+1}^T\left(\pi_{jk+i} + \rho_{j+1}\sum_{t=i}^{k}\beta_{(j-1)k+t}^{(jk+i)}A\omega_{(j-1)k+t} + \sum_{t=1}^{s-1}\beta_{jk+t}^{(jk+i)}(\omega_{jk+t} - \tilde{\omega}_{jk+t})\right) \Big/ q_{s+1}^T(\omega_{jk+s} - \tilde{\omega}_{jk+s})$;

22. *End*

23. $\tilde{\omega}_{jk+i} = \tilde{\pi}_{jk+i} + \sum_{s=i}^{k}\beta_{(j-1)k+s}^{(jk+i)}\omega_{(j-1)k+s} + \sum_{s=1}^{i-1}\beta_{jk+s}^{(jk+i)}\tilde{\omega}_{jk+s}$;

24. $\omega_{jk+i} = \pi_{jk+i} + \sum_{s=i}^{k}\beta_{(j-1)k+s}^{(jk+i)}(\rho_{j+1}A\omega_{(j-1)k+s} + \omega_{(j-1)k+s}) + \sum_{s=1}^{i-1}\beta_{jk+s}^{(jk+i)}\omega_{jk+s}$;

25. *End*

26. *If* $i < k$

27. $\alpha_{jk+i+1} = \dfrac{\rho_{j+1}q_{i+1}^T\tilde{\pi}_{jk+i}}{q_{i+1}^T(\omega_{jk+i} - \tilde{\omega}_{jk+i})}$;

28. $\tilde{\pi}_{jk+i+1} = \tilde{\pi}_{jk+i} - \dfrac{\alpha_{jk+i+1}}{\rho_{j+1}}(\omega_{jk+i} - \tilde{\omega}_{jk+i})$;

29. $\pi_{jk+i+1} = \pi_{jk+i} - \alpha_{jk+i+1}A\omega_{jk+i}$;

30. *End*

31. *End*

32. *End*

Some simplifications can be made to these operations, for instance, (a) resetting the scalar α_{jk+i+1} in Line 27 to be $q_{i+1}^T \tilde{\pi}_{jk+i} / q_{i+1}^T (\omega_{jk+i} - \tilde{\omega}_{jk+i})$ since α_{jk+i+1} is only used in Lines 28 and 29 and the factor ρ_{j+1} will be cancelled in Line 28; (b) merging Lines 9-14 and Lines 19-24 by adding a conditional control "$if\ j \geq 1$" in Line 16 and treating $\beta_l^{(i)}$ with $l < 0$ as zero; such $\beta_l^{(i)}$'s will appear in Lines 19-24 when $j = 0$; (c) introducing the auxiliary vector

$$d_{jk+i} \equiv \omega_{jk+i} - \tilde{\omega}_{jk+i} ,$$

which can be updated by using Lines 23 and 24 as

$$d_{jk+i} = \pi_{jk+i} - \tilde{\pi}_{jk+i} + \rho_{j+1} \sum_{s=i}^{k} \beta_{(j-1)k+s}^{(jk+i)} A\omega_{(j-1)k+s} + \sum_{s=1}^{i-1} \beta_{jk+s}^{(jk+i)} d_{jk+s} .$$

With these changes, we rewrite Lines 8-30 as,

1. *For* $s = i, \cdots, k-1$ *and* $j \geq 1$

2. $\beta_{(j-1)k+s}^{(jk+i)} = -\dfrac{q_{s+1}^T \left(\tilde{\pi}_{jk+i} + \sum_{t=i}^{s-1} \beta_{(j-1)k+t}^{(jk+i)} d_{(j-1)k+t} \right)}{q_{s+1}^T d_{(j-1)k+s}}$;

3. *End*

4. $\beta_{(j-1)k+k}^{(jk+i)} = -\dfrac{q_1^T \left(\pi_{jk+i} + \rho_{j+1} \sum_{t=i}^{k-1} \beta_{(j-1)k+t}^{(jk+i)} A\omega_{(j-1)k+t} \right)}{\rho_{j+1} q_1^T A\omega_{(j-1)k+k}}$;

5. $For\ s = 1, \cdots, i-1$

6. $\beta_{jk+s}^{(jk+i)} = -\dfrac{q_{s+1}^T \left(\pi_{jk+i} + \rho_{j+1} \sum_{t=i}^{k} \beta_{(j-1)k+t}^{(jk+i)} A\omega_{(j-1)k+t} + \sum_{t=1}^{s-1} \beta_{jk+t}^{(jk+i)} d_{jk+t}\right)}{q_{s+1}^T d_{jk+s}};$

7. End

8. $d_{jk+i} = \pi_{jk+i} - \tilde{\pi}_{jk+i} + \rho_{j+1} \sum_{s=i}^{k} \beta_{(j-1)k+s}^{(jk+i)} A\omega_{(j-1)k+s} + \sum_{s=1}^{i-1} \beta_{jk+s}^{(jk+i)} d_{jk+s};$

9. $\omega_{jk+i} = \pi_{jk+i} + \sum_{s=i}^{k} \beta_{(j-1)k+s}^{(jk+i)} (\rho_{j+1} A\omega_{(j-1)k+s} + \omega_{(j-1)k+s}) + \sum_{s=1}^{i-1} \beta_{jk+s}^{(jk+i)} \omega_{jk+s};$

10. $If\ i < k$

11. $\alpha_{jk+i+1} = \dfrac{q_{i+1}^T \tilde{\pi}_{jk+i}}{q_{i+1}^T d_{jk+i}};$

12. $\tilde{\pi}_{jk+i+1} = \tilde{\pi}_{jk+i} - \alpha_{jk+i+1} d_{jk+i};$

13. $\pi_{jk+i+1} = \pi_{jk+i} - \rho_{j+1} \alpha_{jk+i+1} A\omega_{jk+i};$

14. End

in which Lines 1-9 can be further written as,

1. $z_d = \tilde{\pi}_{jk+i};\ z_\omega = \pi_{jk+i};\ z_{A\omega} = 0;$

2. $For\ s = i, \cdots, k-1\ and\ j \geq 1$

3. $\beta_{(j-1)k+s}^{(jk+i)} = -\dfrac{q_{s+1}^T z_d}{q_{s+1}^T d_{(j-1)k+s}};$

4. $z_d = z_d + \beta_{(j-1)k+s}^{(jk+i)} d_{(j-1)k+s};$

5. $z_\omega = z_\omega + \beta^{(jk+i)}_{(j-1)k+s}\omega_{(j-1)k+s}$;

6. $z_{A\omega} = z_{A\omega} + \beta^{(jk+i)}_{(j-1)k+s}A\omega_{(j-1)k+s}$;

7. *End*

8. $\beta^{(jk+i)}_{(j-1)k+k} = -\dfrac{q_1^T\left(\pi_{jk+i} + \rho_{j+1}z_{A\omega}\right)}{\rho_{j+1}q_1^T A\omega_{(j-1)k+k}}$;

9. $z_\omega = z_\omega + \beta^{(jk+i)}_{(j-1)k+k}\omega_{(j-1)k+k}$;

10. $z_{A\omega} = \rho_{j+1}\left(z_{A\omega} + \beta^{(jk+i)}_{(j-1)k+k}A\omega_{(j-1)k+k}\right)$;

11. $z_d = \pi_{jk+i} + z_{A\omega}$;

12. *For* $s = 1, \cdots, i-1$

13. $\beta^{(jk+i)}_{jk+s} = -\dfrac{q_{s+1}^T z_d}{q_{s+1}^T d_{jk+s}}$;

14. $z_d = z_d + \beta^{(jk+i)}_{jk+s}d_{jk+s}$;

15. $z_\omega = z_\omega + \beta^{(jk+i)}_{jk+s}\omega_{jk+s}$;

16. *End*

17. $d_{jk+i} = z_d - \tilde{\pi}_{jk+i}$;

18. $\omega_{jk+i} = z_\omega + z_{A\omega}$;

As the approximate solution x_l at step $l(=jk+i)$ of the ML(k)BiCGSTAB algorithm, we define

$$x_{jk+1} = x_{(j-1)k+k} - \rho_{j+1}\tilde{\pi}_{jk+1} + \alpha_{jk+1}\omega_{(j-1)k+k} \tag{4.20}$$

for $j \geq 0$ and

$$x_{jk+i+1} = x_{jk+i} + \rho_{j+1}\alpha_{jk+i+1}\omega_{jk+i} \tag{4.21}$$

for $j \geq 0, k > i \geq 1$. One can readily verify by induction that π_l is the residual vector of x_l, that is, $\pi_l = b - Ax_l$.

