Mixed Product Krylov Subspace Methods for Solving Nonsymmetric Linear Systems

by
Tracy L. Ewen

A Thesis Submitted to the Faculty of Graduate Studies in
Partial Fulfilment of the Requirements
for the Degree of

MASTER OF SCIENCE

in the Department of Mathematics
Univesity of Manitoba
Winnipeg, Manitoba

0-612-51707-1

THE UNIVERSITY OF MANITOBA

FACULTY OF GRADUATE STUDIES

*****

COPYRIGHT PERMISSION PAGE

Mixed Product Krylov Subspace Methods for Solving Nonsymmetric Linear Systems

BY

Tracy L. Ewen

A Thesis/Practicum submitted to the Faculty of Graduate Studies of The University of Manitoba in partial fulfillment of the requirements of the degree

of

Master of Science

TRACY L. EWEN ©1999

Abstract

Krylov subspace methods are among the most important iterative techniques currently used for solving large sparse linear systems. They are used for solving systems in which the coefficient matrix A is either symmetric or nonsymmetric. While the methods used for symmetric systems are well understood, the nonsymmetric methods are more difficult to analyse theoretically and recently much research has been focused on these methods. In this thesis, a review of the current methods which are built upon the Krylov subspace is presented. New mixed methods for nonsymmetric systems are then derived which are based on either the CGS or BiCGStab algorithms and which allow switching from these methods to BiCG. While maintaining the fast convergence properties of CGS and BiCGStab, the motivation for these new methods is to improve upon certain undesirable properties inherent in either method. In CGS it is often the case that in the early stages of convergence the residuals become very large, a behaviour which may lead to irregular convergence and in some cases even divergence of the method. Although BiCGStab was developed and successful in smoothing this convergence, in certain types of problems this method may result in stagnation of the residuals. The mixed methods are developed to try and remedy these problems by switching to the more stable BiCG method when certain switching criteria are reached. Numerical testing is carried out and examples given to show the benefits of the new methods.

Contents

List of Figures

Chapter 1

Introduction

1.1 Direct and Iterative Methods

Methods for solving linear systems include both direct and iterative ones. Direct methods include those that we are familiar with from our first introductions to linear algebra, namely Gaussian elimination where the solution is obtained through a finite procedure. Direct methods, such as Gaussian Elimination, cannot efficiently solve many matrices that arise in practice. Although these methods are theoretically exact, we must also consider that when we solve these systems on computers, we will never obtain an 'exact' solution due to the roundoff error during these computations. Iterative methods, on the other hand, approximate a solution through a series of iterations. The iterations are terminated when some termination criteria is satisfied for the 'approximated' solution vector x_{n+1}. If we consider solving a system using either a direct or iterative method, we may think it would be better to obtain the 'exact' solution, rather than using an iterative method, as the error of an iterative solution might be far worse than the extra work required in computing an exact answer. If we consider the cost of computing the solution of $Ax = b$ with a direct method such as this, the computational flops (floating point operations) will be $O(m^3)$ (where m is the dimension of the matrix). In the worst case scenario, an iterative method will cost $O(m^3)$, which is usually never the case, and at best, for a sparse or structured system, may only cost $O(m)$. So, given that we will expect error in both methods, the decrease in cost of implementing an iterative scheme is almost always better given a large sparse system. Many of these examples arise from discretizations of partial differential equations; these resulting matrices have certain properties that cannot be adequately exploited using the direct methods. Furthermore, they are often sparse, containing very few nonzero entries and when solving these systems with Gaussian Elimination, it is hard to preserve the sparsity, with many zero entries filled in during the process of elimination. We can take advantage of these sparse matrices by solving them using iterative techniques which use matrix-vector multiplication only. The storage for this operation is just $O(m)$, as there are usually only a few nonzero entries each row, as opposed to a dense matrix-vector operation, which would require $2m^2$ operations.

1.2 Development of Krylov Subspace Methods

Most efficient iterative methods currently available, use a projection process which allows us to extract a solution vector of a linear system from a subspace. The most current and popular iterative techniques use this idea and we extract the approximate solution from the Krylov subspace. Thus, given an initial guess vector x_0, we can create a subspace K which is built upon an orthogonal basis and perform a projection process to extract the solution vector x_n.

Several different Krylov subspace methods exist. These can be broken down into two major groups; those for solving linear systems in which the coefficient matrix A is symmetric and those in which A is nonsymmetric. Below, an overview is given of the current methods used today, from the historical beginnings to some of the many variations which exist in the current literature.

Krylov methods have stemmed from early beginnings, although the development of using them as iterative methods did not truly begin until several decades later. Lanczos introduced the first projection process for symmetric systems. Shortly after in 1951, Arnoldi [1] developed his orthogonal projection process for non-Hermitian matrices. This process was first introduced as a means of reducing a dense matrix into Hessenberg form and was also later discovered to be a good technique for approximating the eigenvalues of large sparse matrices. Unlike the Arnoldi method, the Lanczos algorithm uses a three-term recurrence to solve the symmetric system, and so is a more elegant version. From both methods stemmed several of the current methods used today for solving linear systems. The most important and popular symmetric Krylov method, the Conjugate Gradient method was introduced independently by both Lanczos in 1952 [17] and Hestenes and Steifel [15]. Interestingly, the Conjugate Gradient method was initially developed as a direct solution technique and was found to give poorer results than current direct methods at the time. In exact arithmetic, this method should converge in n steps, although due to loss of orthogonality upon vector formation during the algorithm, this method did not always give adequate results and was abandoned for over two decades. In the early 1970's, it was found that this loss of orthogonality did not actually prevent convergence [21] and in a sense, was rediscovered as an iterative technique. The 1980's saw many developments in the new class of methods for solving nonsymmetric systems. In 1986 Saad and Schultz [22] introduced the Generalized Minimum Residual Method (GMRES) which was based on the Arnoldi process. Unlike the Arnoldi method[1] the GMRES method finds an approximate solution by solving a least squares problem (see [21] for details).

Several variants were developed from both FOM and GMRES. For variations which stemmed from both, the reader is referred to Saad's text on Iterative Methods for Sparse Linear Systems [21] which gives an overview and derivations of most of these methods. Further studies investigated block methods based on the methods above, for cases in which one may want to exploit the fast memory storage with blocks rather than working with single vectors. As one of the first such methods, Ruhe [20] developed a block method for the symmetric case, based on the Lanczos algorithm in 1979. More recent work has focused on nonsymmetric block methods.

The main body of literature which exists for solving nonsymmetric linear systems with methods based on the Krylov subspace developed much later than those of the early CG beginnings, and in fact is an area of much current research. The methods are based again on the Lanczos algorithm, although in this case due to the nonsymmetric nature of A, the methods are based on the formation of a biorthogonal sequence rather than an orthogonal one as in the symmetric case. The first of such methods was introduced by

[1]of which there are several different methods referenced in the literature, in this review we mean the Full Orthogonalized Method of Arnoldi's, here within referred to as FOM

Lanczos in 1952 [17] and later, in the present form by Fletcher [10] in 1974. This method was introduced as the Biconjugate Gradient Method (or BiCG) and is still currently one of the most reliable methods when dealing with nonsymmetric matrices. Also in this class of methods is the Quasi-Minimal Residual method (QMR), which differs from BiCG in that the approximation is obtained by solving a least squares problem (as GMRES differs from the Arnoldi method). Although both of these methods work quite well, they both rely upon using the transpose of A to generate biorthogonal bases. This is sometimes undesirable in practice as in some applications the transpose of A may not be available if A is not available explicitly (see [7] for further details). Sonneveld [27] then introduced the Conjugate Gradient Squared Method (CGS), which does not rely on using the transpose of A as BiCG does and improves the convergence by squaring the residual polynomial. This method was an important development as it allows us to consider these methods in terms of orthogonal polynomials rather than in vector or matrix forms. The CGS method however may have drawbacks when applied to certain types of problems and sometimes develops large residuals near the beginning; as a result convergence may be irregular or the method may fail to converge at all (as we shall see by numerical experiment in later chapters). The Biconjugate Gradient Stabilized Method developed by Van der Vorst [29] in 1992, aimed at treating this unpredictable behaviour in CGS by providing a different polynomial in the residual construction, which was chosen to smooth out the convergence. Rather than squaring the BiCG polynomial, as was done in CGS, BiCGStab uses the BiCG polynomial together with a 'stabilizing polynomial' to smooth the convergence of CGS and also to keep the improved speed of convergence over BiCG. We will see the derivations for both of these methods in the following chapter. Although this method has also proved to be very successful, it may also suffer from breakdown and the method may stagnate with certain problem types.

All three methods, BiCG, CGS and BiCGStab have benefits depending on the problem which they are being applied to. It is for this reason that much research has recently been devoted; we wish to keep the beneficial properties of these methods but hope to increase the numbers of problems which we can apply these methods to. One other aspect which keeps us from major improvements regarding these methods is that unlike the symmetric cases, these methods are theoretically more difficult to analyse and as such have made it more difficult to understand the numerical behaviour. The best tool we have so far in terms of analysing and thus understanding these methods, is through numerical experimentation. Recent attempts at improving the CGS methods include, for example, that done by Fokkema et al. [11] and Chan and Ye [6]. In [6], a mixed method was developed which allows switching between the CGS and BiCGStab methods and thus avoids the potential drawbacks during implementation. Gutknecht [13] attempted to remedy this stagnation, with his development of BiCGStab2 in 1991. This method uses a second degree polynomial and was certainly an improvement to the BiCGStab method although it still suffers from breakdown at times. Sleijpen and Fokkema [23] furthered this study by developing the BiCGStab(l) method in 1993, which generalized Gutknecht's method in a sense, and combined GMRES(l) with BiCG. More recent work which has focused on BiCGStab and BiCG methods include, for example, [26], [25], and [24].

In this thesis, we have attempted to improve upon both the CGS and BiCGStab methods, as is the aim of much current research. We first provide an overview and algorithms for the most important methods noted above. Two mixed methods, similar to the mixed method of Chan and Ye [6] but using BiCG in both, are then derived. The first is a mixed BiCG-CGS algorithm and the second a BiCG-BiCGStab algorithm. Both use a switching variable which depends upon the iteration number and thus allows us to switch freely between the two methods when a certain criterion is reached. With these methods, we attempt to improve the drawbacks that one may encounter in CGS or BiCGStab, namely the instabilities and stagnation that may occur, by switching to the more stable BiCG method in such a situation. We also give an overview and results of numerical testing for a shifted CGS method based on work done by Fokkema et. al [11]. Although we have implemented this method as they have suggested in their paper, it could also be implemented as a special case of the mixed BiCGStab-CGS method as suggested in [6]. As we will see, the shifted CGS and new mixed methods improve the convergence in most cases and for the cases where this is not the case we offer a discussion and possibilities for further study.

Chapter 2

Theoretical Background

2.1 Basic Iterative Methods

Basic iterative methods, including the following familiar methods: Gauss-Seidel method, Jacobi method, SOR (SSOR) and the Alternating Direction Methods, begin with a given approximation x_0 to the solution, and by modifying certain components of the approximate solution at each iteration, reach convergence. For example, in each Jacobi iteration, the $(n+1)^{th}$ component of the next approximation is determined so as to get rid of the n^{th} component of the residual vector $r_{n+1} = b - Ax_n$. In this way, r_n is used as a search vector and x_{n+1} is updated using r_n.

Further to these iterative methods are the Krylov Subspace methods, which are used for solving large linear systems, and extract approximate solutions from the Krylov subspace $K_n = \text{span}\{b, Ab, A^2b, \cdots, A^nb\}$.

We can think of a simple scheme in the following way. Suppose we are given (or guess) an initial solution vector x_0 and we wish to find a good approximate solution.

We can approximate the solution through a series of iterates, $n = 0, 1, \ldots$ using

$$x_{n+1} = x_n + M^{-1}(b - Ax_n) \tag{2.1}$$

where M is a matrix called a preconditioner, usually chosen so that $M^{-1}A$ is close to the identity in some sense, and the residual vector is that defined as $r_n = b - Ax_n$. Depending on the chosen form of M we will end up with one of the methods mentioned above. For M equal to the diagonal of A we have Jacobi iteration, for M equal to the lower triangle of A we have the Gauss-Seidel method. In general, we can write the following algorithm

Algorithm 1 *Basic Iteration*

```
Given $x_0$, compute $r_0 = b - Ax_0$
    solve $z_o = M^{-1}r_0$
    for $n = 0, 1, 2, ...$
    $x_{n+1} = x_n + z_n$
    $r_{n+1} = b - Ax_{n+1}$
    $z_{n+1} = M^{-1}r_{n+1}$
    end
```

To establish necessary and sufficient conditions for the convergence of this algorithm, we consider the error $\varepsilon_n = x_n - x$ where $x = A^{-1}b$. It is easy to see that

$$\begin{aligned} \varepsilon_{n+1} = x_{n+1} - x &= x_n - x + M^{-1}A(x - x_n) & (2.2) \\ &= (I - M^{-1}A)\varepsilon_n = \cdots = (I - M^{-1}A)^k \varepsilon_0 & (2.3) \end{aligned}$$

taking norms on both sides of this equation we obtain

$$||\varepsilon_{n+1}|| \leq ||(I - M^{-1}A)^k|| \, ||\varepsilon_0|| \tag{2.4}$$

and we can state the following.

Lemma 1 *The iteration of Algorithm 1 produces the vector x_n which will approach x and the error will approach zero for every initial x_0 if and only if*

$$\rho(I - M^{-1}A) < 1$$

where $\rho(B)$ is the spectral radius of B, i.e. the largest eigenvalue in absolute value.

From (2.3) we can see that if $||(I - M^{-1}A)|| < 1$, then the error will be reduced by at least this factor at each iteration.

2.2 Projection Methods

Krylov subspace methods extract a solution vector x_n from a subspace that is optimal in some sense, through a projection process.

If we wish to extract the approximation from a subspace K then, in general n constraints must be imposed in order to extract such an approximation. We will define these constraints by imposing n orthogonality conditions. More specifically, we will constrain the residual vector $r_n = b - Ax_n$ to be orthogonal to n linearly independent vectors. This defines another subspace L, which can be called the subspace of constraints. When the subspace L is the same as K, we have an orthogonal projection process, otherwise if L and K are different, we have an oblique projection method.

In general, given an $m \times m$ real matrix A and two subspaces L and K of dimension n, we can define a projection technique such that the approximate solution x_n is found where $x_n - x_0$ is in K and the new residual vector is orthogonal to L or, find $x_n - x_0 \in K$, such that the Petrov Galerkin condition holds,

$$r_n = b - Ax_n \perp L.$$

Or, we can define x_n in the following way,

$$x_n = x_0 + \delta, \quad \delta \in K$$

so that we have,

$$(r_n, q) = 0, \quad \forall q \in L$$

In matrix representation we can consider finding the approximate solution as

$$x = x_0 + Py \tag{2.5}$$

where $P = [p_1, p_2, \ldots, p_n] \in \Re^{m \times n}$ whose column vectors form a basis of K. In the same way, let $Q = [q_1, q_2, \ldots, q_n] \in \Re^{m \times n}$ whose column vectors form a basis of L. Then the orthogonality conditions give,

$$Q^T A P y = Q^T r_0$$

where $Q^T A P \in \Re^{n \times n}$

2.3 Symmetric Krylov Subspace Methods

As a first overview, the process occurs as follows: consider an initial guess vector x_0, if this is not available to us, we may just choose $x_0 = 0$ as an initial guess. Thus we may take the first approximation to the solution:

$$x_1 \in \text{span}\{r_0\}$$

or where our initial guess vector $x_0 = 0$ then we have $r_0 = b - Ax_0 = b$. Next, we compute the matrix-vector product Ab and find the next approximation as some linear combination of b and Ab:

$$x_2 \in \text{span}\{b, Ab\}.$$

This process continues until some convergence criteria is reached and at step $n+1$, the approximation satisfies

$$x_{n+1} \in \text{span}\{b, Ab, A^2 b, \cdots, A^n b\}$$

The following methods are based on projection processes, orthogonal or oblique, onto a Krylov subspace,

$$K_n \in \text{span}\{b, Ab, A^2b, \cdots, A^{n-1}b\}$$

this subspace is spanned by vectors of the form $p(A)b$ where $p(A)$ is a polynomial in A of degree less than n. We are seeking the solution $x_n = A^{-1}b$ and approximate this solution by $p(A)b$.

The Lanczos algorithm builds orthogonal bases of the Krylov subspace K_n for symmetric matrices. Through a three term recurrence, we construct an orthonormal basis $q_1, q_2, q_3, \cdots, q_n$ of Lanczos vectors for

$$\{q_1, Aq_1, A^2q_1, \cdots, A^{n-1}q_1\}$$

such that the following matrix relations hold,

$$\begin{aligned} AQ_n &= Q_nT_n + \beta_{n+1}q_{n+1}e_n^T \\ Q_n^TQ_n &= I \\ Q_n^TAQ_n &= T_n \end{aligned}$$

where $Q_n = [q_1, q_2, \cdots, q_n]$ is $n \times m$ and the tridiagonal matrix T_n is given by

$$\left|\begin{array}{ccccc} \alpha_1 & \beta_2 & & & \\ \beta_2 & \alpha_2 & \beta_3 & & \\ & \cdot & \cdot & \cdot & \\ & & \beta_{n-1} & \alpha_{n-1} & \beta_n \\ & & & \beta_n & \alpha_n \end{array}\right|$$

if we let $q_1 = r_0/||r_0||$ (or when $x_0 = 0$ then $r_0 = b_0$), and $\beta = ||r_0||$ and from the matrix relations above we have

$$Q_n^T r_0 = Q_n^T(\beta q_1) = \beta e_1$$

Where the approximate solution x_n is given by,

$$x_n = x_0 + Q_n y_n \tag{2.6}$$

as in (2.5), or when $x_0 = 0$ as an initial guess

$$x_n = Q_n y_n$$

and

$$y_n = T_n^{-1}(\beta e_1).$$

In another approach, the Minimal Residual Method, we find the approximate solution x_n as above, but we find y_n by solving the least squares problem, $||\beta e_1 - T_{n+1} y||$ (for an overview of this method see [14]).

We can now write the Lanczos algorithm which generates the column vectors to form the matrix Q_n, and the coefficients α and β to form the tridiagonal matrix T_n so that we can find the approximate solution to the linear system as in (2.5).

Algorithm 2 *the Lanczos Algorithm*

```
Choose an initial vector $q_1$ such that $||q_1|| = 1$
Set $\beta_1 = 0; q_0 = 0$
    For $n = 1, 2, ......$ until convergence
        $p_n = Aq_n - \beta_n q_{n-1}$
        $\alpha_n = (p_n, q_n)$
        $p_n = p_n - \alpha_n q_n$
        $\beta_{n+1} = ||p_n||_2$
        $q_{n+1} = p_n / \beta_{n+1}$
    end for
```

Thus, using the Lanczos algorithm, we can derive Krylov subspace methods for symmetric matrices. If we wish to find an approximate solution to the linear system $Ax = b$, we can consider an orthogonal projection process in which, $L = K = K_n(A, r_0)$. Such an approximate solution, x_n can be found in the affine space of $x_0 + K_n$ by imposing the Galerkin condition[1],

$$b - Ax_n \perp K_n$$

This projection process ensures x_n is the optimal approximation from K_n.

[1]when $L = K$ the Petrov-Galerkin conditions are called the Galerkin conditions

Theorem 1 *Let x_n be the approximate solution obtained from an orthogonal projection process onto K_n and the exact solution to the original problem be denoted by x. The imposed Galerkin conditions are necessary and sufficient for the following minimization relation to hold*

$$||x_n - x||_A = min_{u \in K_n} ||u - x||_A. \tag{2.7}$$

2.3.1 Conjugate Gradient Method

From the Lanczos process we can derive the Conjugate Gradient method, used for solving symmetric systems. The process developed above leads to the approximate solution x_{n+1} given as $x_{n+1} = x_0 + Q_n y_n$ where $y_n = T_n^{-1}(\beta e_1)$. This requires inverting T_n at each step. Fortunately, x_{n+1} and the corresponding r_{n+1} can be computed by a recurrence, which is the conjugate gradient method.