We are now ready to state the ML(k)BiCGSTAB algorithm formally. As part of the algorithm, we choose the parameter ρ_{j+1} to minimize the 2-norm of the vector $\pi_{jk+1} = \rho_{j+1}A\tilde{\pi}_{jk+1} + \tilde{\pi}_{jk+1}$, i.e., $\rho_{j+1} = -\tilde{\pi}_{jk+1}^T A\tilde{\pi}_{jk+1}/\|A\tilde{\pi}_{jk+1}\|^2$. Noting that the data $q_1^T A\omega_{(j-1)k+k}$, $q_{s+1}^T d_{(j-1)k+s}$ and $q_{s+1}^T d_{jk+s}$ are repeatly used inside each loop of the control variable j, to save the computational cost, we introduce a variable $c_{j'k+i'}$ such that $c_{j'k+i'} = q_{i'+1}^T d_{j'k+i'}$ if $1 \leq i' \leq k-1$ and $c_{j'k+k} = q_1^T A\omega_{j'k+k}$ if $i' = k$. Moreover, since π_l is the residual of x_l, we relabel π_l as r_l. We also relabel ω_l and $\tilde{\pi}_l$ as g_l and u_l respectively.

ALGORITHM 4.2. **ML(k)BiCGSTAB**

1. *Choose an initial guess* x_0 *and* k *vectors* $q_1, q_2, \cdots, q_k$.
2. *Compute* $r_0 = b - Ax_0$ *and set* $g_0 = r_0$.
3. *For* $j = 0, 1, 2, \cdots$
4. $\quad w_{(j-1)k+k} = Ag_{(j-1)k+k}$;
5. $\quad c_{(j-1)k+k} = q_1^T w_{(j-1)k+k}$;
6. $\quad \alpha_{jk+1} = q_1^T r_{(j-1)k+k}/c_{(j-1)k+k}$;

7. $u_{jk+1} = r_{(j-1)k+k} - \alpha_{jk+1} w_{(j-1)k+k}$;

8. $\rho_{j+1} = -u_{jk+1}^T A u_{jk+1} / \|Au_{jk+1}\|^2$;

9. $x_{jk+1} = x_{(j-1)k+k} - \rho_{j+1} u_{jk+1} + \alpha_{jk+1} g_{(j-1)k+k}$;

10. $r_{jk+1} = \rho_{j+1} A u_{jk+1} + u_{jk+1}$;

11. $For\ i = 1, 2, \cdots, k$

12. $z_d = u_{jk+i}$; $z_g = r_{jk+i}$; $z_w = 0$;

13. $For\ s = i, \cdots, k-1\ and\ j \geq 1$

14. $\beta_{(j-1)k+s}^{(jk+i)} = -q_{s+1}^T z_d / c_{(j-1)k+s}$;

15. $z_d = z_d + \beta_{(j-1)k+s}^{(jk+i)} d_{(j-1)k+s}$;

16. $z_g = z_g + \beta_{(j-1)k+s}^{(jk+i)} g_{(j-1)k+s}$;

17. $z_w = z_w + \beta_{(j-1)k+s}^{(jk+i)} w_{(j-1)k+s}$;

18. End

19. $\beta_{(j-1)k+k}^{(jk+i)} = -\dfrac{q_1^T \left(r_{jk+i} + \rho_{j+1} z_w\right)}{\rho_{j+1} c_{(j-1)k+k}}$;

20. $z_g = z_g + \beta_{(j-1)k+k}^{(jk+i)} g_{(j-1)k+k}$;

21. $z_w = \rho_{j+1} \left(z_w + \beta_{(j-1)k+k}^{(jk+i)} w_{(j-1)k+k}\right)$;

22. $z_d = r_{jk+i} + z_w$;

23. $For\ s = 1, \cdots, i-1$

24. $\beta_{jk+s}^{(jk+i)} = -q_{s+1}^T z_d / c_{jk+s}$;

25. $z_d = z_d + \beta_{jk+s}^{(jk+i)} d_{jk+s}$;

26. $z_g = z_g + \beta_{jk+s}^{(jk+i)} g_{jk+s}$;

27. End

28. $d_{jk+i} = z_d - u_{jk+i}$;

29. $g_{jk+i} = z_g + z_w$;

30. $If\ i < k$

31. $c_{jk+i} = q_{i+1}^T d_{jk+i}$;

32. $\alpha_{jk+i+1} = q_{i+1}^T u_{jk+i}/c_{jk+i}$;

33. $u_{jk+i+1} = u_{jk+i} - \alpha_{jk+i+1} d_{jk+i}$;

34. $x_{jk+i+1} = x_{jk+i} + \rho_{j+1}\alpha_{jk+i+1} g_{jk+i}$;

35. $w_{jk+i} = A g_{jk+i}$;

36. $r_{jk+i+1} = r_{jk+i} - \rho_{j+1}\alpha_{jk+i+1} w_{jk+i}$;

37. *End*

38. *End*

39. *End*

The denominators in the ML(k)BiCGSTAB algorithm appear only in the evaluations of α's and β's and from the process of their derivations, these denominators differ from their counterparts in Algorithm 4.1, but can not be zero if the denominators in Algorithm 4.1 are not zero, assuming the ρ_j's defined in Algorithm 4.2 are nonzero. Hence, under the assumption that $\rho_{j+1} \neq 0$ for all j, we can see that if Algorithm 4.1 does not break down by zero division at some step, then neither does ML(k)BiCGSTAB at the same step. Moreover, since the residual vector r_{jk+i}^2 is by definition the vector $\psi_{j+1}(A) r_{jk+i}^1$, where r_{jk+i}^1 and r_{jk+i}^2 denote the corresponding residual vectors in Algorithms 4.1 and 4.2 respectively, and since $r_{jk+i}^1 \in K_{jk+i+1}(v_0, A)$, $r_{jk+i}^2 \in K_{jk+i+j+2}(v_0, A)$. Thus it is possible that r_{jk+i}^2 vanishes when $jk + i + j + 1 \geq \nu$, or $jk + i \geq \nu - j - 1$. On the other hand, r_ν^2 must be zero because $r_\nu^1 = 0$.

Theorem 4.3 *Under the assumptions of Theorem 4.1 and suppose $\rho_{j+1} \neq 0$ for all j, the ML(k)BiCGSTAB algorithm does not break down by zero division before*

step ν and an exact solution [2] to (4.10) is obtained at or before step ν.

A similar remark to the one at the end of §3 can also be made here. Mathematically, ML(1)BiCGSTAB is equivalent to BiCGSTAB since it was established based on BiCG by using exactly the same techniques used in deriving BiCGSTAB. In the case where $k \geq \nu$, we can obtain an exact solution in the first loop of j, i.e., $j = 0$, and the algorithm now can be regarded as a FOM algorithm (with the q_i's appropriately chosen), for the reasons stated in the following. Since

$$r_i^1 \in v_0 - span\{Av_0, A^2v_0, \cdots, A^iv_0\}$$

from §3, we have

$$r_i^2 = \psi_1(A)r_i^1 \in r_0^F - span\{Ar_0^F, A^2r_0^F, \cdots, A^ir_0^F\},$$

where $v_0 = b - Ax_0$, $r_0^F \equiv \psi_1(A)v_0$ and r_i^1 and r_i^2 are the residual vectors of Algorithm 4.1 and ML(k)BiCGSTAB respectively and $1 \leq i \leq \nu$. Thus if A is nonsingular and recalling that $\psi_1(\lambda) = \rho_1\lambda + 1$, then

$$x_i^2 \in x_0^F + span\{r_0^F, Ar_0^F, \cdots, A^{i-1}r_0^F\}$$

where x_i^2 denotes the approximate solution of ML(k)BiCGSTAB, defined by (4.20) and (4.21), with residual r_i^2 and where $x_0^F = x_0 - \rho_1 v_0$ and $r_0^F = b - Ax_0^F$. If at step i of the ML(k)BiCGSTAB algorithm, we choose $q_1 (= A^0q_1 = p_1) = \psi_1(A^T)\psi_1(A)v_0$

[2] *If the coefficient matrix A is singular, the system (4.10) may have more than one solutions.*

[3] and $q_{i'}(= A^0 q_{i'} = p_{i'}) = \psi_1(A^T) r^2_{i'-1}$ for $2 \le i' \le i$, then

$$(r_0^F)^T r_i^2 \;=\; (\psi_1(A^T)\psi_1(A)v_0)^T r_i^1 \;=\; p_1^T r_i^1 \;=\; 0\,, \qquad 1 \le i\,,$$

and

$$(r_{i'}^2)^T r_i^2 \;=\; (r_{i'}^2)^T \psi_1(A) r_i^1 \;=\; p_{i'+1}^T r_i^1 \;=\; 0\,, \qquad 1 \le i' \le i-1$$

by (4.12). In other words,

$$r_i^2 \perp span\{r_0^F, r_1^2, \cdots, r_{i-1}^2\}\,, \qquad 1 \le i \le \nu\,.$$

As a result, the ML(k)BiCGSTAB ($k \ge \nu$) algorithm with the special choic es for q_i's described above is mathematically equivalent to the FOM algorithm defined by

$$x_i^F \in x_0^F + span\{r_0^F, Ar_0^F, \cdots, A^{i-1}r_0^F\}$$

and

$$r_i^F \perp \{r_0^F, r_1^F, \cdots, r_{i-1}^F\}\,,$$

where the initial guess x_0^F and residual r_0^F are defined as above.

It is quite straightforward to give a preconditioned version of ML(k)BiCGSTAB. Suppose we are solving the right-preconditioned system,

$$AM^{-1}y = b\,, \qquad y = Mx\,.$$

Directly applying the ML(k)BiCGSTAB algorithm to the system $AM^{-1}y = b$ f or the y-variable and then recovering the x-approximation from the y-approximation

[3]Note that ρ_1, the leading coefficient of $\psi_1(\lambda)$, is a function of q_1 according to Steps 4-8 of Algorithm 4.2 and here we suppose the equation $q_1 = \psi_1(A^T)\psi_1(A)v_0$ has solutions for q_1.

with the relation $y_l = Mx_l$ yields the following algorithm.

ALGORITHM 4.3. **ML(k)BiCGSTAB** WITH PRECONDITIONING

1. *Choose an initial guess* x_0 *and* k *vectors* $q_1, q_2, \cdots, q_k$.
2. *Compute* $r_0 = b - Ax_0$ *and set* $g_0 = r_0$.
3. *For* $j = 0, 1, 2, \cdots$
4. $\quad \tilde{g}_{(j-1)k+k} = M^{-1} g_{(j-1)k+k}$;
5. $\quad w_{(j-1)k+k} = A\tilde{g}_{(j-1)k+k}$;
6. $\quad c_{(j-1)k+k} = q_1^T w_{(j-1)k+k}$;
7. $\quad \alpha_{jk+1} = q_1^T r_{(j-1)k+k} / c_{(j-1)k+k}$;
8. $\quad u_{jk+1} = r_{(j-1)k+k} - \alpha_{jk+1} w_{(j-1)k+k}$;
9. $\quad \tilde{u}_{jk+1} = M^{-1} u_{jk+1}$;
10. $\quad \rho_{j+1} = -u_{jk+1}^T A\tilde{u}_{jk+1} / \|A\tilde{u}_{jk+1}\|^2$;
11. $\quad x_{jk+1} = x_{(j-1)k+k} - \rho_{j+1}\tilde{u}_{jk+1} + \alpha_{jk+1}\tilde{g}_{(j-1)k+k}$;
12. $\quad r_{jk+1} = \rho_{j+1} A\tilde{u}_{jk+1} + u_{jk+1}$;
13. $\quad For\ i = 1, 2, \cdots, k$
14. $\quad\quad z_d = u_{jk+i}$; $z_g = r_{jk+i}$; $z_w = 0$;
15. $\quad\quad For\ s = i, \cdots, k-1\ and\ j \geq 1$
16. $\quad\quad\quad \beta_{(j-1)k+s}^{(jk+i)} = -q_{s+1}^T z_d / c_{(j-1)k+s}$;
17. $\quad\quad\quad z_d = z_d + \beta_{(j-1)k+s}^{(jk+i)} d_{(j-1)k+s}$;
18. $\quad\quad\quad z_g = z_g + \beta_{(j-1)k+s}^{(jk+i)} g_{(j-1)k+s}$;
19. $\quad\quad\quad z_w = z_w + \beta_{(j-1)k+s}^{(jk+i)} w_{(j-1)k+s}$;
20. $\quad\quad End$
21. $\quad\quad \beta_{(j-1)k+k}^{(jk+i)} = -\dfrac{q_1^T (r_{jk+i} + \rho_{j+1} z_w)}{\rho_{j+1} c_{(j-1)k+k}}$;
22. $\quad\quad z_g = z_g + \beta_{(j-1)k+k}^{(jk+i)} g_{(j-1)k+k}$;

23. $z_w = \rho_{j+1}\left(z_w + \beta^{(jk+i)}_{(j-1)k+k} w_{(j-1)k+k}\right)$;

24. $z_d = r_{jk+i} + z_w$;

25. $For\, s = 1, \cdots, i-1$

26. $\beta^{(jk+i)}_{jk+s} = -q^T_{s+1} z_d / c_{jk+s}$;

27. $z_d = z_d + \beta^{(jk+i)}_{jk+s} d_{jk+s}$;

28. $z_g = z_g + \beta^{(jk+i)}_{jk+s} g_{jk+s}$;

29. End

30. $d_{jk+i} = z_d - u_{jk+i}$;

31. $g_{jk+i} = z_g + z_w$;

32. $If\, i < k$

33. $c_{jk+i} = q^T_{i+1} d_{jk+i}$;

34. $\alpha_{jk+i+1} = q^T_{i+1} u_{jk+i} / c_{jk+i}$;

35. $u_{jk+i+1} = u_{jk+i} - \alpha_{jk+i+1} d_{jk+i}$;

36. $\tilde{g}_{jk+i} = M^{-1} g_{jk+i}$;

37. $x_{jk+i+1} = x_{jk+i} + \rho_{j+1}\alpha_{jk+i+1}\tilde{g}_{jk+i}$;

38. $w_{jk+i} = A\tilde{g}_{jk+i}$;

39. $r_{jk+i+1} = r_{jk+i} - \rho_{j+1}\alpha_{jk+i+1} w_{jk+i}$;

40. End

41. End

42. End

With suitable changes of variables, it may be shown that both the left and split preconditioning versions of ML(k)BiCGSTAB also lead to Algorithm 4.3 provided that $q_1, q_2, \cdots, q_k$ are appropriately chosen. For the concepts of left, right and split preconditioning, one is referred to [50].

Preconditioning ($M^{-1}v$)	$1+1/k$	*Vector addition*	3
Matvec (Av)	$1+1/k$	*Saxpy*	$2.5k+3.5+1/k$
dot product	$k+2$	*Scalar operation*	$k+3-1/k$
Scalar-vector	1		

Table 4.1: Average cost per step of the preconditioned ML(k)BiCGSTAB.

Each loop of the control variable j in Algorithm 4.3 involves solving $k+1$ systems with coefficient matrix M, $k+1$ matrix-vector multiplications with A, k^2+2k dot products, $2.5k^2+3.5k+1$ saxpy's, $3k$ vector additions, k scalar-vector multiplications and k^2+3k-1 scalar operations. Since there are k steps in each loop of j, the average cost per step can be calculated and is listed in Table 4.1. Regarding the storage, the data $\{q_1,\cdots,q_k\}$, $\{d_{(j-1)k+i},\cdots,d_{jk+i-1}\}$, $\{g_{(j-1)k+i},\cdots,g_{jk+i-1}\}$ and $\{w_{(j-1)k+i},\cdots,w_{jk+i-1}\}$ are used in the process at step $jk+i$ and hence they must be stored. Since they dominate the memory when k is large, the storage of the algorithm is about $4kn$. We note that when $k=1$ the cost is the same as BiCGSTAB's and for large k the cost tends to that of FOM.

4.4 Numerical Experiments

In this section, we shall illustrate the numerical convergence behavior of ML(k)BiCGSTAB. We shall compare ML(k)BiCGSTAB to BiCG, BiCGSTAB and GMRES(m) [48] on a test suite of matrices from the Harwell-Boeing Collection [16]. For the implementation of these latter three methods, we used the versions described in [8]. All

the experiments were run in MATLAB 4.2c on a SUN Sparc station with machine precision about 10^{-16}. As for the initial guesses and right-hand sides, we always chose $x_0 = 0$ and $b = [1, 1, \cdots, 1]^T$. For the initial vectors $q_1, q_2, \cdots, q_k$ in the ML(k)BiCGSTAB algorithm, we first chose k random vectors with independent and identically distributed entries from a normal distribution with mean 0 and variance 1 and then made them orthogonal to each other by using the modified Gram-Schmidt algorithm [31]. With q_i's chosen randomly, the conditions of Theorem 4.1 are satisfied in most of cases and therefore we expect that the chance of the algorithm encountering breakdown is very rare. The iteration was stopped as soon as the true relative error $\|b - Ax_l\|_2/\|b\|_2$ was less than 10^{-7}. Finally, all the figures plot the true relative residual versus the number of matrix-vector multiplies taken.