In matrix notation, let

$$T_n = L_n U_n$$

where L_n and U_n are lower and upper banded matrices respectively. We have,

$$x_{n+1} = x_0 + Q_n U_n^{-1} L_n^{-1}(\beta e_1)$$

let

$$P_n = Q_n U_n^{-1}$$

and

$$z_n = L_n^{-1} \beta e_1$$

then

$$x_{n+1} = x_0 + P_n z_n$$

Proposition 1 *The Lanczos process produces $r_n = b - Ax_n$ and a search vector p_n for $n = 0, 1, \ldots$ and,*

1. the residual vectors are orthogonal to each other, $(r_i, r_j) = 0$ for $i \neq j$, and

2. the auxiliary vectors p_i form an A-conjugate set, $(Ap_i, p_j) = 0$ for $i \neq j$.

With the orthogonality and conjugacy conditions we can now derive the Conjugate Gradient algorithm. The solution vector can be expressed as,

$$x_{n+1} = x_n + \alpha_n p_n$$

and the residuals satisfy the recurrence relation

$$r_{n+1} = r_n - \alpha_n A p_n$$

and if the residual vectors are orthogonal then

$$(r_{n+1}, r_n) = 0$$

$$(r_n - \alpha_n A p_n, r_n) = 0$$

and as a result we have

$$\alpha_n = \frac{(r_n, r_n)}{(Ap_n, r_n)}$$

Where the next search vector p_{n+1} is a linear combination of r_{n+1} and p_n, so that we have the following recurrence relation holds

$$p_{n+1} = r_{n+1} + \beta_n p_n \tag{2.8}$$

A first consequence of the above relation is

$$\begin{aligned}(Ap_n, r_n) &= (Ap_n, p_n - \beta_{n-1}p_{n-1}) \\ &= (Ap_n, p_n)\end{aligned}$$

because Ap_n is orthogonal to p_{n-1} we can write

$$\alpha_n = \frac{(r_n, r_n)}{(Ap_n, p_n)}.$$

In addition, if we write p_{n+1} as in (2.6) which is orthogonal to Ap_n we have

$$\beta_n = -\frac{(r_{n+1}, Ap_n)}{(p_n, Ap_n)}.$$

where from the recurrence for r_{n+1} we obtain

$$Ap_n = -\frac{1}{\alpha_n}(r_{n+1} - r_n)$$

and so

$$\beta_n = \frac{(r_{n+1}, r_{n+1})}{(r_n, r_n)}.$$

We can now write the Conjugate Gradient Algorithm,

Algorithm 3 *Conjugate Gradient*

```
Input $x_0$
Initialize $r_0 = p_0 = b - Ax_0$
    For $n = 0, 1, 2, ......$ until convergence
        $w_n = Ap_n$
        $\rho_n = r_n^T r_n$
        $\sigma_n = p_n^T Ap_n = (p_n, w_n)$
        $\alpha_n = \frac{\rho_n}{\sigma_n}$
        $x_{n+1} = x_n + \alpha_n p_n$
        $r_{n+1} = r_n - \alpha_n Ap_n$
        $\beta_n = \frac{\rho_{n+1}}{\rho_n}$
        $p_{n+1} = r_{n+1} + \beta_n p_n$
    end for
```

It is important to note here that the scalars α_n and β_n of this algorithm are different from those of the Lanczos algorithm. We can also see that each iteration of the CG algorithm will require one matrix vector operation, Ap_n and four vectors to be stored: x, p, Ap and r.

2.4 Nonsymmetric Krylov Subspace Methods

The first Krylov subspace methods introduced to solve nonsymmetric linear systems were based on the normal equations [21], $A^TAx = A^Tb$. Although these methods are quite robust, in most examples arising from the discretization of partial differential equations, the convergence is too slow, and when the problem is somewhat ill-conditioned these methods will fail to converge within a reasonable number of iterations unless a good preconditioner is used [14]. A more direct Krylov method for solving nonsymmetric problems uses a biorthogonalization process due to Lanczos [17].

Given a nonsymmetric coefficient matrix A, it constructs a pair of biorthogonal bases, $q_1, q_2, q_3, \cdots, q_n$ and $p_1, p_2, p_3, \cdots, p_n$ for

$$K_n(A, p) \in \text{span}\{p_1, Ap_1, A^2p_1, \cdots, A^{n-1}p_1\}$$

and

$$K_n(A^T, q) \in \text{span}\{q_1, A^Tq_1, (A^T)^2q_1, \cdots, (A^T)^{n-1}q_1\}$$

so that the following orthogonality condition holds

$$(q_j, p_i) = \delta_{ij}, \ for\ 1 \leq i, j \leq m.$$

Similar to the symmetric case, the following matrix relations hold,

$$\begin{aligned} AP_n &= P_nT_n + \hat{p}_{n+1}e_n^T \\ A^TQ_n &= Q_nT_n^T + \hat{q}_{n+1}e_n^T \\ Q_n^TAP_n &= T_n \end{aligned}$$

Where we have replaced the product $\delta_{n+1}p_{n+1}$ with $\hat{p}_{n+1}$ and $\beta_{n+1}q_{n+1}$ with $\hat{q}_{n+1}$.

Where the tridiagonal matrix T_n is given by

$$\begin{vmatrix} \alpha_1 & \beta_2 & & & \\ \delta_2 & \alpha_2 & \beta_3 & & \\ & \cdot & \cdot & \cdot & \\ & & \delta_{n-1} & \alpha_{n-1} & \beta_n \\ & & & \delta_n & \alpha_n \end{vmatrix}$$

and Q_n and P_n are $n \times n$ matrices formed by the column vectors, q_i and p_i for $i = 1, \dots n$. Thus the Lanczos algorithm for the nonsymmetric case[2] generates the vectors p_{n+1} and q_{n+1} using T_n where α_n is given by the inner product of p_n and q_n and the scalars β and δ are chosen such that they satisfy the relation

$$\delta_{n+1}\beta_{n+1} = (\hat{p}_{n+1}, \hat{q}_{n+1}).$$

We can write the algorithm as follows

Algorithm 4 *the Lanczos Algorithm (for nonsymmetric matrices)*

```
Choose two initial vectors $p_1$, $q_1$ such that $(p_1, q_1) = 1$
Set $\beta_1 = \delta_1 = 0$; $q_0 = p_0 = 0$
    For $n = 1, 2, ......$ until convergence
        $\alpha_n = (Ap_n, q_n)$
        $\hat{p}_{n+1} = Ap_n - \alpha_n p_n - \beta_n p_{n-1}$
```

[2]here after referred to as simply the Lanczos algorithm.

```
    $\hat{q}_{n+1} = A^T q_n - \alpha_n q_n - \delta_n q_{n-1}$
    $\delta_{n+1} = |(\hat{p}_{n+1}, \hat{q}_{n+1})|^{1/2}$
    $\beta_{n+1} = \frac{(\hat{p}_{n+1}, \hat{q}_{n+1})}{\delta_{n+1}}$
    $q_{n+1} = \frac{\hat{q}_{n+1}}{\beta_{n+1}}$
    $p_{n+1} = \frac{\hat{p}_{n+1}}{\delta_{n+1}}$
end for
```

As in the symmetric case, if we wish to apply the algorithm to a linear system to generate the approximate solution x_n where

$$x_n = x_0 + Q_n y_n$$

and

$$y_n = T_n^{-1}(\beta e_1)$$

where $\beta = ||r_0||$, we first generate the Lanczos vectors $q_1, q_2, \ldots, q_n,\ \ p_1, p_2, \ldots, p_n$ and the tridiagonal matrix T_n to find the approximate solution.

2.4.1 Biconjugate Gradient Method (BiCG)

The Biconjugate Gradient Method was introduced by Lanczos [17] and can be derived from the Lanczos biorthogonalization process in the same way that the Conjugate Gradient method was derived for the symmetric case. The algorithm solves the original system, $Ax = b$, with an approximation x_n while also solving a dual linear system, $A^T x = b$ with A^T. In most cases we ignore the formation of this dual system, however the transpose of A must still be formed.

If r_n and p_n are defined from P_n and T_n in a similar way as in 2.3.1, and $\tilde{r}_n$ and $\tilde{p}_n$ are defined from Q_n and T_n, then they satisfy the biorthogonality and conjugacy conditions given as

$$(r_n, \tilde{r}_m) = 0 \quad for \quad n \neq m$$

$$(Ap_n, \tilde{p}_m) = 0 \quad for \quad n \neq m$$

where r_n and p_n are in the same direction as p_{n+1} and $q_1 = r_0/||r_0||$, and $\tilde{r}_n$ and $\tilde{p}_n$ are in the direction of q_{n+1}, we write the BiCG recurrence in the same way as the Conjugate Gradient algorithm, although here we must include $\tilde{r}_n$ and $\tilde{p}_n$.

Algorithm 5 *Biconjugate Gradient*

```
Input $x_0$
Initialize $r_0 = p_0 = b - Ax_0$
Choose $\tilde{r}_0$ such that $\tilde{r}_0^T r_0 \neq 0,\ \tilde{p}_0 = \tilde{r}_0$.
$\rho_0 = \tilde{r}_0^T r_0$
    For $n = 0, 1, 2, ......$ until convergence
        $w_n = Ap_n$
        $\rho_n = r_n^T \tilde{r}_n$
        $\sigma_n = \tilde{p}_n^T Ap_n = (\tilde{p}_n, w_n)$
        $\alpha_n = \frac{\rho_n}{\sigma_n}$
        $x_{n+1} = x_n + \alpha_n p_n$
        $r_{n+1} = r_n - \alpha_n Ap_n$
        $\tilde{r}_{n+1} = \tilde{r}_n - \alpha_n A^T \tilde{p}_n$
        $\beta_n = \frac{\rho_{n+1}}{\rho_n}$
        $p_{n+1} = r_{n+1} + \beta_n p_n$
        $\tilde{p}_{n+1} = \tilde{r}_{n+1} + \beta_n \tilde{p}_n$
    end for
```

Here we see that we will require 2 matrix vector operations for each iteration namely Ap_n and $A^T\tilde{p}_n$. Thus we will require an extra vector to be stored over the CG method in the symmetric case.

Here, it is easier to begin using the theory of orthogonal polynomials to describe the following Lanczos-type algorithms and infact is how Lanczos had introduced his iterative scheme in the 1950's [17]. We can see that the approximations obtained from a Krylov subspace method are of the form $A^{-1}b \approx x_{n+1} = x_0 + h_n(A)r_0$ where h_n is a polynomial of degree n. In the simplest case where $x_0 = 0$ the solution is approximated by $h_n(A)b$ or,

$$A^{-1}b \approx h_n(A)b.$$

Then $r_n = b - Ax_n = r_0 - Ah_{n-1}(A)r_0 = P_n(A)r_0$. We call P_n the nth BiCG polynomial corresponding to r_n. Similarly

$$p_n = T_n(A)r_0.$$

for some polynomial T_n. Note that P_n and T_n are of degree n. Then, from the BiCG recurrence $r_{n+1} = r_n - \alpha_n A p_n$, we have

$$P_{n+1}(t) = P_n(t) - \alpha_n t T_n(t), \tag{2.9}$$

and from $p_{n+1} = r_{n+1} + \beta_{n+1} p_n$, we have

$$T_{n+1}(t) = P_{n+1}(t) + \beta_{n+1} T_n(t), \tag{2.10}$$

The biorthogonality of $r_n, \tilde{r}_n$ and the biconjugacy property of $p_n, \tilde{p}_n$ lead to the following orthogonality properties

$$(p(A^T)\tilde{r}_0, P_n(A)r_0) = 0 \quad \text{and} \quad (p(A^T)\tilde{r}_0, AT_n(A)r_0) = 0 \tag{2.11}$$

where p is any polynomial of degree less than n, we can deduce from the formulas for α_n and β_n that

$$\alpha_n = \frac{(P_n(A^T)\tilde{r}_0, P_n(A)r_0)}{(T_n(A^T)\tilde{r}_0, AT_n(A)r_0)} \quad \text{and} \quad \beta_{n+1} = \frac{(P_{n+1}(A^T)\tilde{r}_0, P_{n+1}(A)r_0)}{(P_n(A^T)\tilde{r}_0, P_n(A)r_0)}.$$

The methods that follow are developed using polynomial representation which we have introduced here for BiCG. The recurrence formulations are also based on the BiCG recurrences of equations (2.9) and (2.10).

2.4.2 Conjugate Gradient Squared Method (CGS)

To improve BiCG, Sonneveld developed the Conjugate Gradient Squared algorithm in 1989 [27]. The improved algorithm avoids using the transpose of A and accelerates the convergence[3] by squaring the BiCG polynomials.

CGS constructs an approximation x_n such that the residual is given by,

$$r_n = b - Ax_n = P_n^2(A)r_0.$$

We can derive the recurrence relations by squaring those from BiCG to obtain

$$P_{n+1}^2 = P_n^2 - 2\alpha_n t P_n T_n + \alpha_n^2 t^2 T_n^2$$

$$T_{n+1}^2 = P_{n+1}^2 + 2\beta_{n+1} P_{n+1} T_n + \beta_{n+1}^2 T_n^2$$

We can see that we will require extra products namely, $P_n T_n$ and $P_{n+1} T_n$. However $P_n T_n$ can be expressed as $P_n^2 + \beta_n P_n T_{n-1}$ and so does not need to be defined explicitly. Thus we define the recurrence relations in terms of the following polynomials,

$$\begin{aligned} r_n &= P_n^2(A)r_0 \\ p_n &= T_n^2(A)r_0 \\ s_n &= P_{n+1}(A)T_n(A)r_0 \end{aligned}$$

[3]Reasons for developing transpose free methods are discussed in Chan et al. [7]. In many cases it may be difficult and expensive to explicitly store A^T in order to compute $A^T v$ and it may also be difficult to compute A^T, depending on whether or not A is computed explicitly.

And in finding the BiCG parameters from the orthogonality conditions, we have

$$\rho_n = (P_n(A^T)\tilde{r}_0, P_n(A)r_0) = (\tilde{r}_0, P_n^2(A)r_0) = (\tilde{r}_0, r_n)$$

and similarly we can show this for σ_n and thus we will be able to obtain α_n and β_{n+1} for CGS. We can now form the CGS algorithm with the desired properties.

Algorithm 6 *Conjugate Gradient Squared*

```
Input $x_0$
Initialize $r_0 = p_0 = b - Ax_0$
Choose $\tilde{r}_n$ such that $\tilde{r}_0^T r_0 \neq 0$
$\rho_0 = \tilde{r}_0^T r_0$
   For $n = 0, 1, 2, ......$ until convergence
      $w_n = Ap_n$
      $\rho_n = r_n^T \tilde{r}_0$
      $\sigma_n = \tilde{p}_n^T Ap_0$
      $\alpha_n = \frac{\rho_n}{\sigma_n}$
      $s_n = t_n - \alpha_n Ap_n$
      $r_{n+1} = r_n - \alpha_n A(t_n + s_n)$
      $\beta_n = \frac{\rho_{n+1}}{\rho_n}$
      $t_{n+1} = r_{n+1} + \beta_n s_n$
      $p_{n+1} = t_{n+1} + \beta_n(s_n + \beta_{n+1} p_n)$
      $x_{n+1} = x_n + \alpha_n(t_n + s_n)$
```

```
end for
```

We note that in deriving the recurrence relations we must construct an auxiliary vector, given in our algorithm as $t_n = r_n + \beta_{n-1} s_n$.

We see that there are no matrix-vector products with the transpose of A, as in BiCG, and that we do not have to explicitly form $\tilde{r}_{n+1}$ or $\tilde{p}_{n+1}$. Instead, two matrix vector products with the matrix A are performed, to produce a reduction with P_n^2, i.e., $r_n = P_n^2(A)r_0$, and we should expect that the new algorithm will converge twice as fast as BiCG as we are replacing the matrix-vector products of A^T with more useful work. This method works quite well in most cases, however because the polynomials are squared rounding errors may become quite large. Van Der Vorst introduced the Biconjugate Gradient Stabilized method in 1992 [29], to remedy the irregular convergence associated with CGS. Instead of squaring the BiCG polynomial this method uses the BiCG polynomial with a different 'stabilizing' polynomial which helps to smooth the convergence.

2.4.3 Biconjugate Gradient Stabilized Method (BiCGStab)

Similar to CGS, we should be able to construct other iteration methods which generate x_{n+1} so that we can form a residual as a product of two polynomials

$$r_n = Q_n(A)P_n(A)r_0$$

In BiCGStab, we take $Q_n(A)$ in the form

$$Q_n(t) = (1-\omega_1 t)(1-\omega_2 t)\ldots(1-\omega_n t)$$

where we select ω_m such that in the mth step of iteration r_m is minimized with respect to ω_m for residuals that can be written as $r_n = P_n(A)Q_n(A)r_0$. So we define the recurrence relations in terms of the polynomials,

$$r_n = P_n(A)Q_n(A)r_0$$

$$p_n = T_n(A)Q_n(A)r_0$$

and derive the recurrences from step n to $n+1$ based on the BiCG recurrence relations. We then write the recurrences for r_{n+1} and p_{n+1} in vector form, generating the following algorithm.

Algorithm 7 *Biconjugate Gradient Stabilized*

```
Input $x_0$
Initialize $r_0 = p_0 = b - Ax_0$
Choose $\tilde{r}_0$ such that $\tilde{r}_0^T r_0 \neq 0$
$\rho_0 = \tilde{r}_0^T r_0$
    For $n = 0, 1, 2, ......$ until convergence
        $\rho_n = r_n^T \tilde{r}_0$
        $\sigma_n = \tilde{p}_n^T A p_0$
    $\alpha_n = \frac{\rho_n}{\sigma_n}$
    $s_n = r_n - \alpha_n A p_n$
    $w = A s_n$
    $\omega_n = \frac{(w, s_n)}{(w, w)}$
    $x_{n+1} = x_n + \alpha_n p_n + \omega_n s_n$
    $r_{n+1} = s_n - \omega_n A s_n$
    $\beta_n = \frac{\rho_{n+1}}{\rho_n}$
    $p_{n+1} = r_{n+1} + \beta_n (p_n + \omega_n A p_n)$
end for
```

Again, α_n and β_n are found using the BiCG coefficients and ensuring the following orthogonality conditions,

$$(p(A^T)\tilde{r}_0, P_n(A)r_0) = 0 \quad \text{and} \quad (p(A^T)\tilde{r}_0, AT_n(A)r_0) = 0 \tag{2.12}$$

where p is a polynomial of degree less than n and ω_n is found by minimizing the 2-norm of the residual vector $r_{n+1} = s_n - \omega_n A s_n$ so that

$$\omega_n = \frac{(As_n, s_n)}{(As_n, As_n)}.$$

We will also need an extra vector to express the recurrence for the residual vector namely, $s_n = r_n - \alpha_n A p_n$. In this method, we observe that the number of matrix vector operations is the same as in CGS and so expect it will be comparable in terms of convergence speed.

Although BiCGStab was developed to remedy the inherent problems associated with CGS by choosing a polynomial which would avoid large intermediate residuals, the computed ω may be zero or close to zero which may result in the breakdown or stagnation of this method [23]. To remedy the problem of stagnation associated with BiCGStab, Gutknecht [13] introduced BiCGStab2 which uses second degree polynomials to better handle this type of situation. Although improved, the methods still stagnates in some cases. Sleijpen and Fokkema [23] introduced a generalized method for this problem, namely BiCGStab(l) which forms an l^{th} degree polynomial after every l steps. Thus for $l = 1$ we have BiCGStab algorithm which is just a special case of the BiCGStab(l) method.

2.4.4 Mixed BiCGStab-CGS Method

This mixed method, developed more recently by Chan and Ye [6], uses both the standard CGS and BiCGStab methods and is derived in such a way that switching can occur between the two methods at each iteration. In a CGS based implementation, it aims at avoiding the increase in residual norm in CGS by switching to BiCGStab in order to improve overall stability.

The method constructs an approximation x_n such that its residual has the form

$$r_n = b - Ax_n = S_n(A)P_n(A)r_0,$$

where

$$S_n(t) = Q_k(t)P_{n-k}(t) \text{ and } Q_k(t) = (1 - \omega_1 t)\cdots(1 - \omega_k t)$$

and k is an integer parameter that determines what kind of residual reduction is used. So when constructing $r_{n+1} = S_{n+1}(A)P_{n+1}(A)r_0$ from r_n, a choice is made between taking either a BiCGStab step or a CGS step and one can choose either $S_{n+1}(t) = Q_k(t)P_{n+1-k}(t)$ or $S_{n+1}(t) = Q_{k+1}(t)P_{n-k}(t)$ where the former is called a CGS step and the latter a BiCGStab step. Thus, in the first n iterations, k steps of BiCGStab are taken and $n-k$ steps of CGS.

The derivation is similar to those of the mixed methods in Chapter 3, and we derive the recurrences for each case; $k(n+1) = k$ (a CGS step) and for $k(n+1) = k+1$ (a BiCGStab step) using the BiCG recurrence relations in equations (2.9) and (2.10). α_n and β_n are found in a way similar to those in sections 2.4.2 and 2.4.3 and again ω_n is found by minimizing the 2-norm of the residual vector where the residual is written as $r_{n+1} = v - \omega Av$. This gives us the following algorithm

Algorithm 8 *Mixed BiCGStab-CGS*

```
Input an initial approximation $x_0$ and an auxiliary vector $\tilde{r}_0$ ;
Initialize $r_0 = u_0 = v_0 = p_0 = b - Ax_0$; $k = 0$; $\rho_0 = \tilde{r}_0^T r_0$.
For $n = 0, 1, 2, \cdots$ until convergence
    Determine whether $k \leftarrow k$ (CGS step) or $k \leftarrow k+1$ (BiCGSTAB step);
    If (CGS), then
        $\alpha_n = \rho_n / \tilde{r}_0^T A p_n$
        $q_n = v_n - \alpha_n A p_n$
        $r_{n+1} = r_n - A(\alpha_n u_n + \alpha_{n-k} q_n)$
        $x_{n+1} = x_n + \alpha_n u_n + \alpha_{n-k} q_n$
```

```
            $\rho_{n+1} = \tilde{r}_0^T r_{n+1};$
            $\beta_{n+1} = \frac{\alpha_n \rho_{n+1}}{\alpha_{n-k}\rho_n} \quad (\text{or} = -\frac{\tilde{r}_0^T A q_n}{\tilde{r}_0^T A p_n})$
            $u_{n+1} = r_{n+1} + \beta_{n+1}(u_n - \alpha_{n-k} A p_n)$
            $v_{n+1} = r_{n+1} + \beta_{n+1-k} q_n$
            $p_{n+1} = u_{n+1} + \beta_{n+1-k}(q_n + \beta_{n+1} p_n)$
        End if
        If (BiCGSTAB), then
            $\alpha_n = \rho_n / \tilde{r}_0^T A u_n$
            $v = r_n - \alpha_n A u_n$
            $\omega = v^T A v / (Av)^T A v;$
            $r_{n+1} = v - \omega A v$
            $x_{n+1} = x_n + \alpha_n u_n + \omega v$
            $\rho_{n+1} = \tilde{r}_0^T r_{n+1};$
            $\beta_{n+1} = \frac{\alpha_n \rho_{n+1}}{\omega \rho_n} \quad (\text{or} = -\frac{\tilde{r}_0^T A v}{\tilde{r}_0^T A u_n})$
            $u_{n+1} = r_{n+1} + \beta_{n+1}(u_n - \omega A u_n)$
            $v_{n+1} = (I - \omega A)(v_n - \alpha_n A p_n)$
            $p_{n+1} = v_{n+1} + \beta_{n+1}(p_n - \omega A p_n)$
        End if
    End for
```

In this method we can see that several auxiliary vectors are needed to express the update of the residual vector. We also note that the number of matrix vector operations for each iteration if a BiCGStab is four, which is double that of the standard method. If a CGS step is taken however, we have two as in the standard method.

Since this method is a CGS based implementation and we only expect to take a BiCGStab step every so often, we expect this method to provide a competitive alternative for problems in which both CGS and BiCGStab diverge, or where the standard CGS is already competitive. It has also motivated further investigations in mixed methods with this thesis.

2.4.5 Shifted CGS method

Here we include a shifted CGS algorithm introduced by Fokkema et al. [11] to try and improve convergence properties of the CGS method, particularly to improve upon the irregular convergence. Below we outline the method as suggested in [11], although this method could also be implemented as a special case of the mixed BiCGStab-CGS method of Chan and Ye [6] of Section 2.4.4.

Suppose we consider, as in CGS and BiCGStab, forming the residual as a product of two polynomials, where in this case, we will use a 'shifted' polynomial as follows

$$r_n = b - Ax_n = \tilde{P}_n(A)P_n(A)r_0,$$

where

$$\tilde{P}_n(t) = (1 - \mu t)P_{n-1}(t)$$

Using the following BiCG recurrences,

$$P_{n+1}(t) = P_n(t) - \alpha_n t T_n(t),$$

$$T_n(t) = P_n(t) + \beta_n T_{n-1}(t),$$

(similarly defined for $\tilde{P}_{n+1}$ and $\tilde{T}_n$) we construct the recurrence for r_{n+1} and p_n from step n to $n+1$

$$\tilde{P}_{n+1}(A)P_{n+1}(A) \quad = \quad \tilde{P}_n(A)P_n(A) - \alpha_n A\tilde{P}_n(A)T_n(A) - \tilde{\alpha}_n A\tilde{T}_n(A)P_{n+1}(A)$$

and

$$\tilde{T}_n(A)T_n(A) \quad = \quad \tilde{T}_n(A)P_n(A) + \beta_n(\tilde{P}_n(A)T_{n-1}(A) + \tilde{\beta}_n\tilde{T}_{n-1}(A)T_{n-1}(A))$$

We can see that we will require two extra products to express each recurrence above. For r_{n+1} we need to formulate the following vectors,

$$v_n = \tilde{P}_n(A)T_n(A)r_0$$

and

$$s_n = \tilde{T}_n(A)P_{n+1}(A)r_0.$$

And for p_n we will need

$$t_n = \tilde{T}_n(A)P_n(A)r_0$$

and

$$u_n = \tilde{P}_n(A)T_{n-1}(A)r_0.$$

(where the polynomial $\tilde{T}_{n-1}(A)T_{n-1}(A)$ is found using p_{n-1}).

Omitting the derivations (see [11] for details), we can write the recurrences in vector form and obtain the following algorithm

Algorithm 9 *Shifted CGS*

```
Input $x_0$
Initialize $r_0 = p_0 = \tilde{r}_0 = t_0 = s_0 = u_0 = b - Ax_0$
$\rho_0 = \tilde{r}_0^T r_0$
$\beta_0 = 0, \ \ \alpha_0 = \mu$
    For $n = 1, 2, ......$ until convergence
        $\rho_n = r_n^T \tilde{r}_n$
        $\beta_n = \frac{\alpha_{n-1}}{\alpha_{n-2}} \frac{\rho_n}{\rho_{n-1}}$
        $v_n = r_n - \beta_n u_{n-1}$
        $t_n = r_n - \beta_{n-1} s_{n-1}$
        $p_n = t_n + \beta_n(u_n + \beta_{n-1} p_{n-1})$
        $c = Ap_n$
        $\sigma_n = (\tilde{p}_n, w_n) = \tilde{p}_n^T A p_n$
        $\alpha_n = \frac{\rho_n}{\sigma_n}$
    $s_n = t_n - \alpha_n c$
    $u_n = v_n - \alpha_{n-1} c$
    $r_{n+1} = r_n - A(\alpha_n v_n - \alpha_{n-1} s_n)$
    $x_{n+1} = x_n + \alpha_n v_n + \alpha_{n-1} s_n$
end for
```

where α_n and β_n are found in a way similar to those in sections 2.4.2 and 2.4.3.

In this implementation we also need to make the following modification for the first step ($n = 1$): $\tilde{\beta}_1 = \beta_0 = 0$ and $\tilde{\alpha}_1 = \alpha_0 = \mu$ so that the recurrence

$$r_n = b - Ax_n = \tilde{P}_n(A)P_n(A)r_0,$$

where

$$\tilde{P}_n(t) = (1 - \mu t)P_{n-1}(t)$$

holds. μ is chosen to be the inverse of an approximation of the largest eigenvalue of A.

It is easy to see that this implementation could be modified by choosing $\mu = \omega_1$, where ω_1 is the BiCGStab coefficient from minimizing the residual in the first step. This gives us a special case of the mixed BiCGStab-CGS of Chan and Ye [6] where the number of CGS and BiCGStab steps are fixed from the beginning with the first step taken as BiCGStab and all remaining steps are taken as CGS. We also note that we will need to perform two matrix vector operations in the shifted CGS method, which is the same as in the standard method.

2.4.6 Numerical testing of the shifted CGS algorithm

Although the shifted method has been tested in [11] it has only been tested for symmetric problems. Here we validate the improvements of this method for the nonsymmetric cases through our testing below. The method has been tested where the algorithm uses a shifted CGS polynomial where the residual is written as

$$r_n = b - Ax_n = \tilde{P}_n(A)P_n(A)r_0,$$

where

$$\tilde{P}_n(t) = (1 - \mu t)P_{n-1}(t)$$

as suggested in Section 2.4.5 where we take for the first step, $n = 1$ we take $\tilde{\beta}_1 = 0$ and $\tilde{\alpha}_1 = \mu$ where μ, is chosen to be the inverse of an approximation to the largest eigenvalue of A.[4] As mentioned, we could also substitute $\mu = \omega_1$ where ω_1 is the BiCGStab coefficient obtained from minimizing the 2-norm of the residual in the first step. This gives us a special case of the mixed BiCGStab-CGS of Chan and Ye [6] from the previous section. We would expect this implementation to give similar convergence histories[5] and it is somewhat less expensive than using μ as a the inverse to an approximation for the largest eigenvalue of A.

Next, we show a convergence history for one numerical example. This example will be used in subsequent numerical testing sections and is referred to as Example

[4]The largest magnitude eigenvalue was found using the *eigs* function in Matlab with a specified convergence tolerance of 1e1. This lack of accuracy did not seem to affect the convergence history of the observed tests and was far less expensive than finding a more accurate eigenvalue.

[5]as when computing only a rough approximation of the largest eigenvalue, as long as μ is within a certain range we would expect similar convergence histories if we only implement it for the first step.

1. Additional test problems are introduced in Section 3.2 and all methods have been tested with these. Convergence histories for the remaining examples for the shifted CGS method are given in Appendix A.

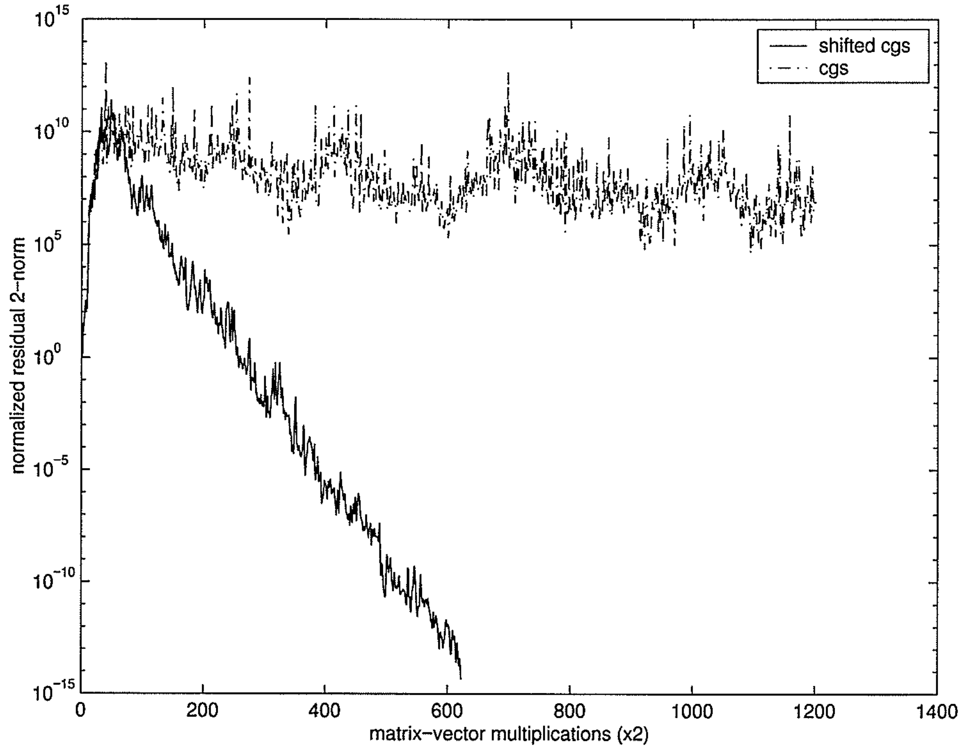
Example 1: The matrix is a finite-difference discretization on a 40×40 grids of the following convection diffusion equation

$$-\triangle u + \gamma(xu_x + yu_y) + \beta u = f(x,y) \quad \text{on} \quad (0,1)^2;$$

with the homogeneous Dirichlet boundary condition. f is a constant. This example was suggested by Freund [12] in his paper on a Transpose-Free Quasi-Minimal Residual algorithm and also used by Chan and Ye [6] to test their mixed BiCGStab-CGS method.

Figure 2.1 is the convergence history of the computed residuals for the set of parameters: $\beta = 100, \gamma = -100$ for Example 1. This is a good example where CGS stagnates. Here we see that shifted CGS for a nonsymmetric problem, not only improves the large residuals of CGS but also eventually converges in just over 600 matrix-vector operations.

Figure 2.1: **Convergence History for Example 1** $\beta = 100, \gamma = -100$



Chapter 3

Mixed-Product Methods

In this chapter, we will develop two mixed product methods. The first is based on CGS and switches to BiCG to remedy the problem of large residuals that may result CGS. The second method based on BiCGStab also switches to BiCG, in this case to remedy the problem of breakdown or stagnation that may result when the computed ω is zero or close to zero.

Since the difficulties encountered in BiCG, CGS and BiCGStab are of different type and usually occur only at a small subset of the iteration steps, it might be advantageous to consider a combination of them that can choose either of the two kinds of construction at each iteration and avoid using the one for which difficulties arise. In [6], a general concept of switching product Krylov subspace methods from one type to another through appropriately defining the sequence of polynomials has been introduced, and a mixed method that is based on the CGS and BiCGStab iterations has been derived. In this work, we consider mixed methods based on BiCG and BiCGStab as well as on BiCG and CGS, which is not unlike the generalized BiCG introduced in the previous chapter. We shall derive algorithms that allow switching between the two types of methods and use it to improve the stability of the algorithms.

3.1 Mixed BiCG-CGS method

In this formulation, we wish to construct an approximate solution x_n at step n, such that

$$r_n = P_k(A)P_n(A)r_0$$

and

$$\tilde{r}_n = P_{n-k}(A^T)r_0$$

where $k = k(n)$ is a parameter depending on n that determines whether a CGS or BiCG step is taken. Namely, we take k steps of CGS and $n - k$ steps of BiCG. Proceeding from step n to $n + 1$, we first choose $k(n + 1)$ as either $k(n) = k$ or $k(n) + 1 = k + 1$ (so either a BiCG step or CGS step respectively) and then construct

$$r_{n+1} = P_{k(n+1)}(A)P_{n+1}(A)r_0$$

using the BiCG recurrence relations,

$$P_{n+1}(t) = P_n(t) - \alpha_n t T_n(t),$$

$$T_{n+1}(t) = P_{n+1}(t) + \beta_{n+1}T_n(t).$$

In order to construct r_{n+1}, we define the following auxiliary polynomials and corresponding vectors (writing k = k(n))

$$\phi_n(t) = P_k(t)P_n(t), \quad \text{and} \quad r_n = \phi_n(A)r_0 = P_k(A)P_n(A)r_0; \tag{3.1}$$

$$\xi_n(t) = T_k(t)T_n(t), \quad \text{and} \quad p_n = \xi_n(A)r_0 = T_k(A)T_n(A)r_0; \tag{3.2}$$

$$\eta_n(t) = P_{n-k}(t), \quad \text{and} \quad \tilde{r}_n = \eta_n(A^T)r_0 = P_{n-k}(A^T)r_0; \tag{3.3}$$

$$\delta_n(t) = T_{n-k}(t), \quad \text{and} \quad \tilde{p}_n = \psi_n(A^T)r_0 = T_{n-k}(A^T)r_0; \tag{3.4}$$

$$\zeta_n(t) = T_k(t)P_n(t), \quad \text{and} \quad v_n = \zeta_n(A)r_0 = T_k(A)P_n(A)r_0; \tag{3.5}$$

$$\gamma_n(t) = P_k(t)T_n(t), \quad \text{and} \quad s_n = \gamma_n(A)r_0 = P_k(A)T_n(A)r_0 \tag{3.6}$$

Suppose the above polynomials have been obtained, we construct the corresponding polynomial for $n+1$. We first generate $\psi_n(t)$ by

$$\begin{aligned}\psi_n(t) &= T_k(t)P_{n+1}(t)\\ &= T_k(t)(P_n(t) - \alpha_n t T_n(t))\\ &= T_k(t)P_n(t) - \alpha_n t T_k(t)T_n(t))\\ &= \zeta_n(t) - \alpha_n t \xi_n(t)\end{aligned}$$

We proceed the construction by considering two cases.

Case 1: $k(n+1) = k(n) + 1$ (a CGS step).

$$\begin{aligned}
\phi_{n+1}(t) &= P_{k+1}(t)P_{n+1}(t) \\
&= (P_k(t) - \alpha_k t T_k(t))P_{n+1}(t) \\
&= P_k(t)P_{n+1}(t) - \alpha_n t T_k(t)P_{n+1}(t) \\
&= P_k(t)(P_n(t) - \alpha_n t T_n(t)) - \alpha_n t T_k(t)P_{n+1}(t) \\
&= \phi_n(t) - \alpha_n t\gamma_n(t) - \alpha_k t\psi_n(t), \\
\eta_{n+1}(t) &= P_{n+1-k-1} = \eta_n, \\
\delta_{n+1}(t) &= T_{n+1-(k+1)}(t) = \delta_n(t), \\
\zeta_{n+1}(t) &= T_{k+1}(t)P_{n+1}(t) \\
&= (P_{k+1}(t) + \beta_{k+1}T_k(t))P_{n+1}(t) \\
&= P_{k+1}(t)P_{n+1}(t) + \beta_{k+1}T_k(t)P_{n+1}(t) \\
&= \phi_{n+1}(t) + \beta_{k+1}\psi_n(t), \\
\gamma_{n+1}(t) &= P_{k+1}(t)T_{n+1}(t) \\
&= P_{k+1}(P_{n+1}(t) + \beta_{n+1}T_n(t)) \\
&= P_{k+1}(t)P_{n+1}(t) + \beta_{n+1}(P_k(t) - \alpha_k t T_k(t))T_n(t) \\
&= \phi_{n+1} + \beta_{n+1}(\gamma_n(t) - \alpha_k t\xi_n(t)), \\
\xi_{n+1}(t) &= T_{k+1}(t)T_{n+1}(t) \\
&= (P_{k+1}(t) + \beta_{k+1}T_k(t))T_{n+1}(t) \\
&= P_{k+1}(t)T_{n+1}(t) + \beta_{k+1}T_k(t)T_{n+1}(t) \\
&= \gamma_{n+1}(t) + \beta_{k+1}(\psi_n(t) + \beta_{n+1}\xi_n(t)).
\end{aligned}$$

Case 2. $k(n+1) = k(n)$ (a BICG step).

$$\begin{aligned}
\phi_{n+1} &= P_k(t)P_{n+1}(t) \\
&= P_k(t)(P_n(t) - \alpha_n t T_n(t)) \\
&= \phi_n(t) - \alpha_n t \gamma_n(t), \\
\eta_{n+1} &= P_{n+1-k} = P_{n-k}(t) - \alpha_{n-k} t T_{n-k}(t) \\
&= \eta_n - \alpha_{n-k} t \delta_n(t), \\
\delta_{n+1} &= T_{n+1-k} = P_{n+1-k}(t) + \beta_{n+1-k} T_{n-k}(t) \\
&= \eta_{n+1} + \beta_{n+1-k} \delta_n(t), \\
\zeta_{n+1} &= T_k(t)P_{n+1}(t) = \psi_n(t), \\
\gamma_{n+1} &= P_k(t)T_{n+1}(t) \\
&= P_k(t)P_{n+1}(t) + \beta_{n+1} P_k(t)T_n(t) \\
&= \phi_{n+1}(t) + \beta_{n+1}\gamma_n(t), \\
\xi_{n+1} &= T_k(t)T_{n+1}(t) \\
&= T_k(t)P_{n+1}(t) + \beta_{n+1} T_k(t)T_n(t) \\
&= \zeta_{n+1}(t) + \beta_{n+1}\xi_n(t).
\end{aligned}$$

In vector form we first generate $q_n = \psi_n(A)r_0 = v_n - \alpha_n A p_n$, and the remaining vector recurrences can be expressed as follows.