We ran all four methods, on a representative group of matrices from the Harwell-Boeing collection. The results are summarized in Table 4.2. We observe that, in terms of number of matvec's, ML(50)BiCGSTAB and ML(100)BiCGSTAB are always better than the other four methods, at least for this collection of matrices. The only exception is the matrix *watt2* where only BiCG and GMRES converged. We can also see that ML(k)BiCGSTAB for $k = 25$ is almost as good as $k = 50$, whereas $k = 100$ does not give much improvement over $k = 50$ in most cases. We believe that the improvement of ML(k)BiCGSTAB over BiCGSTAB can be attributed to the use of multiple starting vectors. In prin-

ciple, ML(k)BiCGSTAB can never be better than full GMRES, but as we can see from the table, it can be much better than *restarted* GMRES. We can also see from the table that ML(k)BiCGSTAB and BiCG tend to converge and diverge more or less on the same subset of matrices, but ML(k)BiCGSTAB typically requires many fewer matvec's when they all converge.

Next, we present the convergence history for three matrices from Table 4.2. These matrices are described below. We have used ML(30)BiCGSTAB in these examples.

Example 1. This example is the first matrix named IMPCOL D from the CHEMIMP group of the Harwell-Boeing collection. The order of the matrix is 425 and it has 1339 nonzero entries. In this example, no preconditioner was used and the convergence curves are plotted in Figure 4.1. BiCGSTAB encounter a breakdown after 450 matvec's.

Example 2. The matrix is the second one named ORSIRR1 from the OILGEN group. The order of the matrix is 1030 and the number of nonzero entries is 6858. We first run the algorithms without preconditioning and then with ILU(0) preconditioning. The results are shown in Figures 4.2 and 4.3 respectively.

Example 3. This is the HOR131 matrix from the NNCENG group. The order is 434 and it has 4710 nonzero entries. The ILU(0) preconditioner was used and the result is plotted in Figure 4.4.

We observe that when all four methods converge (as in Examples 2 and 3),

Matrix	*Order*	*BiCG*	*BiCGSTAB*	*GMRES(100)*	*ML(25)*	*ML(50)*	*ML(100)*
1138bus	1138	4748	5872	–	1966	1384	1395
bcspwr06	1454	5810	14246	–	3167	2720	1899
bcsstk08	1074	15844	–	–	3859	1902	1242
bcsstk14	1806	29294	–	–	–	13315	6336
bcsstk19	817	–	–	–	–	–	–
bcsstm27	1224	–	–	–	–	–	–
can1054	1054	9908	–	–	4058	3126	2606
dwt1005	1005	1178	2934	–	673	625	645
eris1176	1176	1426	1530	1197	698	532	499
fs5414	541	2738	2640	–	728	469	403
gr3030	900	76	52	38	40	40	40
gre1107	1107	–	*b*	–	–	8676	3262
hor131	434	–	–	–	1945	1268	1048
impcold	425	–	*b*	–	1619	916	597
jagmesh2	1009	1726	2958	–	1152	995	1129
jpwh991	991	100	58	49	55	53	55
lns511	511	–	–	–	–	–	–
lock1074	1068	*o*	–	–	–	–	–
lshp1270	1270	2492	4458	–	1628	1591	1445
mahindas	1258	–	*b*	–	–	–	–
mcfe	765	–	–	–	–	–	–
nnc1374	1374	–	*b*	–	–	–	–
nos3	960	494	384	1968	251	249	246
orsirr1	1030	2068	3318	1270	838	781	772
plat1919	1919	–	–	–	–	–	–
pores2	1224	–	–	–	–	–	–
saylr3	1000	*o*	–	–	*o*	*o*	*o*
sherman2	1080	–	*b*	–	–	–	–
watt2	1856	19406	–	1131	–	–	–
west0989	989	–	–	–	–	–	–

Table 4.2: Comparison of Methods on a Representative Group of Matrices from the Harwell-Boeing Collection. ML(25), ML(50) and ML(100) stand for ML(25)BiCGSTAB, ML(50)BiCGSTAB and ML(100)BiCGSTAB respectively. The numbers in the table are number of matvec's. "-" means no convergence within $20n$ matvec's for BiCG and $10n$ for the other methods, "b" denotes breakdown, "o" denotes overflow.