Case 1: $k(n+1) = k(n) + 1$ (a CGS step)

$$r_{n+1} = r_n - \alpha_n A s_n - \alpha_k A q_n = r_n - A(\alpha_n s_n - \alpha_k q_n),$$

$$\begin{aligned}
\tilde{r}_{n+1} &= \tilde{r}_n, \\
\tilde{p}_{n+1} &= \tilde{p}_n, \\
s_{n+1} &= r_{n+1} + \beta_{n+1}(s_n - \alpha_k A p_n), \\
v_{n+1} &= r_{n+1} + \beta_{k+1} q_n, \\
p_{n+1} &= s_{n+1} + \beta_{k+1}(q_n + \beta_{n+1} p_n).
\end{aligned}$$

Case 2. $k(n+1) = k(n)$ (a BICG step)

$$\begin{aligned}
r_{n+1} &= r_n - \alpha_n A s_n, \\
\tilde{r}_{n+1} &= \tilde{r}_n - \alpha_{n-k} A^T \tilde{p}_n, \\
\tilde{p}_{n+1} &= \tilde{r}_{n+1} + \beta_{n-k+1} \tilde{p}_n, \\
s_{n+1} &= r_{n+1} + \beta_{n+1} s_n, \\
v_{n+1} &= q_n, \\
p_{n+1} &= v_{n+1} + \beta_{n+1} p_n.
\end{aligned}$$

We next recover the BiCG coefficients $\alpha_n = \rho_n/\sigma_n$ and $\beta_{n+1} = \rho_{n+1}/\rho_n$. Note that α_{n-k+1}, β_{n-k+1} etc. have been computed in the earlier steps and can be stored.

From BiCG we have $\rho_n = (P_n(A^T)\tilde{r}_0, P_n(A)r_0)$. By construction,$P_n(A)r_0$ is orthogonal to all vectors $p(A^T)\tilde{r}_0$, where p is an arbitrary polynomial of degree less than n. Thus, we only need consider the highest order term of $P_n(A^T)$ when computing ρ_n, namely, $P_n(A^T) = (-1)^n \alpha_0, \alpha_1, \cdots \alpha_{n-1}(A^T)^n +$ (lower degree terms). Using the biconjugacy property, we have

$$\begin{aligned}\rho_n &= (P_n(A^T)\tilde{r}_0, P_n(A)r_0) \\ &= (-1)^n\alpha_0, \alpha_1 \ldots \alpha_{n-1}((A^T)^n\tilde{r}_0, P_n(A)r_0)\end{aligned}$$

Define

$$\tilde{\rho}_n = (\tilde{r}_n, r_n)$$

and we compute,

$$\begin{aligned}\tilde{\rho}_n &= (P_{n-k}(A^T)\tilde{r}_0, P_k(A)P_n(A)r_0) \\ &= (P_k(A^T)P_{n-k}(A^T)\tilde{r}_0, P_n(A)r_0)\end{aligned}$$

where the highest order term is given by $P_{n-k}(A^T)P_k(A^T) = (-1)^n\alpha_0, \ldots \alpha_{n-k-1} \cdot \alpha_0, \ldots \alpha_{k-1}(A^T)^n +$ (lower degree terms). We obtain,

$$\tilde{\rho}_n = (-1)^n\alpha_0, \ldots, \alpha_{n-k-1} \cdot \alpha_0, \ldots, \alpha_{k-1}((A^T)^n\tilde{r}_0, P_n(A)r_0).$$

Thus

$$\rho_n = \frac{\alpha_k \cdots \alpha_{n-1}}{\alpha_0 \cdots \alpha_{n-k-1}}\tilde{\rho}_n.$$

In a similar way, we can find σ_n,

$$\sigma_n = (T_n(A^T)\tilde{r}_0, AT_n(A)r_0)$$

where, using the biconjugacy property, the highest order term is,

$$T_n(A^T) = (-1)^n \alpha_0 \ldots \alpha_{n-1} (A^T)^n + \ldots$$

and we have,

$$\sigma_n = (-1)^n \alpha_0 \ldots \alpha_{n-1} ((A^T)^n \tilde{r}_0, AT_n(A) r_0)$$

where $\tilde{\sigma}_n$, as defined by $\tilde{\sigma}_n = (\tilde{p}_n, Ap_n)$, is found in the same manner, using the highest order term, we have

$$\begin{aligned} \tilde{\sigma}_n &= (\tilde{p}_n, Ap_n) = (T_{n-k}(A^T)\tilde{r}_0, AT_k(A)T_n(A)r_0) \\ &= (T_k(A^T)T_{n-k}(A^T)\tilde{r}_0, AT_n(A)r_0) \\ &= (-1)^n \alpha_0 \ldots \alpha_{k-1} \cdot \alpha_0 \ldots \alpha_{n-k-1} ((A^T)^n \tilde{r}_0, AT_n(A) r_0) \end{aligned}$$

Thus,

$$\sigma_n = \frac{\alpha_{n-1} \cdots \alpha_k}{\alpha_{n-k-1} \cdots \alpha_0} \tilde{\sigma}_n.$$

So

$$\alpha_n = \rho_n / \sigma_n = \tilde{\rho}_n / \tilde{\sigma}_n$$

and

$$\beta_{n+1} = \frac{\rho_{n+1}}{\rho_n} = \frac{\alpha_{k(n+1)} \cdots \alpha_n}{\alpha_0 \ldots \alpha_{n+1-k(n+1)-1}} \frac{\alpha_0 \cdots \alpha_{n-k(n)-1}}{\alpha_{k(n)} \cdots \alpha_{n-1}} \frac{\tilde{\rho}_{n+1}}{\tilde{\rho}_n}.$$

$$= \begin{cases} \dfrac{\alpha_n \tilde{\rho}_{n+1}}{\alpha_k \tilde{\rho}_n}, & \text{if } k(n+1) = k(n) + 1, \text{ i.e. a CGS step,} \\ \dfrac{\alpha_n \tilde{\rho}_{n+1}}{\alpha_{n-k} \tilde{\rho}_n}, & \text{if } k(n+1) = k(n), \text{ i.e., a BiCG step} \end{cases}$$

Summarising the above derivation and writing in the vector recurrence form, we have the following algorithm:

Algorithm 10 *BiCG-CGS*

```
Input $x_0$;
Initialize $r_0 = p_0 = v_0 = s_0 = b - Ax_0$;  $\tilde{r}_0 = \tilde{p}_0$ and $\rho_0 = \tilde{r}_0^T r_0$; $k = k_{new} = 0$.
  For $n = 0, 1, 2, ......$ until convergence
      $k = k_{new}$; Determine $k_{new} = k$ or $k + 1$;
      $w_n = Ap_n$
      $\sigma_n = (\tilde{p}_n, w_n) = \tilde{p}_n^T A p_n$
      $\alpha_n = \frac{\rho_n}{\sigma_n}$
      $q_n = v_n - \alpha_n w_n$
      If $k_{new} = k + 1$ (CGS step), then
          $r_{n+1} = r_n - A(\alpha_n s_n + \alpha_k q_n)$
          $x_{n+1} = x_n + \alpha_n s_n + \alpha_k q_n$
          $\tilde{r}_{n+1} = \tilde{r}_n$
          $\tilde{p}_{n+1} = \tilde{p}_n$
          $\rho_{n+1} = \tilde{r}_{n+1}^T r_{n+1}$
          $\beta_{n+1} = \frac{\alpha_n}{\alpha_k} \frac{\rho_{n+1}}{\rho_n}$
          $s_{n+1} = r_{n+1} + \beta_{n+1}(s_n - \alpha_k w_n)$
```

```
            $v_{n+1} = r_{n+1} + \beta_{k+1} q_n$
            $p_{n+1} = s_{n+1} + \beta_{k+1}(q_n + \beta_{n+1} p_n)$
        End if
        If $k_{new} = k$ (BiCG step), then
            $r_{n+1} = r_n - \alpha_n A s_n$
            $x_{n+1} = x_n + \alpha_n s_n$
            $\tilde{r}_{n+1} = \tilde{r}_n - \alpha_{n-k} A^T \tilde{p}_n$
            $\tilde{p}_{n+1} = \tilde{r}_{n+1} + \beta_{n+1-k} \tilde{p}_n$
            $\rho_{n+1} = \tilde{r}_{n+1}^T r_{n+1}$
            $\beta_{n+1} = \frac{\alpha_n}{\alpha_{n-k}} \frac{\rho_{n+1}}{\rho_n}$
            $s_{n+1} = r_{n+1} + \beta_{n+1} s_n$
            $v_{n+1} = q_n$
            $p_{n+1} = v_{n+1} + \beta_{n+1} p_n$
        End if
    end for
```

In the CGS step above, the algorithm implements

$$r_{n+1} = r_n - A(\alpha_n s_n + \alpha_k q_n)$$

correspondingly,

$$x_{n+1} = x_n - \alpha_n s_n + \alpha_k q_n.$$

In this algorithm we observe that there are two matrix vector operations carried out for each iteration regardless of which method is used. For each CGS step, two matrix vector operations with the matrix A are performed and for each BiCG step there is one matrix vector operation with A and another with the transpose of A. In general, one would expect this algorithm to converge at least as fast as the standard BiCG method since we are still performing one matrix vector operation with the transpose of A, although in this case, we will have only a few iterations where this operation will be carried out.

3.2 Numerical testing for mixed BiCG-CGS

In the Numerical testing sections that follow, we present numerical examples to demonstrate improvements made by the new methods. We also include testing of Example 1 which has been introduced in Section 2.4.6. All testing that follows, was carried out using Matlab with double precision on a SUN workstation. Throughout our Numerical Testing, the following Examples (2 through 4) as well as Example 1, are used and will be referred to throughout the remainder of the Chapter. We have attempted to use these examples with each method both as a means of comparison between the various methods herewithin and as comparison with various methods in the literature, as these examples have been extensively used. Where a certain Example has not been shown with a particular method, it is because it has provided no additional information to us and as such is an uninteresting case. In almost all cases, we have used a standard stopping criterion [26] with the iteration terminating when

$$||r_{n+1}||/||r_n|| \leq 10^{-15}$$

In certain examples round-off error created instabilities at this level of accuracy and so we have used a termination threshold of 10^{-10} in those cases.

Example 2: The matrix is a finite-difference discretization (centre difference) on a 40×40 grid of the following convection diffusion equation

$$-\triangle u + \beta u_x + \gamma u_y = f(x,y) \quad \text{on} \quad (0,1)^2;$$

with the homogeneous Dirichlet boundary condition. f is constant.

Example 3: The matrix is Sherman1 from the Sherman set in the Harwell-Boeing collection [9] of sparse test matrices. This set includes five nonsymmetric test matrices which result from oil resevoir modeling. Sherman1 in particular is found from a black oil simulation with shale barriers. The order of the matrix is 1000 by 1000 and it has 3750 nonzero entries. The right-hand side b is chosen to be the vector of ones.

Example 4: The matrix is a finite-difference discretization on a 40×40 grids of the following convection diffusion equation

$$-\triangle u + \gamma(a(x,y)u_x + b(x,y)u_y) = 0 \quad \text{on} \quad (0,1)^2;$$

where $a(x,y) = x(x-1)(1-2y)$, $b(x,y) = y(1-y)(1-2x)$, with the homogeneous Dirichlet boundary condition. This example was used by Sleijpen and Fokkema [23] to test their BiCGStab(l) method.

Throughout all test examples that follow for the BiCG-CGS method, the Tolerance is chosen to be 10^2 where the criteria for a switch from CGS to BiCG is an increase of the local residual namely,

$$||r_{n+1}||/||r_0|| \leq Tol. \tag{3.7}$$

Figure 3.1 shows the convergence history for Example 1 for the set of parameters: $\beta = 100$, $\gamma = -100$. Here we see that the mixed method converges more smoothly and in fewer steps than the standard CGS method with two switches occurring in this run.

Figure 3.1: **Convergence History for Example 1** $\beta = -110$,$\gamma = 110$

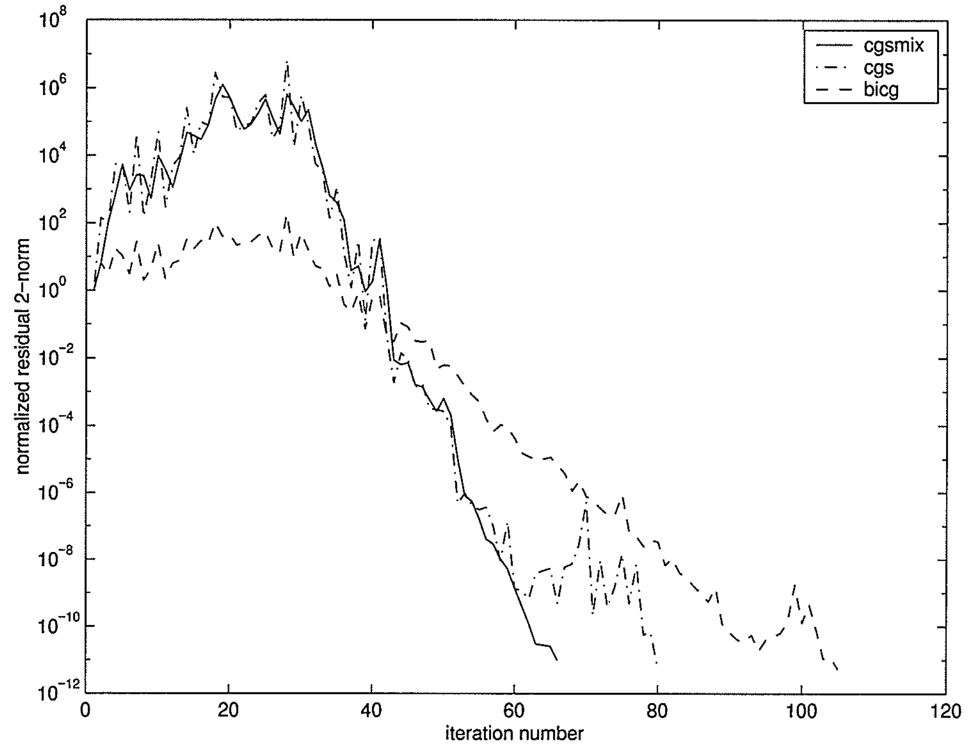
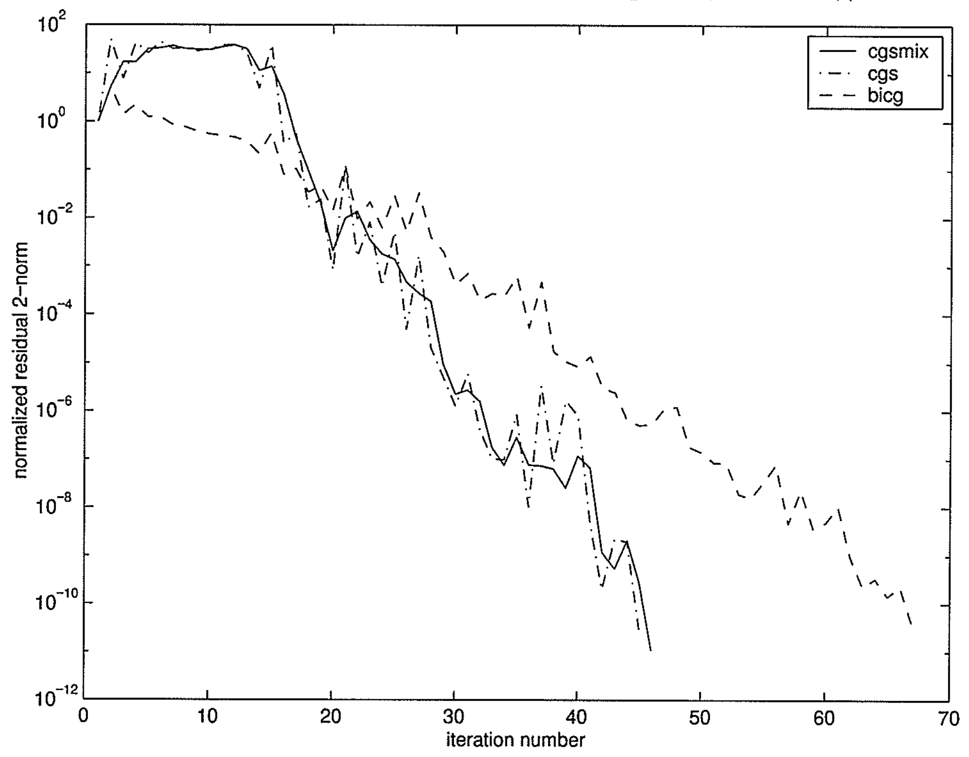


Figure 3.2 gives the convergence history of the computed residuals for Example 2 with the set of parameters: $\beta = -100$, $\gamma = 100$. We can see although the mixed method converges in about the same number of iterations, it is an improvement over the standard CGS method in that it smoothes the large residuals with only one switch between methods.

Figure 3.2: **Convergence History for Example 2** $\beta = -100, \gamma = 100$



Both Examples problems 3 and 4 have been tested for this method, but were not found to have any benefits over the standard CGS or BiCG methods and so have not been included here.

The mixed BiCG-CGS initially experienced instabilities due to the formulation of the coefficient β. This sensitivity is common in many iterative methods depending on such parameters and so it is an easy candidate to look at when numerical instabilities arise [21]. Formulations for the coefficient β have proved to give extremely varying results when tests were run with these code in Matlab. Since the coefficients are derived directly from computing the inner products, $\sigma_n = (T_n(A^T)\tilde{r}_0, AT_n(A)r_0)$, $\rho_n = (P_n(A^T)\tilde{r}_0, P_n(A)r_0)$, any computational error in either inner product will thus lead to similar relative perturbation error in both coefficients.[1] In the mixed BiCG-CGS β has been coded as

$$\beta_{n+1} = \rho_{n+1}/\rho_n \tag{3.8}$$

[2] although another formulation is found from

$$\beta_{n+1} = \frac{\theta_{n+1}}{\theta_n}\frac{\rho_{n+1}}{\sigma_n}$$

as in [11] where, for BiCG, θ_{n+1} and θ_n are the nontrivial leading coefficients of the polynomials $P_{n+1}(A)$ and $P_n(A)$ respectively. The leading coefficient of $P_{n+1}(A)$ is $(-1)^n\alpha_n\cdots\alpha_0$ we therefore obtain

$$\frac{\theta_{n+1}}{\theta_n} = \frac{-1}{\alpha_n}$$

and

$$\beta_{n+1} = \frac{1}{\alpha_n}\frac{\rho_{n+1}}{\sigma_n}$$

where the negative has been incorporated into the BiCG recurrence,

$$T_{n+1}(t) = P_{n+1}(t) + \beta_{n+1}T_n(t),$$

This formulation for β, was found to be numerically unstable when implemented and tested in Matlab. For all methods that have been tested, this formulation made the

[1]see [26] and [25] for more discussion concerning error bounds when computing inner products and also choosing suitable polynomials for computing σ which do not degrade the convergence.

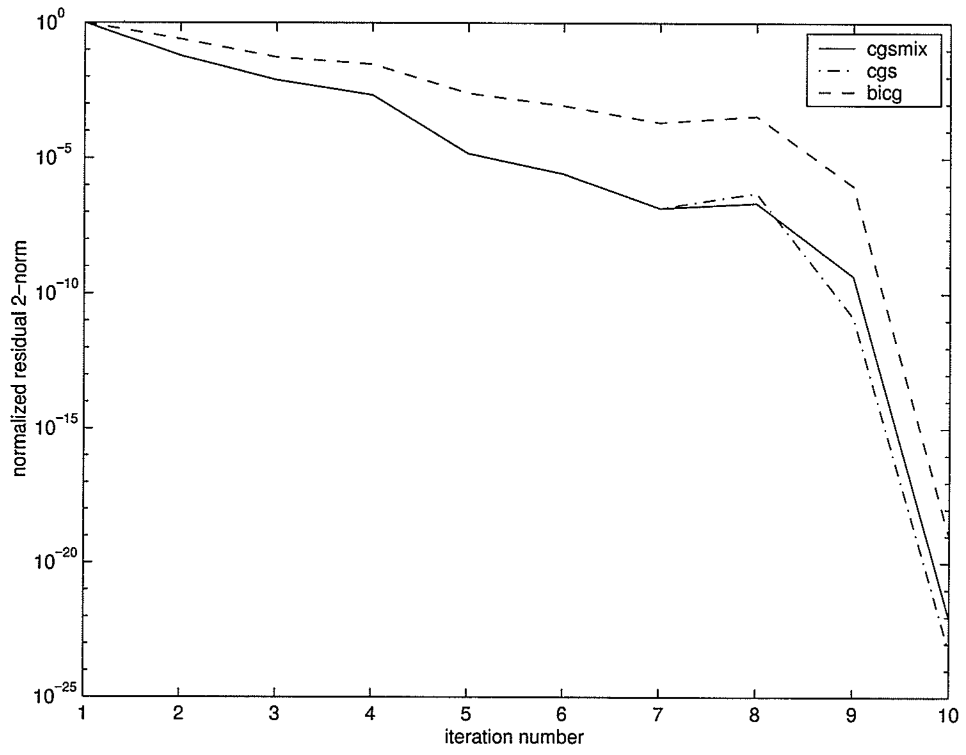
[2]this formulation for β is also used for both the BiCG-BiCGStab mixed method as well as the shifted CGS method, tested in the sections that follow.

method diverge. Since the formulation of coefficients can often give unstable results for a given method, we tested the formulation given in (3.8) immediately after and obtained improved convergence. Thus we have used this formulation for all methods that follow.

The second potential problem which we have encountered is instabilities due to the derivation of the recurrence formulations in the algorithm. In deriving the new formulations it is important to keep in mind that at most we would like two matrix-vector multiplications per iteration to keep the method competitive. Although we would like to try and beat the convergence in both the BiCG and CGS and if not, at the least we would like to be able to improve the convergence in certain well-defined cases. In general, we would expect that by using BiCG we would have a more stable method, as BiCG seems to be inherently more stable than CGS. However, we have not found this to be the case in the two formulations which we have tried. Although the recurrence which we have used from Section 3.1 is quite simple, we have found that for a tolerance of $1e2$, certain examples were unstable. Although they may have been stable for other tolerance levels, it is still necessary to do tests to find a stable tolerance and so we do not include these here.

Although we may expect some instabilities from the mixed methods, we hope that regardless of input tolerance we would at least get convergence from the mixed method when both the standard CGS and BiCG methods are converging. Although this stability was not observed all the time, there were several examples where, for an input tolerance of $1e2$, the method gave improved convergence. We initially experienced instabilities for all examples and tried to remedy this problem first by making certain that the coefficient formulations (for β in particular) were not causing any numerical instabilities and then next, by deriving a second formulation for the method. The formulation of Algorithm 6 was found to be the most stable among several possible formulations tested and we show this through our tests of the finite termination properties on a few 10×10 matrices. Here we include one such test (seen in Figure 3.4) to demonstrate that our code is correct. When implemented, we forced a switch to occur in each test to make sure that the mixing of the derived methods worked together.

Figure 3.3: **Finite Termination Test for a 10×10 Matrix**



3.3 Mixed BiCG-BiCGStab method

In this formulation, we wish to construct an approximation x_n at step n, such that

$$r_n = Q_k(A)P_n(A)r_0$$

and

$$\tilde{r}_n = P_{n-k}(A^T)r_0$$

where $k = k(n)$ is a parameter depending on n that determines whether a BiCGStab or BiCG step is taken. Namely, we take k steps of BiCGStab and $n-k$ steps of BiCG. Proceeding from step n to $n+1$, we first choose $k(n+1)$ as either $k(n)$ or $k(n)+1$ (so either a BiCG step or BiCGStab step respectively) and then construct r_{n+1}.

Using the BiCG recurrence relations,

$$P_{n+1}(t) = P_n(t) - \alpha_n t T_n(t),$$

$$T_{n+1}(t) = P_{n+1}(t) + \beta_{n+1} T_n(t),$$

In order to construct r_{n+1}, we define the following auxiliary polynomials and corresponding vectors

$$\phi_n(t) = Q_k(t)P_n(t), \quad \text{and} \quad r_n = \phi_n(A)r_0 = Q_k(A)P_n(A)r_0; \tag{3.9}$$

$$\xi_n(t) = Q_k(t)T_n(t), \quad \text{and} \quad p_n = \xi_n(A)r_0 = Q_k(A)T_n(A)r_0; \tag{3.10}$$

$$\eta_n(t) = P_{n-k}(t), \quad \text{and} \quad \tilde{r}_n = \eta_n(A)r_0 = P_{n-k}(A)r_0; \tag{3.11}$$

$$\psi_n(t) = T_{n-k}(t), \quad \text{and} \quad \tilde{p}_n = \psi_n(A)r_0 = T_{n-k}(A)r_0; \tag{3.12}$$

Suppose the above polynomials have been obtained, we construct the corresponding polynomial for $n+1$.

Case 1: $k(n+1) = k(n) + 1$ (a BiCGStab step).

$$\begin{aligned}
\phi_{n+1} &= Q_{k+1}(t)P_{n+1}(t) \\
&= (1 - \omega_{k+1}t)Q_k(t)(P_n(t) - \alpha_n t T_n(t)) \\
&= (1 - \omega_{k+1}t)(\phi_n(t) - \alpha_n t \xi_n(t)) \\
\xi_{n+1} &= Q_{k+1}(t)T_{n+1}(t) \\
&= Q_{k+1}(t)P_{n+1}(t) + \beta_{n+1}Q_{k+1}(t)T_n(t) \\
&= \phi_{n+1}(t) + \beta_{n+1}(1 - \omega_{k+1}t)\xi_n(t) \\
\eta_{n+1} &= P_{n+1-k-1} = \eta_n \\
\psi_{n+1} &= T_{n+1-(k+1)}(t) = \psi_n(t)
\end{aligned}$$

Case 2. $k(n+1) = k(n)$ (a BICG step).

$$\begin{aligned}
\phi_{n+1} &= Q_k(t)P_{n+1}(t) \\
&= Q_k(t)(P_n(t) - \alpha_n t T_n(t)) \\
&= \phi_n(t) - \alpha_n t \xi_n(t) \\
\xi_{n+1} &= Q_k(t)T_{n+1}(t) \\
&= Q_k(t)P_{n+1}(t) + \beta_{n+1}Q_k(t)T_n(t) \\
&= \phi_{n+1}(t) + \beta_{n+1}\xi_n(t) \\
\eta_{n+1} &= P_{n+1-k} = P_{n-k}(t) - \alpha_{n-k}tT_{n-k}(t) \\
&= \eta_n - \alpha_{n-k}t\psi_n(t)
\end{aligned}$$

$$\begin{aligned}\psi_{n+1} &= T_{n+1-k} = P_{n+1-k}(t) + \beta_{n+1-k} T_{n-k}(t) \\ &= \eta_{n+1} + \beta_{n+1-k} \psi_n(t)\end{aligned}$$

To simplify the equations, we first define v_n as $v_n = r_n - \alpha_n A p_n$. and we express the recurrences in vector form as follows.

Case 1: $k(n+1) = k(n) + 1$ (a BiCGStab step)

$$\begin{aligned}r_{n+1} &= v_n - \omega_{k+1} A v_n, \\ \tilde{r}_{n+1} &= \tilde{r}_n, \\ \tilde{p}_{n+1} &= \tilde{p}_n, \\ p_{n+1} &= r_{n+1} + \beta_{n+1}(1 + \omega_{k+1} A) p_n.\end{aligned}$$

Case 2. $k(n+1) = k(n)$ (a BICG step)

$$\begin{aligned}r_{n+1} &= r_n - \alpha_n A p_n, \\ \tilde{r}_{n+1} &= \tilde{r}_n - \alpha_{n-k} A^T \tilde{p}_n, \\ \tilde{p}_{n+1} &= \tilde{r}_{n+1} + \beta_{n+1-k} \tilde{p}_n, \\ p_{n+1} &= r_{n+1} + \beta_{n+1} p_n.\end{aligned}$$

We next recover the BiCG coefficients $\alpha_n = \rho_n / \sigma_n$ and $\beta_{n+1} = \rho_{n+1} / \rho_n$. Note that α_{n-k+1}, β_{n-k+1} etc. have been computed in the earlier steps and can be stored.

Noting that $T_n(t) = (-1)^n \alpha_{n-1} \cdots \alpha_0 t^n + \cdots$ and using the bi-conjugacy property, we have

$$\sigma_n = (T_n(A^T)\tilde{r}_0, AT_n(A)r_0) = (-1)^n \alpha_{n-1} \cdots \alpha_0((A^T)^n\tilde{r}_0, AT_n(A)r_0)$$

Similarly, from $T_{n-k}(t)Q_k(t) = (-1)^n \alpha_{n-k-1} \cdots \alpha_0 \omega_k \cdots \omega_1 t^n + \cdots$, it follows

$$\begin{aligned}
\tilde{\sigma}_n &= (\tilde{p}_n, Ap_n) \\
&= (T_{n-k}(A^T)\tilde{r}_0, AQ_k(A)T_n(A)r_0) \\
&= (Q_k(A^T)T_{n-k}(A^T)\tilde{r}_0, AT_n(A)r_0) \\
&= (-1)^n \alpha_{n-k-1} \cdots \alpha_0 \omega_k \cdots \omega_1((A^T)^n\tilde{r}_0, AT_n(A)r_0)
\end{aligned}$$

Thus,

$$\sigma_n = \frac{\alpha_{n-1} \cdots \alpha_{n-k}}{\omega_k \cdots \omega_1}\tilde{\sigma}_n.$$

Also $\rho_n = (P_n(A^T)\tilde{r}_0, P_n(A)r_0) = (-1)^n \alpha_{n-1} \cdots \alpha_0((A^T)^n\tilde{r}_0, P_n(A)r_0)$ and

$$\begin{aligned}
\tilde{\rho}_n &= (\tilde{r}_n, r_n) = (P_{n-k}(A^T)\tilde{r}_0, Q_k(A)P_n(A)r_0) \\
&= (Q_k(A^T)P_{n-k}(A^T)\tilde{r}_0, P_n(A)r_0) \\
&= (-1)^n \alpha_{n-k-1} \cdots \alpha_0 \omega_k \cdots \omega_1((A^T)^n\tilde{r}_0, P_n(A)r_0).
\end{aligned}$$

Thus

$$\rho_n = \frac{\alpha_{n-1} \cdots \alpha_{n-k}}{\omega_k \cdots \omega_1}\tilde{\rho}_n.$$

So

$$\alpha_n = \rho_n/\sigma_n = \tilde{\rho}_n/\tilde{\sigma}_n$$

and

$$\begin{aligned}
\beta_{n+1} &= \frac{\rho_{n+1}}{\rho_n} = \frac{\alpha_n \cdots \alpha_{n+1-k(n+1)}}{\omega_{k(n+1)} \cdots \omega_1} \frac{\omega_{k(n)} \cdots \omega_1}{\alpha_{n-1} \cdots \alpha_{n-k(n)}} \frac{\tilde{\rho}_{n+1}}{\tilde{\rho}_n}. \\
&= \left\{ \begin{array}{ll} \dfrac{\alpha_n \tilde{\rho}_{n+1}}{\alpha_{n-k} \tilde{\rho}_n}, & \text{if } k(n+1) = k(n), \text{ i.e. a BiCG step,} \\ \dfrac{\alpha_n \tilde{\rho}_{n+1}}{\omega_{k+1} \tilde{\rho}_n}, & \text{if } k(n+1) = k(n)+1, \text{ i.e., a BiCGStab step} \end{array} \right.
\end{aligned}$$

where $k = k(n)$.

Summarising the above derivation and writing in the vector recurrence form, we have the following algorithm:

Algorithm 11 *BiCG-BiCGStab*

```
Input $x_0$;
Initialize $r_0 = p_0 = b - Ax_0$; $\tilde{r}_0 = \tilde{p}_0$ and $\rho_0 = \tilde{r}_0^T r_0$; $k = k_{new} = 0$.
    For $n = 0, 1, 2, ......$ until convergence
        $k = k_{new}$; Determine $k_{new} = k$ or $k + 1$;
        $w = Ap_n$
        $\sigma_n = (\tilde{p}_n, w) = \tilde{p}_n^T Ap_n$
        $\alpha_n = \frac{\rho_n}{\sigma_n}$
        If $k_{new} = k + 1$ (BiCGStab step), then
            $v = r_n - \alpha_n w$
            $r = Av$
        $\omega_{k_{new}} = (r, v)/(r, r);$
        $r_{n+1} = r = v - \omega_{k_{new}} r$
        $x_{n+1} = x_n + \alpha_n p_n + \omega_{k_{new}} v$
        $\tilde{r}_{n+1} = \tilde{r}_n$
        $\rho_{n+1} = \tilde{r}_{n+1}^T r_{n+1};$
        $\beta_{n+1} = \frac{\alpha_n \tilde{\rho}_{n+1}}{\omega_{k_{new}} \tilde{\rho}_n}$
        $p_{n+1} = r_{n+1} + \beta_{n+1}(p_n - \omega_{k_{new}} w)$
        $\tilde{p}_{n+1} = \tilde{p}_n$
    End if
    If $k_{new} = k$ (BiCG step), then
        $r_{n+1} = r_n - \alpha_n w$
        $x_{n+1} = x_n + \alpha_n p_n$
        $\tilde{r}_{n+1} = \tilde{r}_n - \alpha_{n-k} A^T \tilde{p}_n$
        $\rho_{n+1} = \tilde{r}_{n+1}^T r_{n+1};$
        $\beta_{n+1} = \frac{\alpha_n \tilde{\rho}_{n+1}}{\alpha_{n-k} \tilde{\rho}_n}$
        $p_{n+1} = r_{n+1} + \beta_{n+1} p_n$
        $\tilde{p}_{n+1} = \tilde{r}_{n+1} + \beta_{n-k+1} \tilde{p}_n$
    End if
end for
```

In the BiCGSTAB step above, the algorithm implements

$$r_{n+1} = r_n - \alpha_n A p_n - \omega_{k_{new}} A v_n.$$

correspondingly,

$$x_{n+1} = x_n - \alpha_n p_n - \omega_{k_{new}} v_n.$$

In this algorithm we observe that there are two matrix vector operations carried out for each iteration regardless of which method is used as in the mixed BiCG-CGS implementation. For each BiCGStab step, two matrix vector operations with the matrix A are performed and for each BiCG step there is one matrix vector operation with A and another with the transpose of A. As in the mixed BiCG-CGS, we would expect this algorithm to converge at least as fast as the standard BiCG method since we are still performing one matrix vector operation with the transpose of A, although in this case, we will have only a few iterations where this more expensive operation will be carried out.

3.4 Numerical testing of mixed BiCG-BiCGStab

In this mixed method we have tried to remedy the problem inherent in BiCGStab, namely that it is prone to stagnation in certain types of problems. Example problem 1 was a particularily good candidate, as for many different input parameters we find that there is stagnation in the standard BiCGStab method. Often the stagnation occurs mid-way through convergence, but often it occurs near the beginning and in almost all cases, the new mixed BiCG-BiCGStab has fixed this stagnation and converged. We have also hoped that our new method might beat both BiCGStab and BiCG in cases where both converged as the mixed method of Chan and Ye [6] and although it did not happen in all cases, we have included a couple problems where this is the case.

3.4.1 Switching Criterion

Beginning with a BiCGStab step, we switch to a BiCG step when some switching threshold is reached. There have been no theoretical results up until now to show us exactly when BiCGStab fails and thus when we should switch to BiCG. Only with numerical testing can we better understand these difficulties and hence the switching criterion with the mixed methods. For this method, we have tried to switch to BiCG when we find ω to be below a certain prescribed tolerance. We have tried several other implementations, these include switching when β has exceeded this tolerance and also when some combination of ω with the s and v vectors has been reached. In all these other attempts, the mixed method has a very large range of activity. By range we mean that in changing the tolerance level (where small tol $\approx 1e-7$) it follows BiCGStab[3] and a large tolerance ($\approx 1e1$) it follows BiCG. Within this range the method does not seem to have the positive effects that it did using only ω as switching criterion and the residuals often diverge which suggests that with these alternate switching criterion, this method becomes unstable.

The tests were carried out by starting with BiCGStab at the first step and taking a BiCG step when ω was smaller than the input tolerance. For every iteration this switching test was carried out and so the number of switchings varied depending on the example and the tolerance that was used for each. We have also found that the prescribed level of tolerance often had a significant effect in the performance of the mixed method. For all tests we have used a prescribed tolerance level of $Tol = 5e-3$

[3]this is seen in Figure B.6

unless otherwise noted.

As is seen in the convergence histories for Examples 1 and 2 (Figures 3.4 and 3.5 respectively) the mixed method shows improved convergence over the standard methods. In Example 1 this is particularly apparent as the standard BiCGStab stagnates and the mixed method converges in fewer iterations than even BiCG. In Example problems 3 and 4, there were no noticeable improvements over the BiCGStab (the standard method converged quite quickly over BiCG and thus there was no benefit in using the mixed method in these cases) although we include the results from Example 4 in Figure 3.6. This example has converged due to the fact that we have input a large input tolerance and allowed many BiCG switchings to occur.