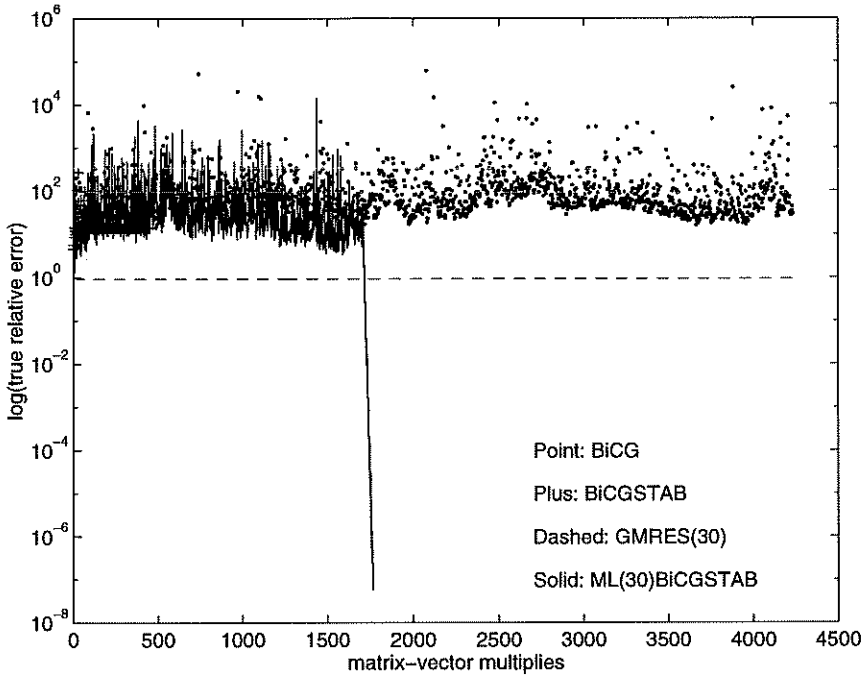

Figure 4.1: Example 1: with no preconditioning

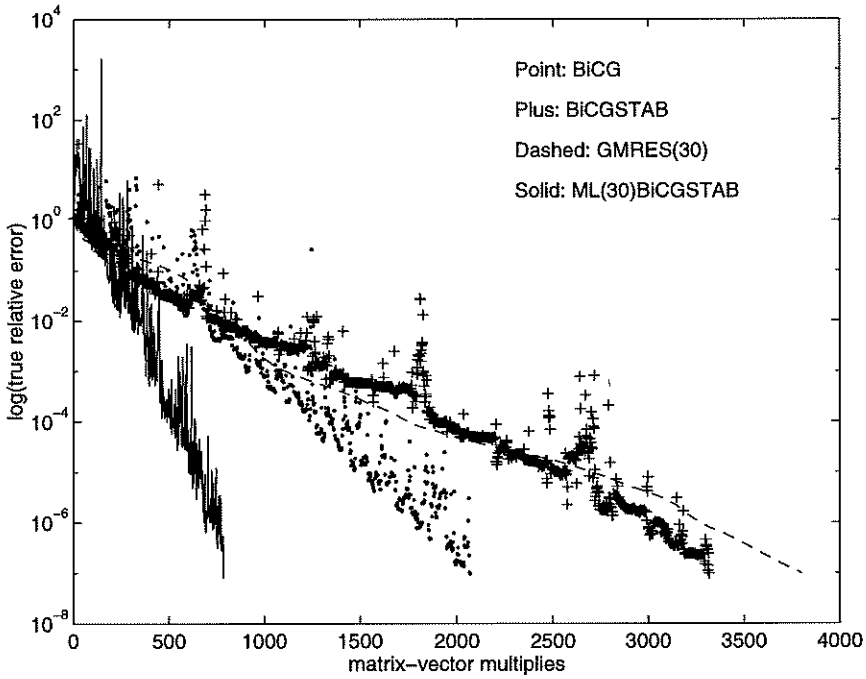

Figure 4.2: Example 2: with no preconditioning

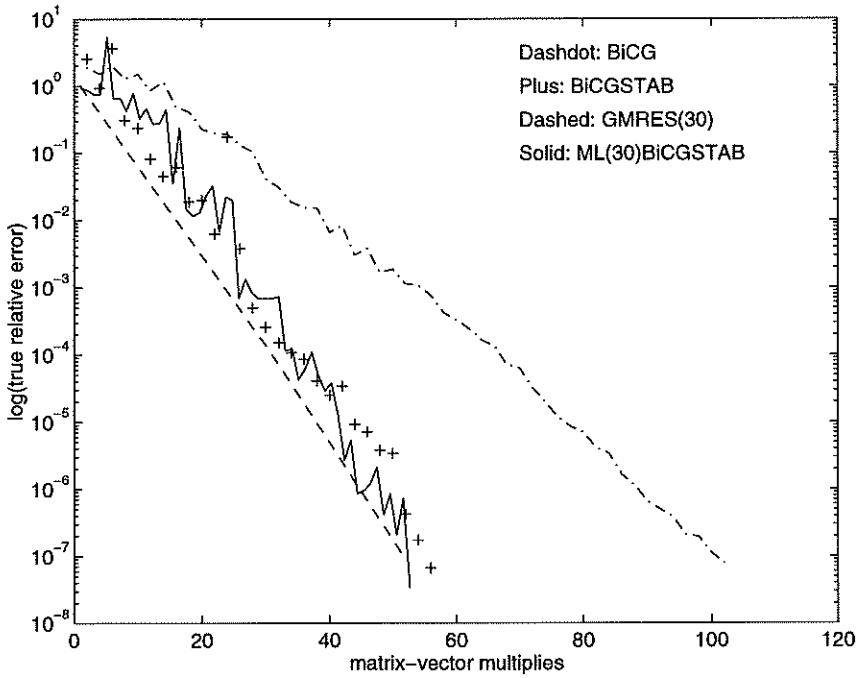

Figure 4.3: Example 2: with ILU(0) preconditioning

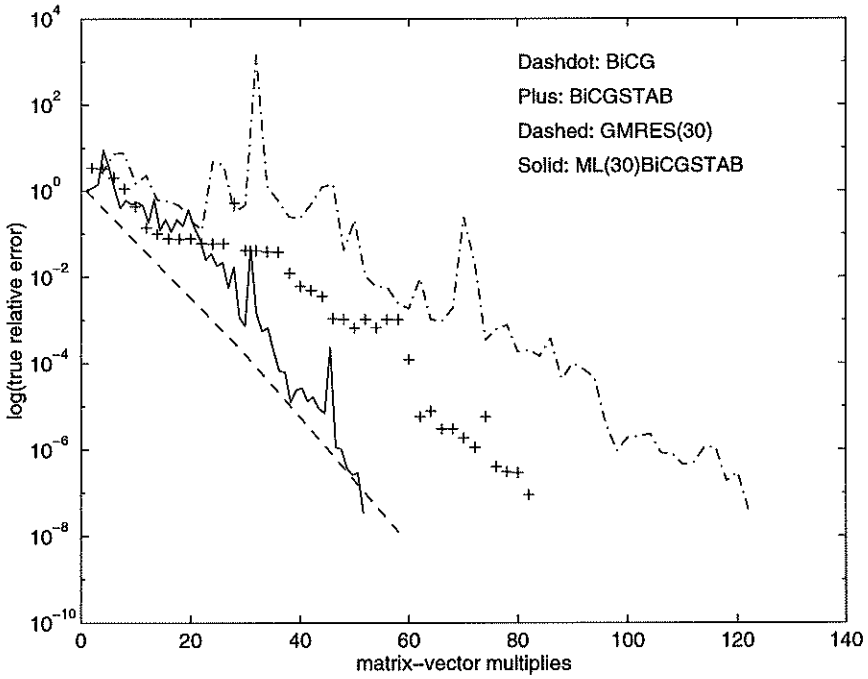

Figure 4.4: Example 3: with ILU(0) preconditioning

ML(30)BiCGSTAB requires approximately the same or fewer matvec's than the other three methods. In fact, it can be significantly faster than the other three methods, as in Example 2. Moreover, ML(30)BiCGSTAB manages to converge when the other three methods fail, as in Example 1.

Finally, we present some numerical results to demonstrate the dependence of the performance of ML(k)BiCGSTAB on the value of k. In Figure 4.5, we plot the number of matvec's (scaled by $1/10n$) versus k for the two matrices ORSIRR1 and HOR131. In order to illustrate the improvement of ML(k)BiCGSTAB over BiCGSTAB for $k > 1$, we plot for $k = 1$ the number of matvec's for BiCGSTAB instead of for the mathematically equivalent ML(1)BiCGSTAB. We observe that for both matrices there is a dramatic improvement in performance as k increases from 1. This behavior is typical for the matrices that we have tested and this can be partially observed from Table 4.2. Thus the advantage of ML(k)BiCGSTAB can be realized even for small values of k. On the other hand, we can also see that for large enough values of k (e.g. $k > 10$ for ORSIRR1 and $k > 30$ for HOR131), the performance is not sensitive to the value of k. Thus, it is not crucial to choose an optimal value of k as long as k is large enough. We have also found that the performance is not sensitive to the specific choice of the random starting vectors q_i's, provided that k is large enough. However, we should caution that the performance could be sensitive to the choice of q_i's for *small* values of k.

More testings are of course needed to better understand and assess the per-

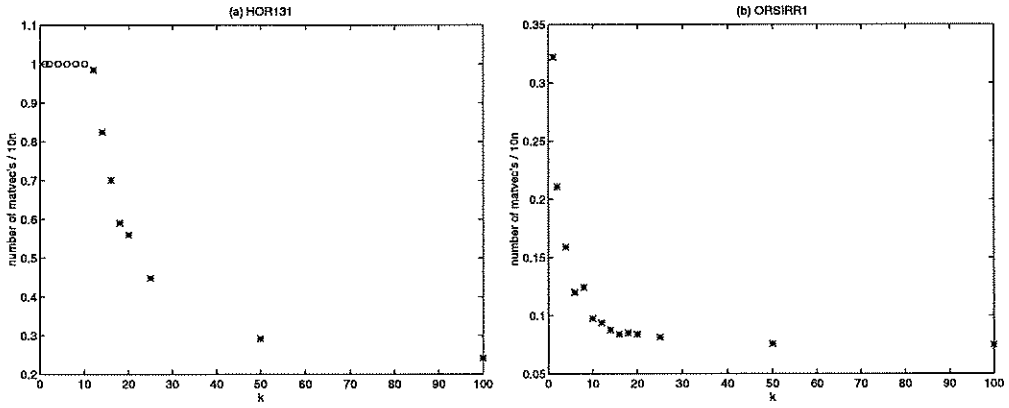

Figure 4.5: Number of matvec's / $10n$ vs. k for the matrices HOR131 (a) and ORSIRR1 (b). "o" denotes no convergence within $10n$ matvec's. For $k = 1$, we plotted the number of matvec's for BiCGSTAB. Note that there is a dramatic improvement in performance as k increases from 1, but that for $k \geq 30$, the performance is not sensitive to k.

formance of ML(k)BiCGSTAB, but we hope we have at least demonstrated the potential advantages of this new method.