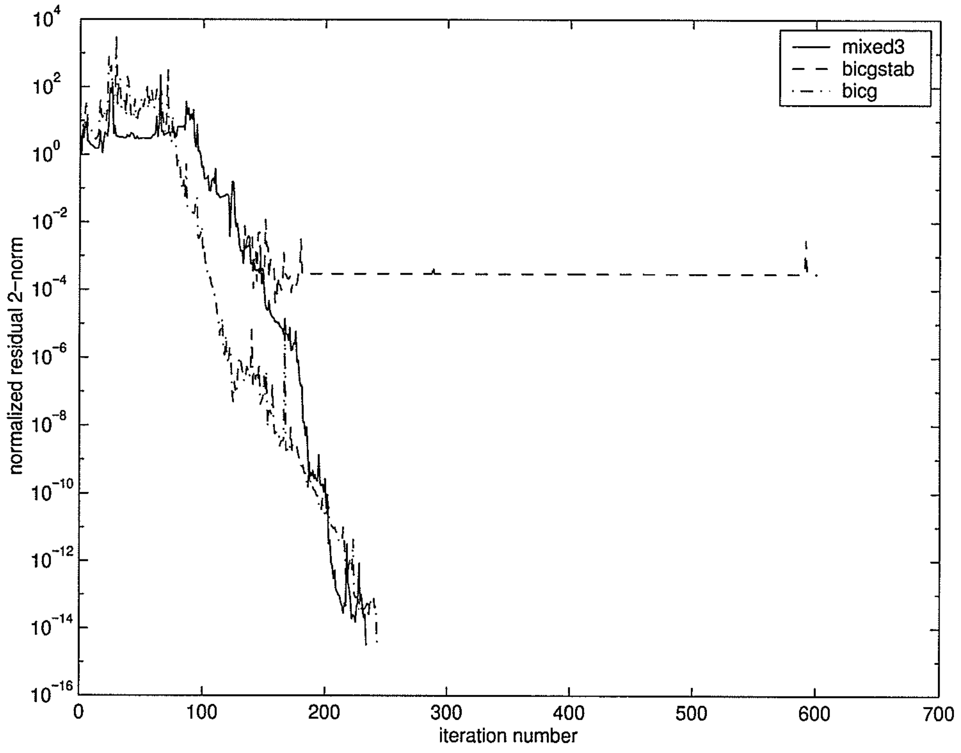
In a couple of tests, we found that a tolerance of $5e-3$ gave somewhat unstable convergence and by adjusting this to $3e-3$, the mixed method converged quite well. These examples have been left for the Appendix and are shown in Figures B.1 and B.2. Since this method experienced some unstable behaviour when different tolerance levels were used, it is possible that some other switching criterion could be used which may not be so sensitive and thus could be used for all the problems. Although a few other criteria were tested (as mentioned previously) none was found to give such improved test results. We have also tried this method (as in the generalized BiCG method) to begin with BiCG and switch to BiCGStab when a large ω was reached. Although this is beneficial when using CGS, as the local residuals often become quite large in the beginning, we have found no benefit with this approach with BiCG-BiCGStab. We feel that this method has its maximum benefit as it has been presented although further study should be done into switching criterion in order to make this method more user friendly.

The number of switchings was quite consistent for each Example as the chosen tolerance level for ω was generally the same for each. To summarise the number of switchings, Example 1 problems all had fewer than 5 switchings. The convergence history in Figure 3.5 for example had only one switching which occurred at iteration 142. We can see that at this point BiCGStab begins it's stagnation, and is beginning to bounce back and forth. For the Example 2 problems, all had more than 10 switchings and with these examples we have used a very large input $Tol = 1e-2$. It is clear in these cases that many more BiCG steps are taken, and although BiCGStab also converges in this problem, we have included it as an interesting case as this mixed method beats both BiCG and BiCGStab in terms of convergence.

The convergence histories in this method, as with the mixed BiCG-CGS method, are plotted with iteration number rather than number of matrix-vector multiplications. As we have mentioned in the previous section, two matrix vector operations are performed at each iteration for either step (BiCG or BiCGStab), and so the number of matrix vector operations for the mixed method is comparable to those of the standard BiCGStab and BiCG methods.

We should also note that in this mixed method we have not included tests to show the finite termination property as it was obvious that this method works quite well when implemented.

Figure 3.4: **Convergence History for Example 1** $\beta = -200, \gamma = 200$

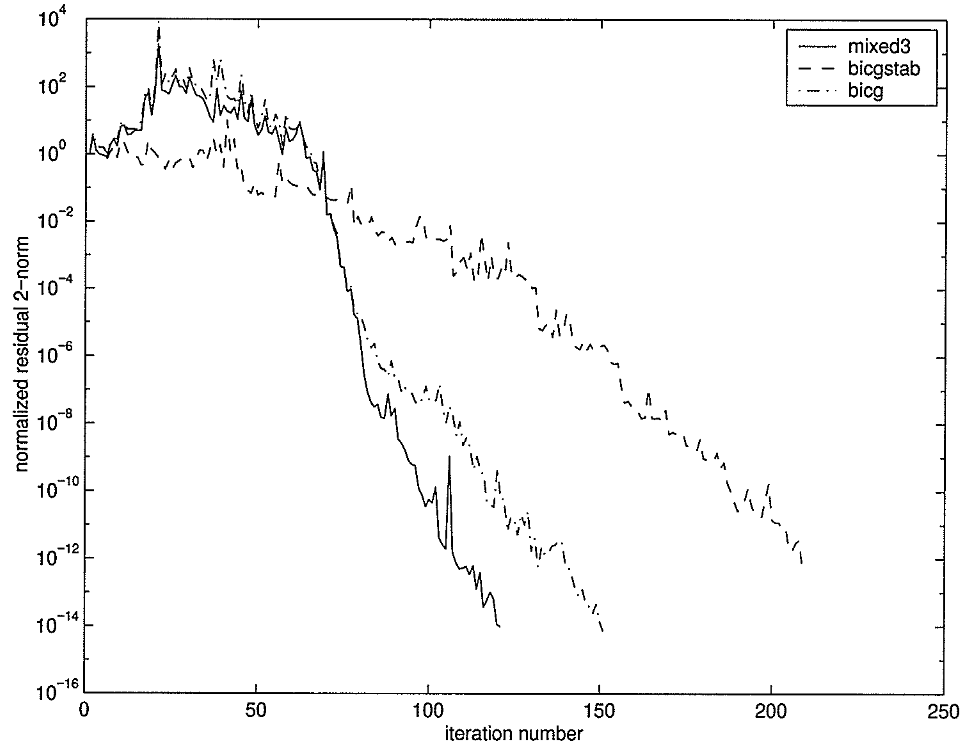


3.5 Numerical Instabilities

The motivation of these new methods are to eliminate the negative effects of the individual methods themselves by beginning with one method and then switching to the second method when a certain switching criterion is reached. In remedying problems in CGS, we try to improve upon the method by eliminating the large residuals and the instability that this method sometimes encounters. With BiCGStab, this means eliminating the problem of stagnation that is encountered in some examples, and hopefully improving the convergence.

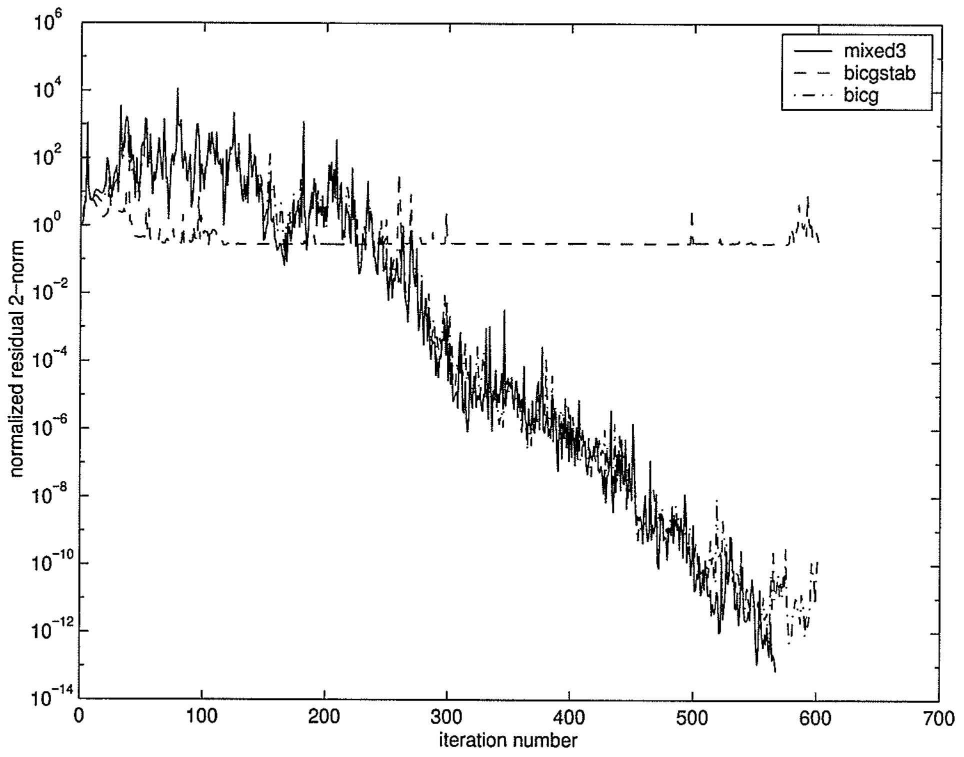
Although these mixed algorithms can be derived in several different ways, of which all are mathematically equivalent, among the different derivations of the recurrence

Figure 3.5: **Convergence History for Example 2** $\beta = -122, \gamma = 190$



for the residual vector r_n or search vector p_n, they do not give the same numerical results. This problem was particularly apparent in the mixed BiCG-CGS algorithm which we discuss in Section 3.2. Other problems associated with numerical instabilities that have been found in this study are the result of differences in the formulation of the coefficients, namely α and β. This has also been cited by [26] as a source for error and as is further pointed out, the polynomial coefficients which have been found from the use of a particular choice of polynomial at the beginning of the scheme do not seem to affect the sensitivity of the computations in this respect. Since we are concerned more with choosing a polynomial with which to achieve a significant reduction in the residual in each iteration, and which converges faster than BiCG itself, we are more concerned with choosing or formulating the polynomial in such a way that it does not lead to the

Figure 3.6: **Convergence History for Example 4** $\beta = -122, \gamma = 190$



deterioration of the BiCG part of the scheme, through numerical instabilities. So we may choose a polynomial which leads to a less than optimal reduction at a given step, but which improves the numerical stability of the BiCG coefficients.

Formulations for the coefficient β have proved to give extremely varying results when tests were run with these code in Matlab, as we have discussed in the Numerical testing section for the mixed BiCG-CGS method. Throughout the numerical testing, in all methods tested, β has been coded as

$$\beta_{n+1} = \rho_{n+1}/\rho_n$$

Other formulations from the literature were tested (as noted in Section 3.2) and this formulation was used as it was found to give the most stable results.

3.6 Discussion and Conclusions

In this thesis, we have introduced Krylov subspace methods for symmetric and nonsymmetric matrices A. The methods used for solving the nonsymmetric case which we have discussed include well known methods; the Biconjugate Gradient Method, the Conjugate Gradient Squared Method and the Biconjugate Gradient Stabilized Method. The latter method was developed to remedy the irregular convergence in CGS which was in turn developed to try and improve the convergence of BiCG and avoid using the transpose of A. To further these improvements, we have developed two new mixed methods based on either CGS or BiCGStab and have furthered the study done by Fokkema et al. [11] to include nonsymmetric test problems with the shifted CGS method with great success.

In all examples, shifted CGS tested on nonsymmetric matrices, has improved effects over the standard method. The convergence was improved by way of smaller residual spikes. This is an advantage as in the standard method these large spikes may cause instabilities or divergence near the beginning of the method. This adds to the body of work carried out in [11] and confirms what they have found in the symmetric test cases. In Appendix A, for different parameters β and γ, we see that in most cases the shifted CGS method is an improvement over the standard method. One case out of all the examples fails in this respect; in figure A.3 we see that shifted CGS fails to converge when CGS does. As with all iterative methods, this reinforces the fact that at present no one method is the winner in all cases, although for the majority of our tests, the shifted CGS algorithm is an improvement over the standard CGS method.

In the mixed BiCG-CGS we have attempted to further this study by allowing the switching to BiCG to occur whenever a large local residual in CGS is reached. It was hoped that this new method would have a nice stabilizing effect over the standard CGS method. Although this method did not greatly improve convergence performance in all cases, it did smooth the residuals of the standard CGS method and so was beneficial in some sense. This method was found to give unstable results for some problems, and unlike the BiCG-BiCGStab mixed method, here we did not find that we could 'fine tune' the tolerance level to give improved results. Because it is difficult to show these instabilities theoretically, we rely upon good results in numerical testing to show whether or not a method is a good choice and in this case we have shown that the mixing of the methods is working through testing of the finite termination property. It is possible that a reformulation of the algorithm may give improved results, but at present we have no additional insight into how this should be done. In Chan and Ye [6] it was suggested that perhaps an increased number of switchings may become too large to compensate any gain in stability with CGS. Although this study does not directly gain insight into this issue, it is possible that any number of switchings in this case has lead to the unstable nature of the mixed method. At present we do not have any theoretical results to confirm the precise cause for instabilities in CGS and perhaps it is here that we need to look, and then make improvements on the CGS-based mixed methods.

Promising results were found in the second mixed method, that based on BiCGStab and switching to BiCG when ω was found to be too small. In the majority of our test problems we found that the mixed method improved the convergence over the standard methods and in many cases where BiCGStab was found to stagnate, the mixed method converged, sometimes even beating the convergence in the standard BiCG method. Although the majority of tests were carried out with a prescribed tolerance of $5e-3$, we sometimes had to fine tune it to improve convergence.[4] It is possible however, that some other switching criterion could be used which may not be so sensitive and thus could be used for all the problems.

We have shown through numerical experiment the benefits (and problems) of each method. The Mixed BiCG-BiCGStab method offers an improvement to the class of problems in which BiCGStab stagnates and both the shifted CGS method and mixed BiCG-CGS method could be used in place of CGS to improve upon the convergence qualities of the standard method, particularly to smooth the large residuals. The shifted method is also a competitive method for nonsymmetric problems where the standard CGS fails to converge. Many more interesting cases could be looked at to further this study, and more theoretical insight is certainly needed to add to the body of knowledge we have of Krylov subspace iterative methods.

[4]In particular, in two problems where BiCGStab was stagnating, the prescribed tolerance had to be fine tuned to $3e-3$ so that the mixed method converged.

Appendix A

Selected Convergence Histories for the mixed BiCG-CGS method

Additional convergence histories for Example problems 1 through 4 follow. Two tests are presented from the mixed BiCG-CGS method in Figures A.1 and A.2. The remaining figures in Appendix A are convergence histories for the shifted CGS method presented in Section 2.4.5 and 2.4.6. We can see that in almost all cases tested this method converges faster and with fewer large residual spikes than the standard CGS although cases where this method have not improved the standard methods have also been included, which shows that no method is best for every problem.

Figures A.3 through A.5 give additional convergence histories for Example 1. Figures A.6 through A.8 follow for Example 2. We can see that shifted CGS has improved the convergence in almost all cases and that it also 'dampens' the large residuals found in the standard method.

Results from Example 3 are given in Figure A.9 and we include this example only to show the improvement of the smaller residuals over the spiky residuals in the standard method. The least improved is seen in this Sherman1 example, although here one improvement is made by way of the smaller residual spikes. This is an advantage as in the standard method these large spikes may cause instabilities or divergence near the beginning of the method. It is perhaps this problem which leads to the stagnation of CGS in both Examples 1 and 4 and which shifted CGS has nicely remedied.

Convergence histories for Example 4 are shown in Figures A.10 through A.13. This example again emphasizes the overall benefits of the shifted algorithm and we can see that even when the standard method fails to converge the shifted method improves this in all cases and converges. In all examples presented for the shifted method it is clear that this new method is better than the standard CGS method for all nonsymmetric test problems given here and improves upon both the convergence subduing the large residual spikes that are encountered in all the examples (here it improves in all tests). Overall, as with the mixed BiCG-CGS method, this method does not improve the convergence dramatically, but does improve it by smoothing the large residuals in the standard CGS method. We also see that it is capable of turning a divergent CGS into a convergent one, which we see in two separate test problems - a much improved history!

Figure A.1: **Convergence History for Example 1** $\beta = 200, \gamma = -200$

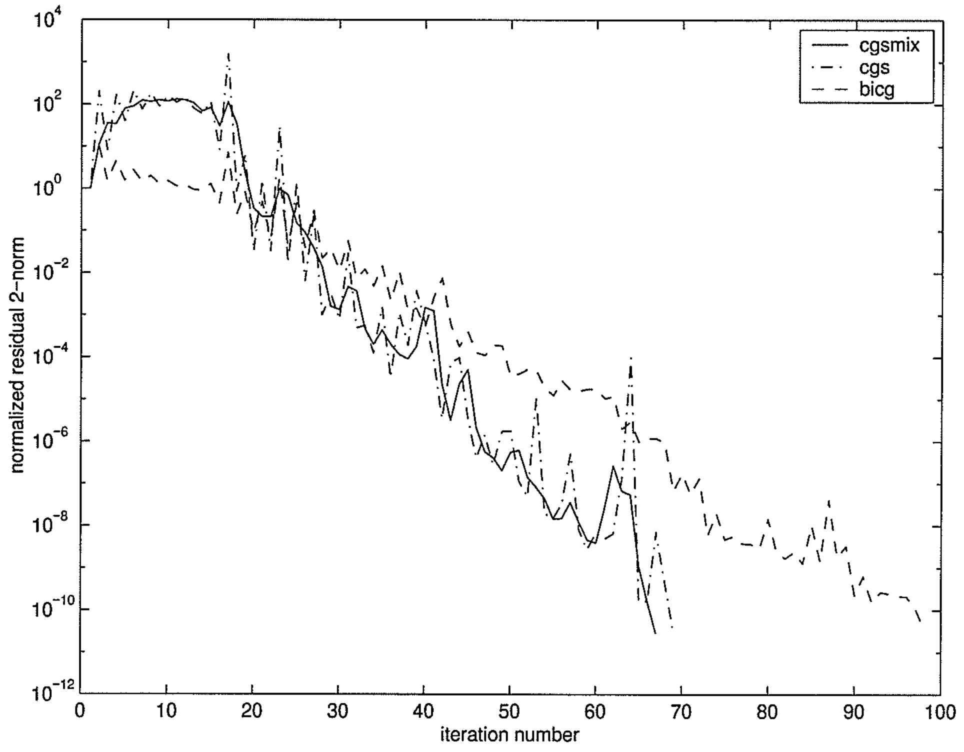


Figure A.2: **Convergence History for Example 2** $\beta = 200, \gamma = -200$

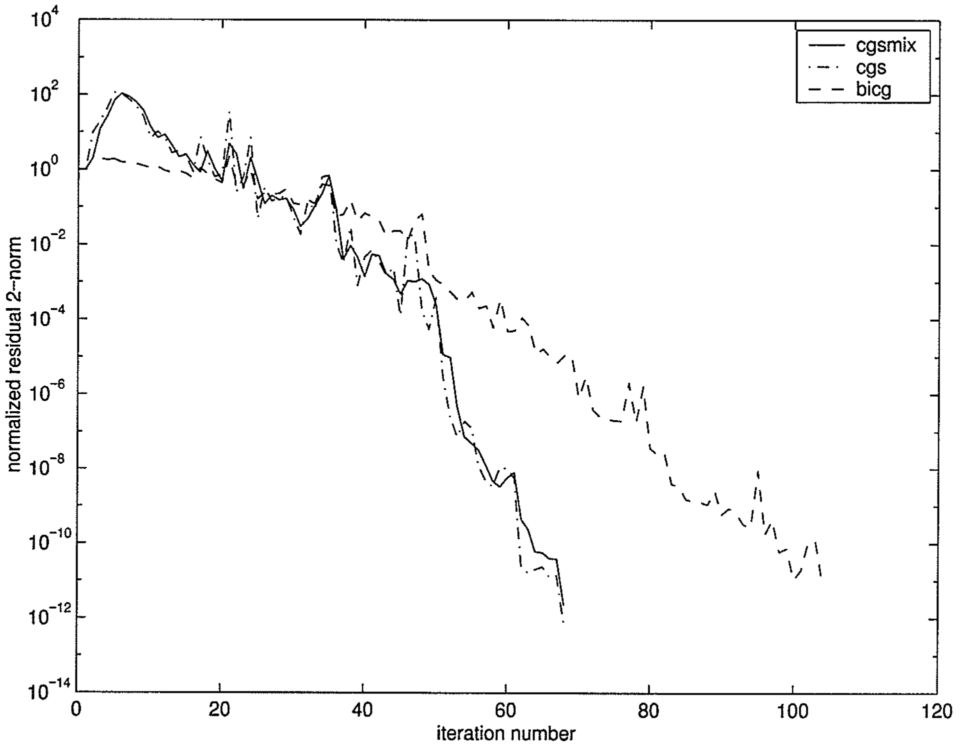


Figure A.3: **Convergence History for Example 1** $\beta = -250, \gamma = 100$

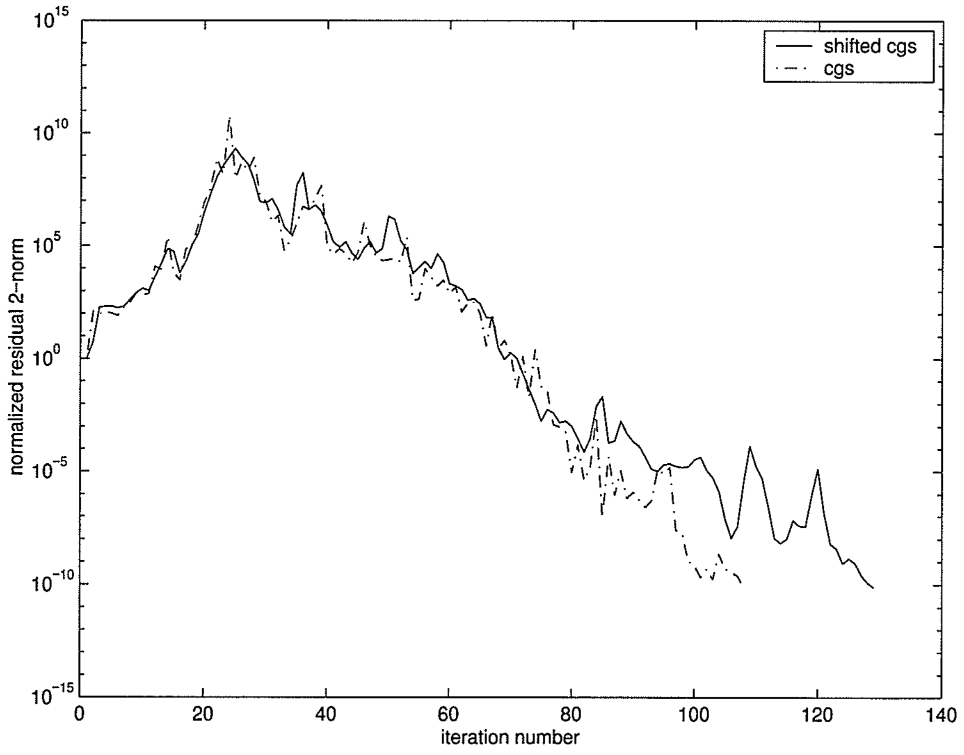


Figure A.4: **Convergence History for Example 1** $\beta = 100, \gamma = -360$

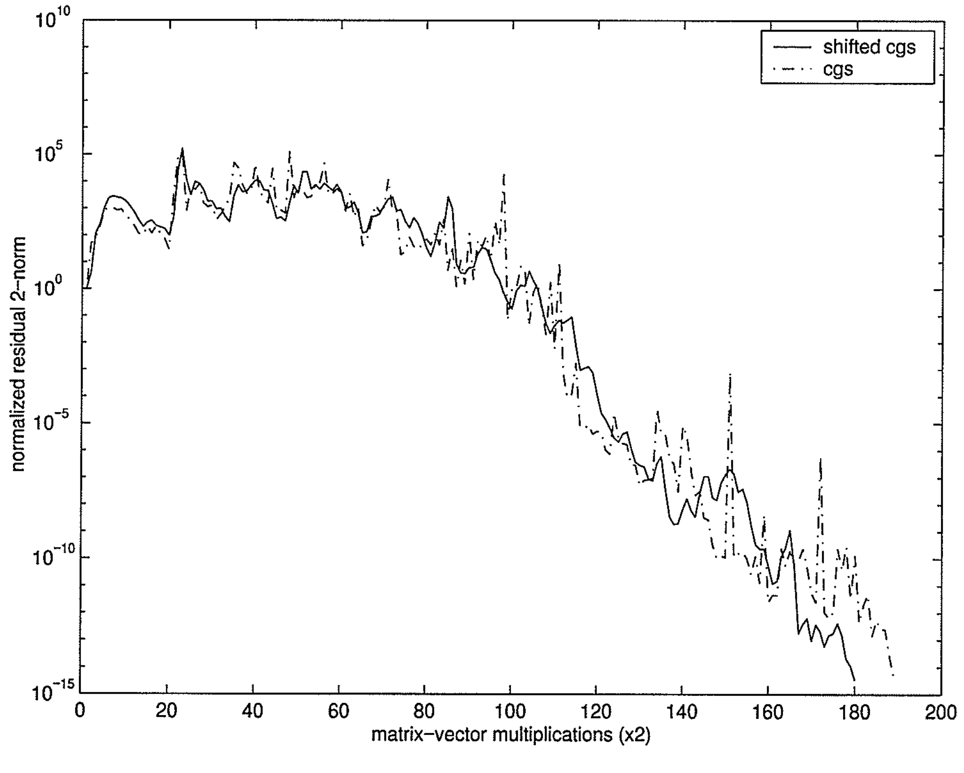


Figure A.5: **Convergence History for Example 1** $\beta = -220, \gamma = 220$

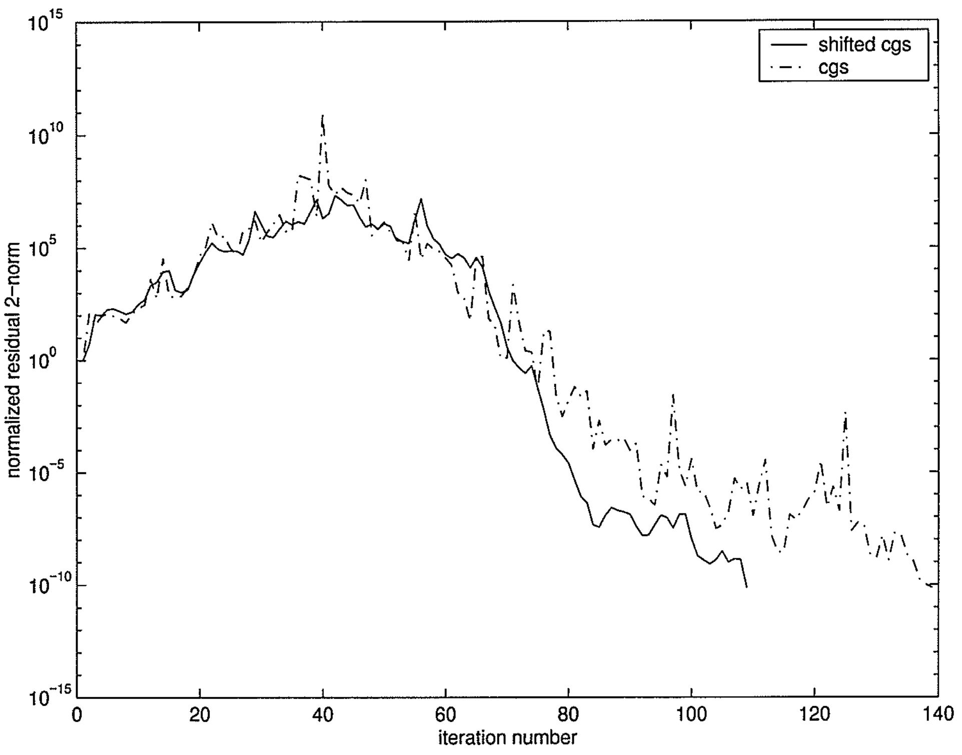


Figure A.6: **Convergence History for Example 2** $\beta = 123, \gamma = -22$

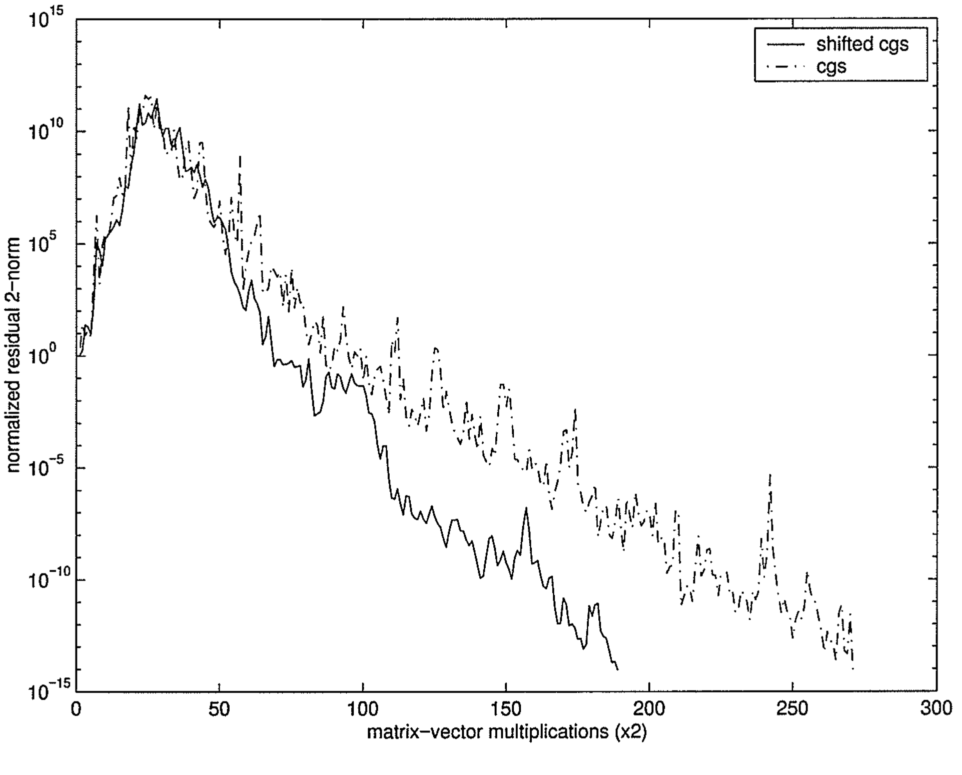


Figure A.7: **Convergence History for Example 2** $\beta = 200$ $\gamma = -200$

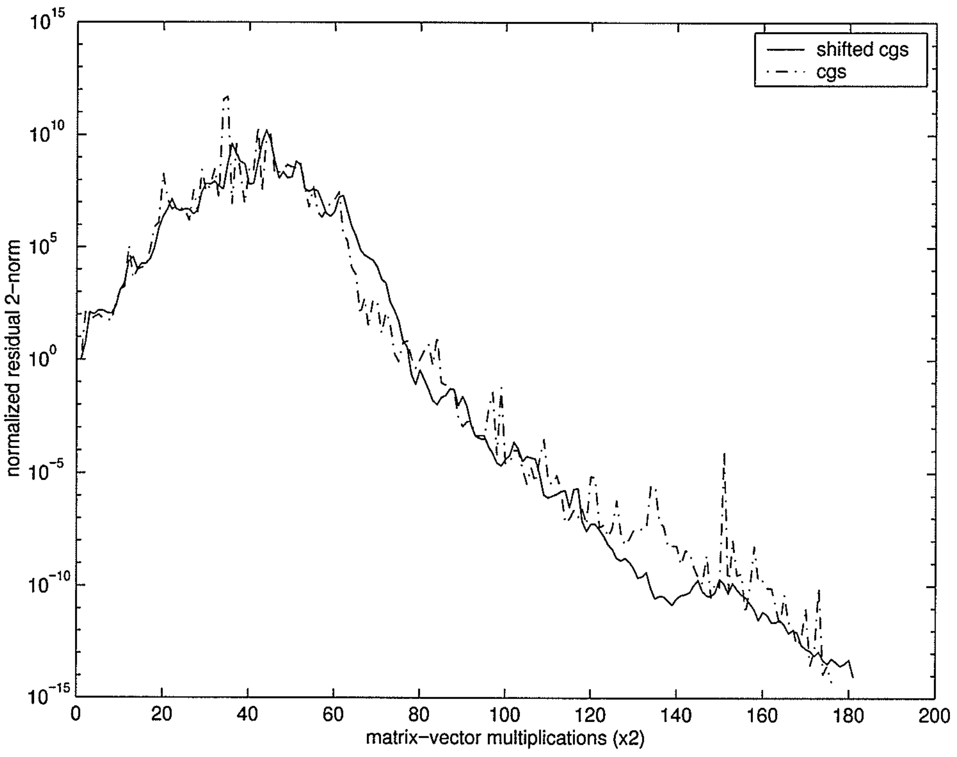


Figure A.8: **Convergence History for Example 2** $\beta = -120, \gamma = 190$

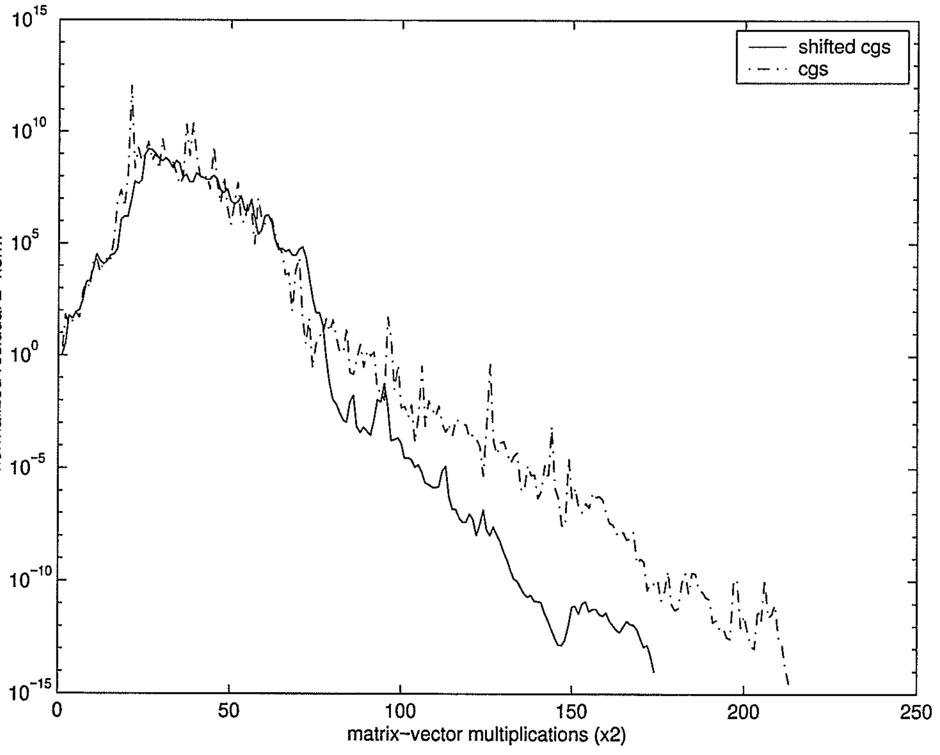


Figure A.9: **Convergence History for Example 3** Sherman 1

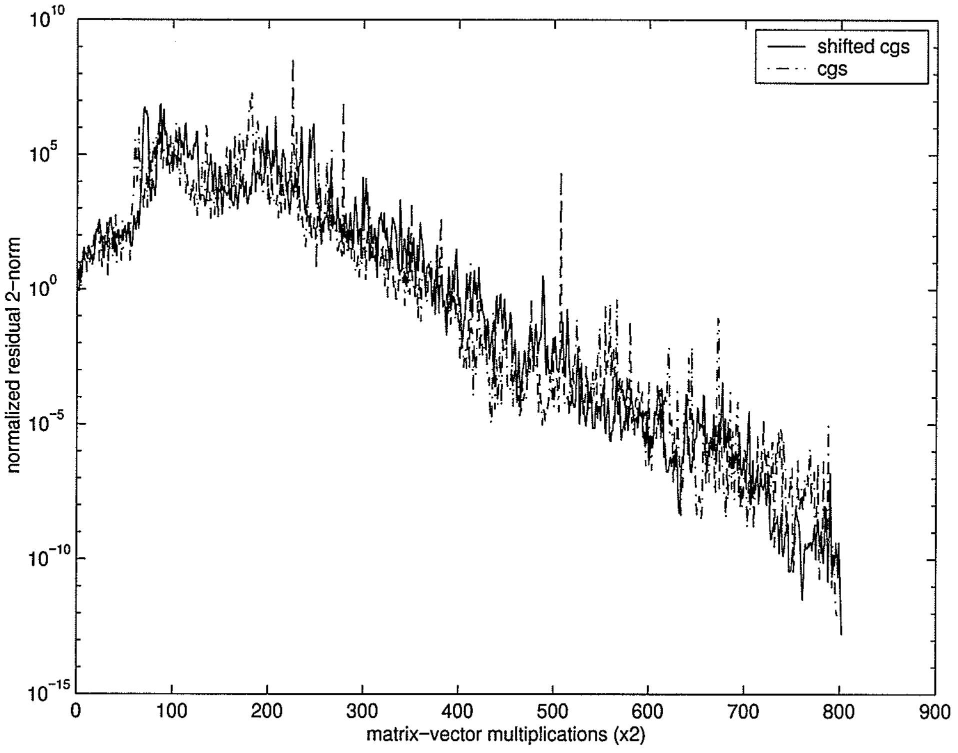


Figure A.10: **Convergence History for Example 4** $\beta = -200, \gamma = 200$

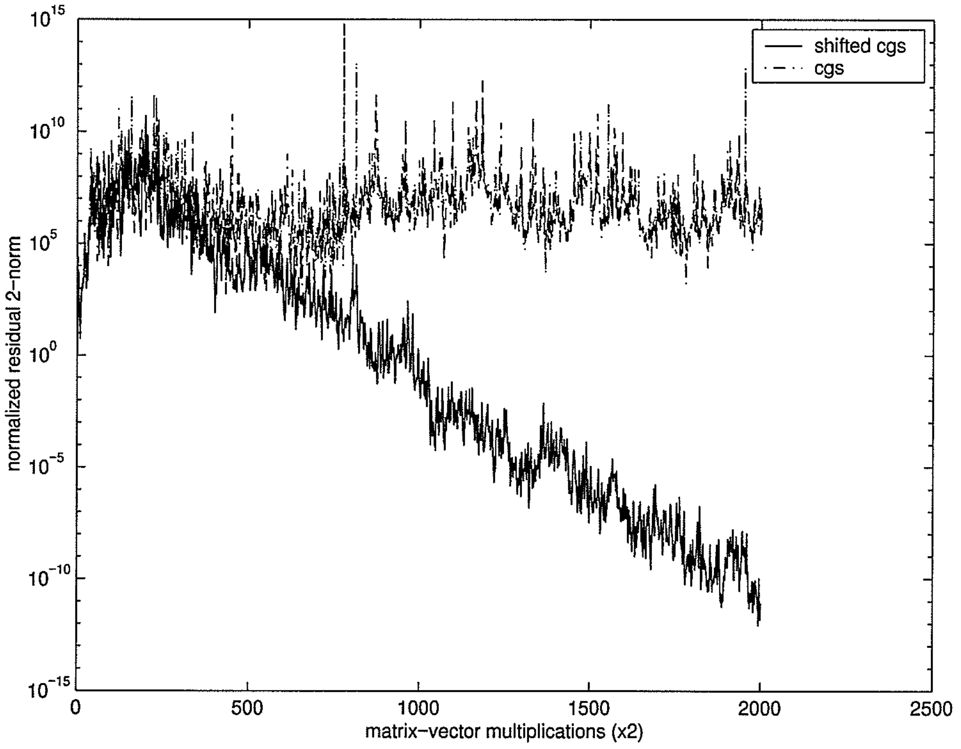


Figure A.11: **Convergence History for Example 4** $\beta = -122, \gamma = 190$

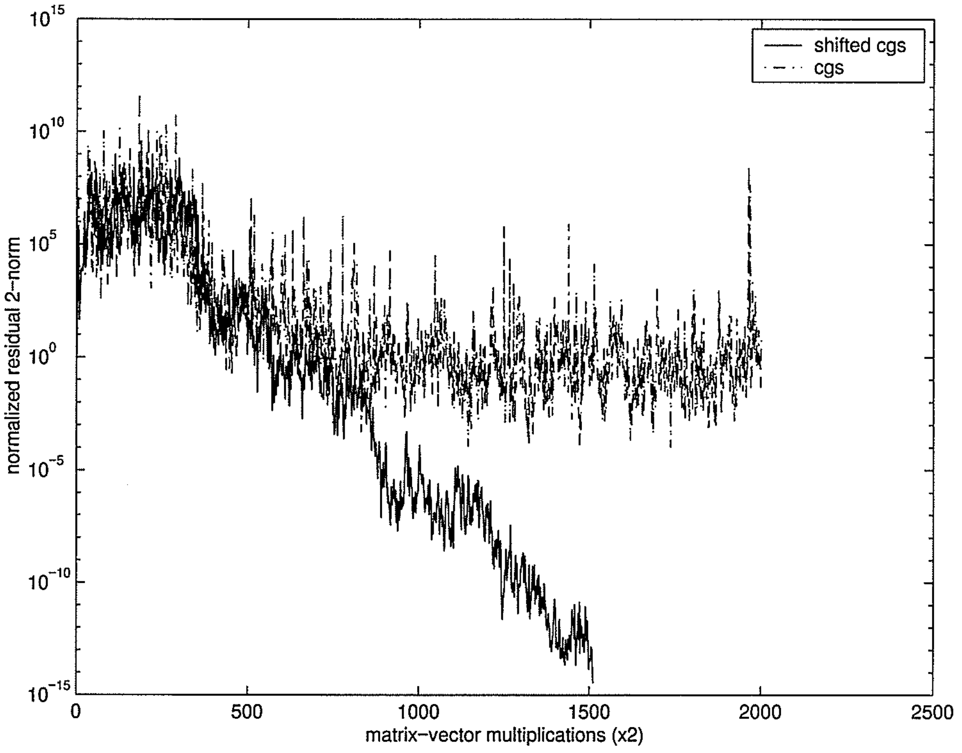


Figure A.12: **Convergence History for Example 4** $\beta = -190, \gamma = 100$