4.5 Appendix

Here we give the detailed derivation of the coefficients α_l's and β_l's of the ML(k)BiCGSTAB algorithm.

$$\alpha_{jk+1} = \frac{\eta_j^{(j)} p_{jk+1}^T r_{(j-1)k+k}}{\eta_j^{(j)} p_{jk+1}^T A g_{(j-1)k+k}} = \frac{\sum_{s=0}^{j} \eta_s^{(j)} p_{sk+1}^T r_{(j-1)k+k}}{\sum_{s=0}^{j} \eta_s^{(j)} p_{sk+1}^T A g_{(j-1)k+k}}$$

$$= \frac{\sum_{s=0}^{j} \eta_s^{(j)} q_1^T A^s r_{(j-1)k+k}}{\sum_{s=0}^{j} \eta_s^{(j)} q_1^T A^{s+1} g_{(j-1)k+k}} = \frac{q_1^T \psi_j(A) r_{(j-1)k+k}}{q_1^T A \psi_j(A) g_{(j-1)k+k}}$$

$$= \frac{q_1^T \pi_{(j-1)k+k}}{q_1^T A \omega_{(j-1)k+k}}$$

for $0 \leq j, i = 1$;

$$\alpha_{jk+i+1} = \frac{\eta_j^{(j)} p_{jk+i+1}^T r_{jk+i}}{\eta_j^{(j)} p_{jk+i+1}^T A g_{jk+i}} = \frac{\sum_{s=0}^{j} \eta_s^{(j)} p_{sk+i+1}^T r_{jk+i}}{\sum_{s=0}^{j} \eta_s^{(j)} p_{sk+i+1}^T A g_{jk+i}}$$

$$= \frac{\sum_{s=0}^{j} \eta_s^{(j)} q_{i+1}^T A^s r_{jk+i}}{\sum_{s=0}^{j} \eta_s^{(j)} q_{i+1}^T A^{s+1} g_{jk+i}} = \frac{q_{i+1}^T \psi_j(A) r_{jk+i}}{q_{i+1}^T A \psi_j(A) g_{jk+i}}$$

$$= \frac{\rho_{j+1} q_{i+1}^T \psi_j(A) r_{jk+i}}{q_{i+1}^T (\psi_{j+1}(A) - \psi_j(A)) g_{jk+i}} = \frac{\rho_{j+1} q_{i+1}^T \tilde{\pi}_{jk+i}}{q_{i+1}^T (\omega_{jk+i} - \tilde{\omega}_{jk+i})}$$

for $0 \leq j, 1 \leq i < k$;

$$\beta_0^{(i)} = -\frac{p_1^T(\eta_1^{(1)} A r_i + \eta_0^{(1)} r_i)}{\eta_1^{(1)} p_1^T A g_0} = -\frac{q_1^T \psi_1(A) r_i}{\rho_1 q_1^T A g_0}$$

$$= -\frac{q_1^T \pi_i}{\rho_1 q_1^T A \omega_0}$$

for $1 \leq i \leq k, j = 0$;

$$\beta_s^{(i)} = -\frac{p_{s+1}^T \left(\eta_1^{(1)} A r_i + \beta_0^{(i)} \eta_1^{(1)} A g_0 + \sum_{t=1}^{s-1} \beta_t^{(i)} \eta_1^{(1)} A g_t\right)}{\eta_1^{(1)} p_{s+1}^T A g_s}$$

$$= -\frac{p_{s+1}^T \left(\psi_1(A) r_i + \beta_0^{(i)} \eta_1^{(1)} A g_0 + \sum_{t=1}^{s-1} \beta_t^{(i)} (\psi_1(A) - \psi_0(A)) g_t\right)}{p_{s+1}^T (\psi_1(A) - \psi_0(A)) g_s}$$

$$= -\frac{q_{s+1}^T \left(\pi_i + \beta_0^{(i)} \rho_1 A \omega_0 + \sum_{t=1}^{s-1} \beta_t^{(i)} (\omega_t - \tilde{\omega}_t)\right)}{q_{s+1}^T (\omega_s - \tilde{\omega}_s)}$$

for $1 \leq s < i \leq k, j = 0$;

$$\beta_{(j-1)k+s}^{(jk+i)} = -\frac{\eta_j^{(j)} p_{(j-1)k+s+1}^T A r_{jk+i} + \eta_j^{(j)} \sum_{t=i}^{s-1} \beta_{(j-1)k+t}^{(jk+i)} p_{(j-1)k+s+1}^T A g_{(j-1)k+t}}{\eta_j^{(j)} p_{(j-1)k+s+1}^T A g_{(j-1)k+s}}$$

$$= -\frac{\eta_j^{(j)} p_{jk+s+1}^T r_{jk+i} + \rho_j \sum_{t=i}^{s-1} \beta_{(j-1)k+t}^{(jk+i)} \eta_{j-1}^{(j-1)} p_{(j-1)k+s+1}^T A g_{(j-1)k+t}}{\rho_j \eta_{j-1}^{(j-1)} p_{(j-1)k+s+1}^T A g_{(j-1)k+s}}$$

$$= -\frac{\sum_{u=0}^{j} \eta_u^{(j)} p_{uk+s+1}^T r_{jk+i} + \rho_j \sum_{t=i}^{s-1} \beta_{(j-1)k+t}^{(jk+i)} \sum_{u=0}^{j-1} \eta_u^{(j-1)} p_{uk+s+1}^T A g_{(j-1)k+t}}{\rho_j \sum_{u=0}^{j-1} \eta_u^{(j-1)} p_{uk+s+1}^T A g_{(j-1)k+s}}$$

$$= -\frac{\sum_{u=0}^{j} \eta_u^{(j)} q_{s+1}^T A^u r_{jk+i} + \rho_j \sum_{t=i}^{s-1} \beta_{(j-1)k+t}^{(jk+i)} \sum_{u=0}^{j-1} \eta_u^{(j-1)} q_{s+1}^T A^{u+1} g_{(j-1)k+t}}{\rho_j \sum_{u=0}^{j-1} \eta_u^{(j-1)} q_{s+1}^T A^{u+1} g_{(j-1)k+s}}$$

$$= -\frac{q_{s+1}^T \psi_j(A) r_{jk+i} + \rho_j \sum_{t=i}^{s-1} \beta_{(j-1)k+t}^{(jk+i)} q_{s+1}^T A \psi_{j-1}(A) g_{(j-1)k+t}}{\rho_j q_{s+1}^T A \psi_{j-1}(A) g_{(j-1)k+s}}$$

$$= -\frac{q_{s+1}^T \psi_j(A) r_{jk+i} + \sum_{t=i}^{s-1} \beta_{(j-1)k+t}^{(jk+i)} q_{s+1}^T (\psi_j(A) - \psi_{j-1}(A)) g_{(j-1)k+t}}{q_{s+1}^T (\psi_j(A) - \psi_{j-1}(A)) g_{(j-1)k+s}}$$

$$= -\frac{q_{s+1}^T \left(\tilde{\pi}_{jk+i} + \sum_{t=i}^{s-1} \beta_{(j-1)k+t}^{(jk+i)} (\omega_{(j-1)k+t} - \tilde{\omega}_{(j-1)k+t}) \right)}{q_{s+1}^T (\omega_{(j-1)k+s} - \tilde{\omega}_{(j-1)k+s})}$$

for $1 \leq j, 1 \leq i \leq s \leq k-1$;

$$\beta_{(j-1)k+k}^{(jk+i)} = -\frac{\eta_{j+1}^{(j+1)} p_{jk+1}^T A r_{jk+i} + \sum_{t=i}^{k-1} \beta_{(j-1)k+t}^{(jk+i)} \eta_{j+1}^{(j+1)} p_{jk+1}^T A g_{(j-1)k+t}}{\eta_{j+1}^{(j+1)} p_{jk+1}^T A g_{(j-1)k+k}}$$

$$= -\frac{\eta_{j+1}^{(j+1)} p_{(j+1)k+1}^T r_{jk+i} + \rho_{j+1} \sum_{t=i}^{k-1} \beta_{(j-1)k+t}^{(jk+i)} \eta_j^{(j)} p_{jk+1}^T A g_{(j-1)k+t}}{\rho_{j+1} \eta_j^{(j)} p_{jk+1}^T A g_{(j-1)k+k}}$$

$$= -\frac{\sum_{u=0}^{j+1} \eta_u^{(j+1)} p_{uk+1}^T r_{jk+i} + \rho_{j+1} \sum_{t=i}^{k-1} \beta_{(j-1)k+t}^{(jk+i)} \sum_{u=0}^{j} \eta_u^{(j)} p_{uk+1}^T A g_{(j-1)k+t}}{\rho_{j+1} \sum_{u=0}^{j} \eta_u^{(j)} p_{uk+1}^T A g_{(j-1)k+k}}$$

$$= -\frac{\sum_{u=0}^{j+1} \eta_u^{(j+1)} q_1^T A^u r_{jk+i} + \rho_{j+1} \sum_{t=i}^{k-1} \beta_{(j-1)k+t}^{(jk+i)} \sum_{u=0}^{j} \eta_u^{(j)} q_1^T A^{u+1} g_{(j-1)k+t}}{\rho_{j+1} \sum_{u=0}^{j} \eta_u^{(j)} q_1^T A^{u+1} g_{(j-1)k+k}}$$

$$= -\frac{q_1^T \psi_{j+1}(A) r_{jk+i} + \rho_{j+1} \sum_{t=i}^{k-1} \beta_{(j-1)k+t}^{(jk+i)} q_1^T A \psi_j(A) g_{(j-1)k+t}}{\rho_{j+1} q_1^T A \psi_j(A) g_{(j-1)k+k}}$$

$$= -\frac{q_1^T \left(\pi_{jk+i} + \rho_{j+1} \sum_{t=i}^{k-1} \beta_{(j-1)k+t}^{(jk+i)} A \omega_{(j-1)k+t} \right)}{\rho_{j+1} q_1^T A \omega_{(j-1)k+k}}$$

for $1 \leq j, 1 \leq i \leq k$;

$$\beta_{jk+s}^{(jk+i)} = -\left(\eta_{j+1}^{(j+1)} p_{jk+s+1}^T A r_{jk+i} + \eta_{j+1}^{(j+1)} \sum_{t=i}^{k} \beta_{(j-1)k+t}^{(jk+i)} p_{jk+s+1}^T A g_{(j-1)k+t} + \right.$$

$$\eta_{j+1}^{(j+1)} \sum_{t=1}^{s-1} \beta_{jk+t}^{(jk+i)} p_{jk+s+1}^T A g_{jk+t} \Bigg) \Big/ \eta_{j+1}^{(j+1)} p_{jk+s+1}^T A g_{jk+s}$$

$$= -\Bigg(\eta_{j+1}^{(j+1)} p_{(j+1)k+s+1}^T r_{jk+i} + \rho_{j+1} \sum_{t=i}^{k} \beta_{(j-1)k+t}^{(jk+i)} \eta_j^{(j)} p_{jk+s+1}^T A g_{(j-1)k+t} +$$

$$\rho_{j+1} \sum_{t=1}^{s-1} \beta_{jk+t}^{(jk+i)} \eta_j^{(j)} p_{jk+s+1}^T A g_{jk+t} \Bigg) \Big/ \rho_{j+1} \eta_j^{(j)} p_{jk+s+1}^T A g_{jk+s}$$

$$= -\Bigg(\sum_{u=0}^{j+1} \eta_u^{(j+1)} p_{uk+s+1}^T r_{jk+i} + \rho_{j+1} \sum_{t=i}^{k} \beta_{(j-1)k+t}^{(jk+i)} \sum_{u=0}^{j} \eta_u^{(j)} p_{uk+s+1}^T A g_{(j-1)k+t} +$$

$$\rho_{j+1} \sum_{t=1}^{s-1} \beta_{jk+t}^{(jk+i)} \sum_{u=0}^{j} \eta_u^{(j)} p_{uk+s+1}^T A g_{jk+t} \Bigg) \Big/ \rho_{j+1} \sum_{u=0}^{j} \eta_u^{(j)} p_{uk+s+1}^T A g_{jk+s}$$

$$= -\Bigg(\sum_{u=0}^{j+1} \eta_u^{(j+1)} q_{s+1}^T A^u r_{jk+i} + \rho_{j+1} \sum_{t=i}^{k} \beta_{(j-1)k+t}^{(jk+i)} \sum_{u=0}^{j} \eta_u^{(j)} q_{s+1}^T A^{u+1} g_{(j-1)k+t} +$$

$$\rho_{j+1} \sum_{t=1}^{s-1} \beta_{jk+t}^{(jk+i)} \sum_{u=0}^{j} \eta_u^{(j)} q_{s+1}^T A^{u+1} g_{jk+t} \Bigg) \Big/ \rho_{j+1} \sum_{u=0}^{j} \eta_u^{(j)} q_{s+1}^T A^{u+1} g_{jk+s}$$

$$= -\Bigg(q_{s+1}^T \psi_{j+1}(A) r_{jk+i} + \rho_{j+1} \sum_{t=i}^{k} \beta_{(j-1)k+t}^{(jk+i)} q_{s+1}^T A \psi_j(A) g_{(j-1)k+t} +$$

$$\rho_{j+1} \sum_{t=1}^{s-1} \beta_{jk+t}^{(jk+i)} q_{s+1}^T A \psi_j(A) g_{jk+t} \Bigg) \Big/ \rho_{j+1} q_{s+1}^T A \psi_j(A) g_{jk+s}$$

$$= -\Bigg(q_{s+1}^T \psi_{j+1}(A) r_{jk+i} + \rho_{j+1} \sum_{t=i}^{k} \beta_{(j-1)k+t}^{(jk+i)} q_{s+1}^T A \psi_j(A) g_{(j-1)k+t} +$$

$$\sum_{t=1}^{s-1} \beta_{jk+t}^{(jk+i)} q_{s+1}^T(\psi_{j+1}(A) - \psi_j(A)) g_{jk+t} \Bigg) \Big/ q_{s+1}^T(\psi_{j+1}(A) - \psi_j(A)) g_{jk+s}$$

$$= -\frac{q_{s+1}^T \left(\pi_{jk+i} + \rho_{j+1} \sum_{t=i}^{k} \beta_{(j-1)k+t}^{(jk+i)} A\omega_{(j-1)k+t} + \sum_{t=1}^{s-1} \beta_{jk+t}^{(jk+i)} (\omega_{jk+t} - \tilde{\omega}_{jk+t}) \right)}{q_{s+1}^T(\omega_{jk+s} - \tilde{\omega}_{jk+s})}$$

for $1 \leq j, 1 \leq s < i \leq k$.

Bibliography

[1] J. Aliaga, D. Boley, R. Freund and V. Hernández, *A Lanczos-type method for multiple starting vectors,* Numerical Analysis Manuscript No. 96-18, Bell Laboratories, Murray Hill, NJ, 1996. (Available on WWW at **http://cm.bell-labs.com/cs/doc/96**)

[2] T. W. Anderson, *An Introduction to Multivariate Statistical Analysis,* John Wiley & Sons, New York, 1958.

[3] W. E. Arnoldi, *The principle of minimized iteration in the solution of the matrix eigenvalue problem,* Quart. Appl. Math., 9:17-29, 1951.

[4] S. F. Ashby, T. A. Manteuffel and P. E. Saylor, *A taxonomy for conjugate gradient methods,* SIAM J. Numer. Anal., 27 (1990), pp. 1542-1568.

[5] O. Axelsson, *Conjugate gradient type methods for unsymmetric and inconsistent systems of linear equations,* Lin. Alg. Appl., 29 (1980), pp. 1-16.

[6] R. E. Bank and T. F. Chan, *An analysis of the composite step biconjugate gradient algorithm for solving nonsymmetric systems,* Numer. Math., 66 (1993), pp. 295-319.

[7] ——, *A composite step bi-conjugate gradient algorithm for solving nonsymmetric systems,* UCLA CAM Tech. Report 93-21 (1993).

[8] R. Barrett, *et. al., Templates for the solution of linear systems: building blocks for iterative methods,* SIAM Publications, 1994.

[9] P. Billingsley, *Probability and Measure,* John Wiley & Sons, New York, 1979.

[10] D. L. Boley, S. Elhay, G. H. Golub and M. H. Gutknecht, *Nonsymmetric Lanczos and finding orthogonal polynomials associated with indefinite weights,* Numer. Algorithms 1 (1991), 21-44.

[11] D. L. Boley and G. H. Golub, *The nonsymmetric Lanczos algorithm and controllability,* Systems Control Lett. 16 (1991), 97-105.

[12] C. Brezinski and H. Sadok, *Avoiding breakdown in the CGS algorithm,* Numer. Alg. 1(1991), pp.199-206.

[13] P. N. Brown and A. C. Hindmarsh, *Matrix-free methods for stiff systems of ODEs,* SIAM J. Numer. Anal., 23 (1986), pp. 610-638.

[14] T. F. Chan, L. de Pillis and H. A. van der Vorst, *A transpose-free squared Lanczos algorithm and application to solving nonsymmetric linear systems,* Technical Report CAM 91-17, Department of Mathematics, University of California, Los Angeles, CA, 1991.

[15] H. Cramér, *Mathematical Methods of Statistics,* Princeton University Press, 1946.

[16] I. S. Duff, R. G. Grimes and J. G. Lewis, *Sparse matrix test problems,* ACM Trans. Math. Softw., 15 (1989), pp. 1-14.

[17] A. Edelman, *Eigenvalues and condition numbers of random matrices,* SIAM J. Matrix Anal. Appl., 9 (1988), 543-560.

[18] ——, *Eigenvalues and condition numbers of random matrices,* Ph.D. Thesis, Department of Mathematics, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139.

[19] ——, *The distribution and moments of the smallest eigenvalue of a random matrix of Wishart type,* Linear Alg. Appl. 159 (1991) 55-80.

[20] A. Edelman and W. Mascarenhas, *On the complete pivoting conjecture for a Hadamard matrix of order 12,* J of Linear and Multilinear Algebra 38 (1995), 181-187.

[21] V. Faber and T. Manteuffel, *Necessary and sufficient conditions for the existence of a conjugate gradient method,* SIAM J. Numer. Anal., 21 (1984), pp. 352-361.

[22] R. Fletcher, *Conjugate gradient methods for indefinite systems,* In G. A. Watson, editor, Proceedings of the Dundee Biennal Conference on Numerical

Analysis 1974, pages 73-89. Springer Verlag, New York, 1975.

[23] L. V. Foster, *The probability of large diagonal elements in the QR factorization,* SIAM J. Sci. Stat. Comput., 11 (1990), 531-544.

[24] ——, *Gaussian elimination with partial pivoting can fail in practice,* SIAM J. Matrix Anal. Appl., 15 (1994), 1354-1362.

[25] R. W. Freund, *A transpose-free quasi-minimal residual algorithm for non-Hermitian linear systems,* SIAM J. Sci. Stat. Comput., 14(1993), pp. 470-482.

[26] R. W. Freund, *The look-ahead Lanczos process for nonsymmetric matrices and its applications,* Proceedings of the Cornelius Lanczos International Centenary Conference (J. D. Brown, M. T. Chu, D. C. Ellison, and R. J. Plemmons, eds.), SIAM, Philadelphia, 1994, pp. 33-47.

[27] R. W. Freund, M. H. Gutknecht and N. M. Nachtigal, *An implementation of the look-ahead Lanczos algorithm for non-Hermitian matrices,* SIAM J. Sci. Comput. 14 (1993), 137-158.

[28] R. Freund and M. Malhotra, *A block-QMR algorithm for non-Hermitian linear systems with multiple right-hand sides,* Numerical Analysis Manuscript No. 95-09, AT&T Bell Laboratories, Murray Hill, NJ, 1995. (Available on WWW at **http://cm.bell-labs.com/cs/doc/95**)

[29] R. W. Freund and N. M. Nachtigal, *QMR: a quasi-minimal residual method for non-Hermitian linear systems,* Numerische Mathematik 60 (1991), pp. 315-339.

[30] R. W. Freund and N. M. Nachtigal, *A look ahead TFQMR method,* Presented at the Cornelius Lanczos International Centenary Conference, Raleigh, NC, December 1993.

[31] G. H. Golub and C. F. Van Loan, *Matrix Computations,* second edition, The Johns Hopkins University Press (Baltimore), 1989.

[32] M. H. Gutknecht, *A completed theory of the unsymmetric Lanczos process and related algorithms, part I,* SIAM J. Matrix Anal. Appl. 13 (1992), 594-639.

[33] ——, *Variants of BiCGStab for matrices with complex spectrum,* SIAM J. Sci. Comput. 14, 1020-1033, 1993.

[34] ——, *A completed theory of the unsymmetric Lanczos process and related algorithms, part II,* SIAM J. Matrix Anal. Appl. 15 (1994), 15-58.

[35] W. Hackbusch, *Iterative Solution of Large Sparse Systems of Equations,* Springer, New York, 1994.

[36] M. R. Hestenes and E. Stiefel, *Methods of conjugate gradients for solving linear systems,* J. Res. Nat. Bur. Stand., 49(1952), pp. 409-436.

[37] H. T. Kung, *Why systolic architectures?,* Computer, 15(1):37-46, January 1982.

[38] C. Lanczos, *An iteration method for the solution of the eigenvalue problem of linear differential and integral operators,* J. Res. Nat. Bur. Standards 45 (1950), 255-282.

[39] ——, *Solution of systems of linear equations by minimized iterations,* Journal of Research of the National Bureau of Standards, 49:33-53, 1952.

[40] D. Lê, *Benefits and costs of solving Gaussian elimination related problems using randomization on block, systolic, and recursive block decomposition schemes,* Ph.D. Thesis, 1995, Department of Computer Science, University of California, Los Angeles, CA 90095.

[41] A. Meyer, *The concept of special inner products for deriving new conjugate gradient-like solvers for non-symmetric sparse linear systems,* Num. Lin. Alg. Appl., 1 (1994), pp. 129-140.

[42] D. S. Parker, *Random butterfly transformations with applications in computational linear algebra,* Technical Report CSD-950023, UCLA Computer Science Department, 1995.

[43] D. S. Parker and B. Pierce, *The randomizing FFT: an alternative to pivoting in Gaussian elimination,* Technical Report CSD-950037, UCLA Computer Science Department, 1995.

[44] B. N. Parlett, *Reduction to tridiagonal form and minimal realizations,* SIAM J. Matrix Anal. Appl. 13 (1992), 567-593.

[45] B. N. Parlett, D. R. Taylor and Z. A. Lin, *A look-ahead Lanczos algorithm for unsymmetric matrices,* Math. Comp. 44 (1985), 105-124.

[46] A. Ruhe, *Implementation aspects of band Lanczos algorithms for computation of eigenvalues of large sparse symmetric matrices,* Math. Comp. 33 (1979), 680-687.

[47] Y. Saad, *Practical use of some Krylov subspace methods for solving indefinite and unsymmetric linear systems,* SIAM J. Sci. Statist. Comput., 5:203-228, 1984.

[48] ——, *GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems,* SIAM J. Sci. Statist. Comput., 7 (1986), pp. 856-869.

[49] ——, *A flexible inner-outer preconditioned GMRES algorithm,* SIAM J. Sci. Stat. Comput., 14 (1993), pp. 461-469.

[50] ——, *Iterative methods for sparse linear systems,* PWS Publishing Company, 1996.

[51] Y. Saad and K. Wu, *DQGMRES: A direct quasi-minimal residual algorithm based on incomplete orthogonalization,* Num. Lin. Alg. Appl., 3 (1996), pp. 329-343.

[52] J. W. Silverstein, *The smallest eigenvalue of a large-dimensional Wishart matrix,* Ann. Prob. 13 (1985), 1364-1368.

[53] G. L. G. Sleijpen and D. R. Fokkema, *BiCGSTAB(k) for linear equations involving insymmetric matrices with complex spectrum,* Electronic Trans. Numer. Anal. 1, 11-32, 1993.

[54] G. L. G. Sleijpen and H. A. van der Vorst, *Reliable updated residuals in hybrid Bi-CG methods,* Computing, 56:141-163, 1996.

[55] P. Sonneveld, *CGS, a fast Lanczos-type solver for nonsymmetric linear systems,* SIAM Journal on Scientific and Statistical Computing, 10(1):36-52, 1989.

[56] D. R. Taylor, *Analysis of the look ahead Lanczos algorithm,* Ph.D. Thesis, Department of Mathematics, University of California, Berkeley, CA, 1982.

[57] L. N. Trefethen and R. S. Schreiber, *Average-case stability of Gaussian elimination,* SIAM J. Matrix Anal. Appl., Vol. 11, No. 3, pp. 335-360, July 1990.

[58] H. F. Trotter, *Eigenvalue distributions of large Hermitian matrices; Wigner's semi-circle law and a theorem of Kac, Murdock, and Szegő,* Advances in Math. 54 (1984), 67-82.

[59] A. M. Turing, *Rounding-off errors in matrix processes,* Quart. J. Mech. Appl. Math., 1(1948), pp. 287-308.

[60] H. A. van der Vorst, *Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of non-symmetric linear systems,* SIAM Journal on Scientific and Statistical Computing, 12:631-644, 1992.

[61] J. von Neumann and H. H. Goldstine, *Numerical inverting of matrices of high order,* Bull. Amer. Math. Soc., 53 (1947), pp. 1021-1099; von Neumann's *Collected Works,* Vol. 5, A. H. Taub, ed., Pergamon, Elmsford, NY, 1963.

[62] ——, *Numerical inverting of matrices of high order, Part II,* Proc. Amer. Math. Soc., 2 (1951), pp. 188-202; von Neumann's *Collected Works,* Vol. 5, A. H. Taub, ed., Pergamon Press, Elmsford, NY, 1963.

[63] J. H. Wilkinson, *Error analysis of direct methods of matrix inversion,* J. Assoc. Comput. Mach., 8 (1961), pp. 281-330.

[64] ——, *The Algebraic Eigenvalue Problem,* Clarendon Press, Oxford, 1965.

[65] D. M. Young and K. C. Jea, *Generalized conjugate-gradient acceleration of nonsymmetrizable iterative methods,* Lin. Alg. Appl., 34 (1980), pp. 159-194.