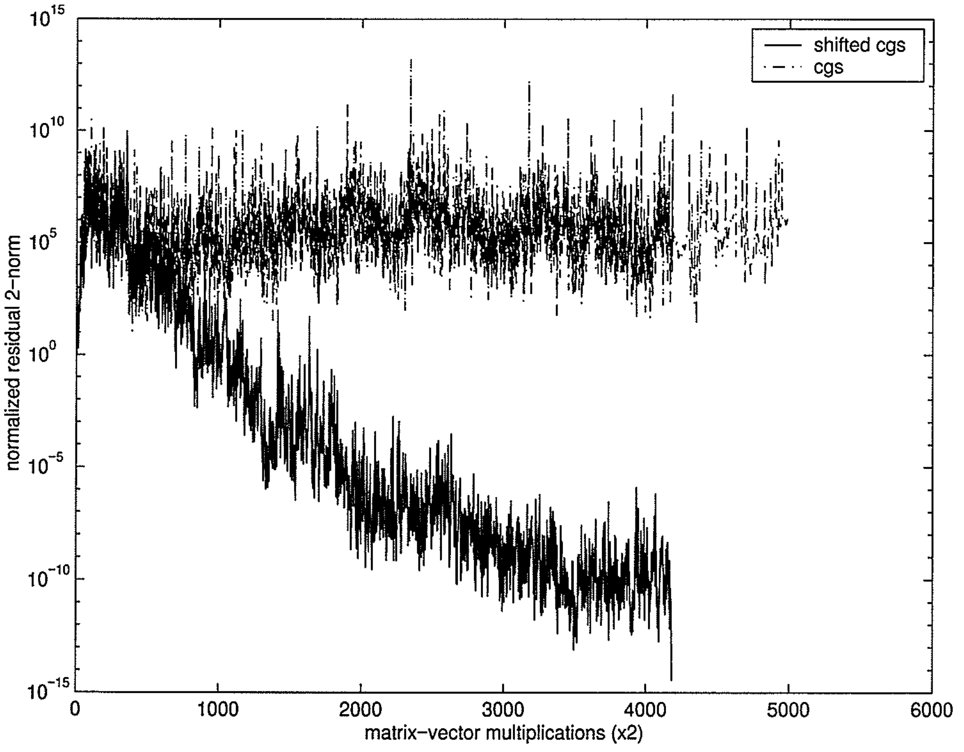
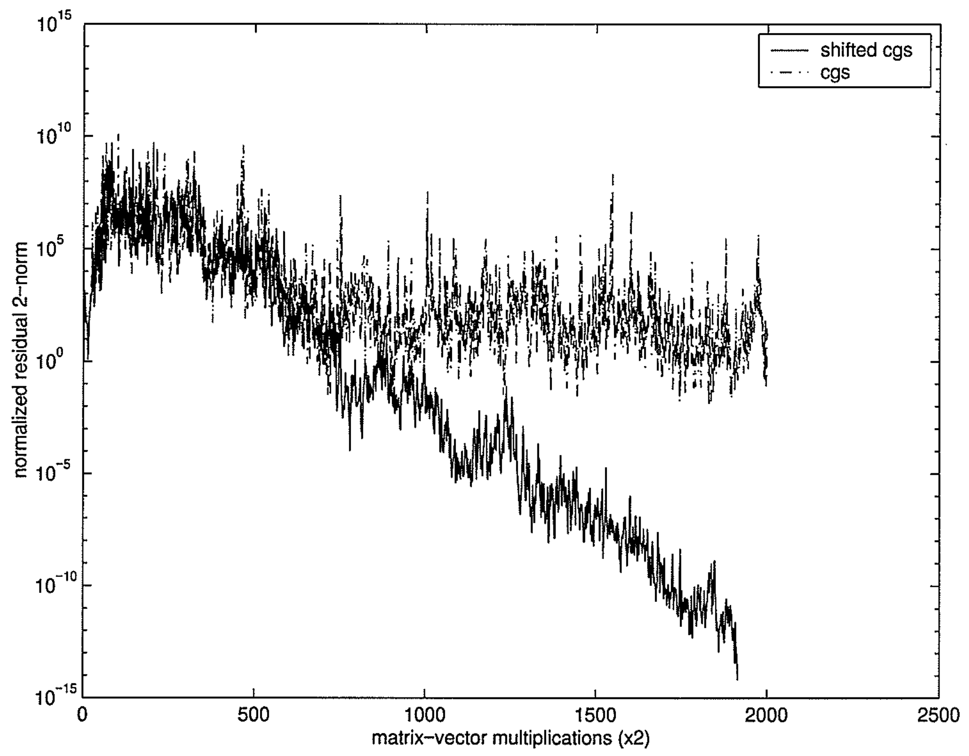


Figure A.13: **Convergence History for Example 4** $\beta = -200, \gamma = 100$



Appendix B

Selected Convergence Histories for Mixed BiCG-BiCGStab Method

Additional convergence histories for Example problems 1 and 2 follow. In all the Example 1 problems we see that the mixed method converges while BiCGStab is stagnating. We also see the method beating the convergence of both BiCG and BiCGStab in Example 2. Although this characteristic is not indicative of the overall performance of the method, we have included it here as an interesting case. Figure B.6 is included to show the effects of the range of input Tol for ω. Here we see that the method is essentially BiCGStab when a very small input is used and hence no switchings occur.

Figure B.1: **Convergence History for Example 1** $\beta = -220, \gamma = 220$

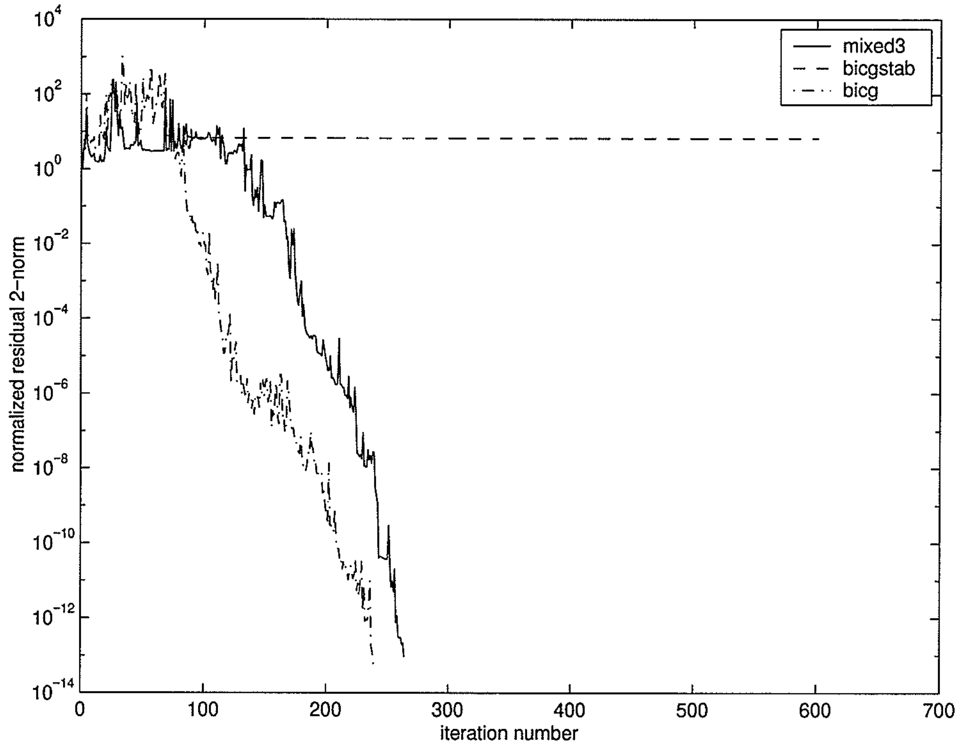


Figure B.2: **Convergence History for Example 1** $\beta = -190, \gamma = 100$

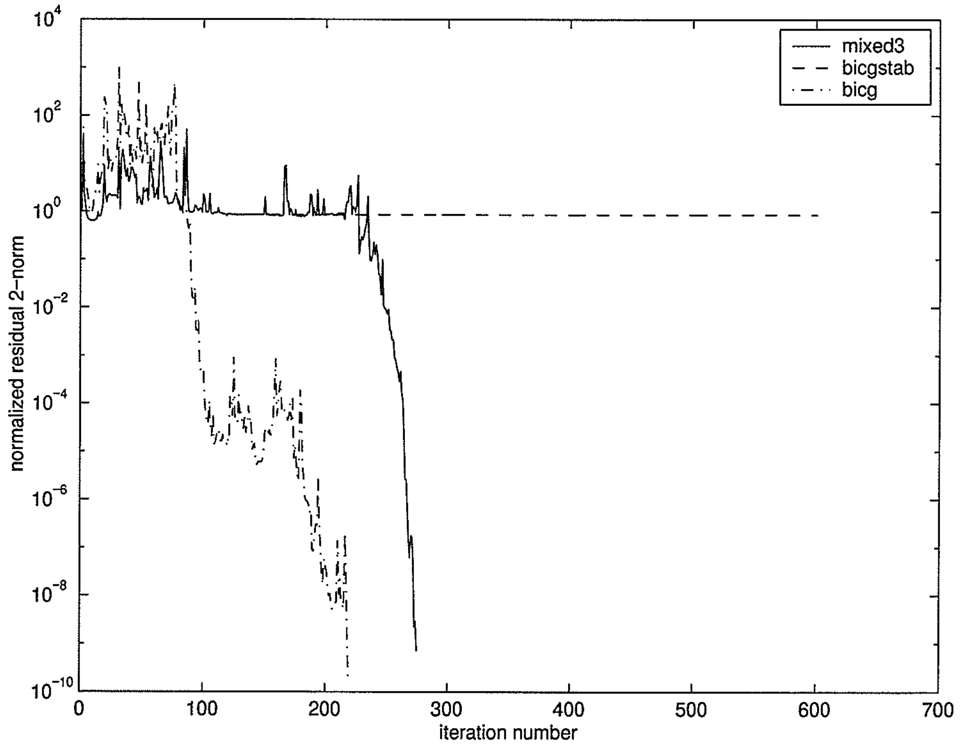


Figure B.3: **Convergence History for Example 1** $\beta = -250, \gamma = 100$

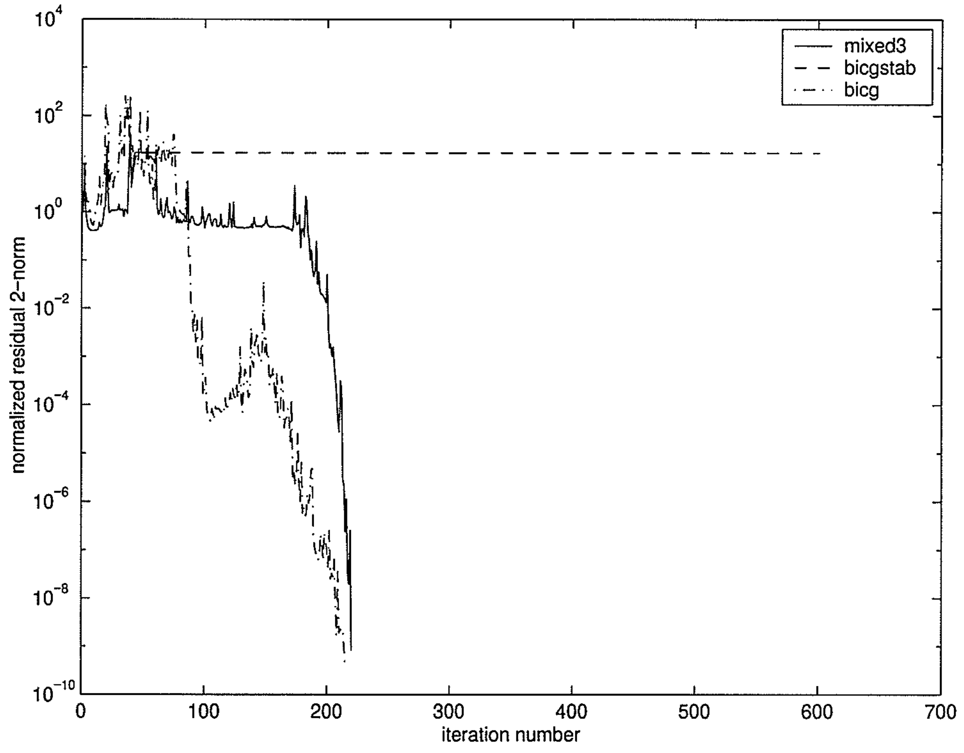


Figure B.4: **Convergence History for Example 1** $\beta = -300, \gamma = 300$

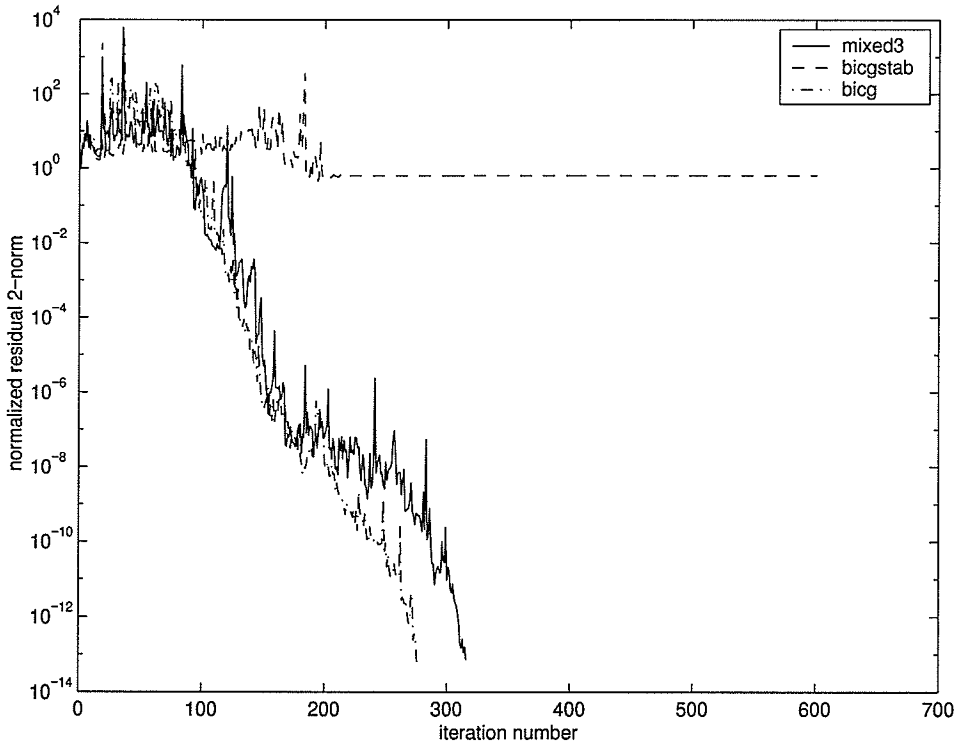


Figure B.5: **Convergence History for Example 2** $\beta = -220, \gamma = 220$

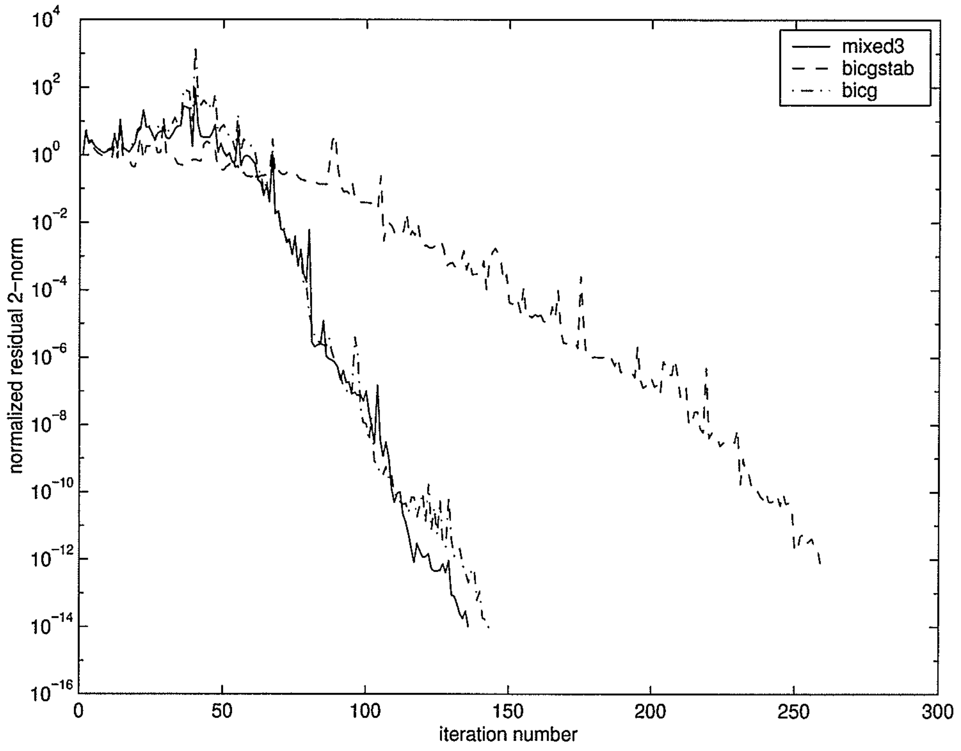
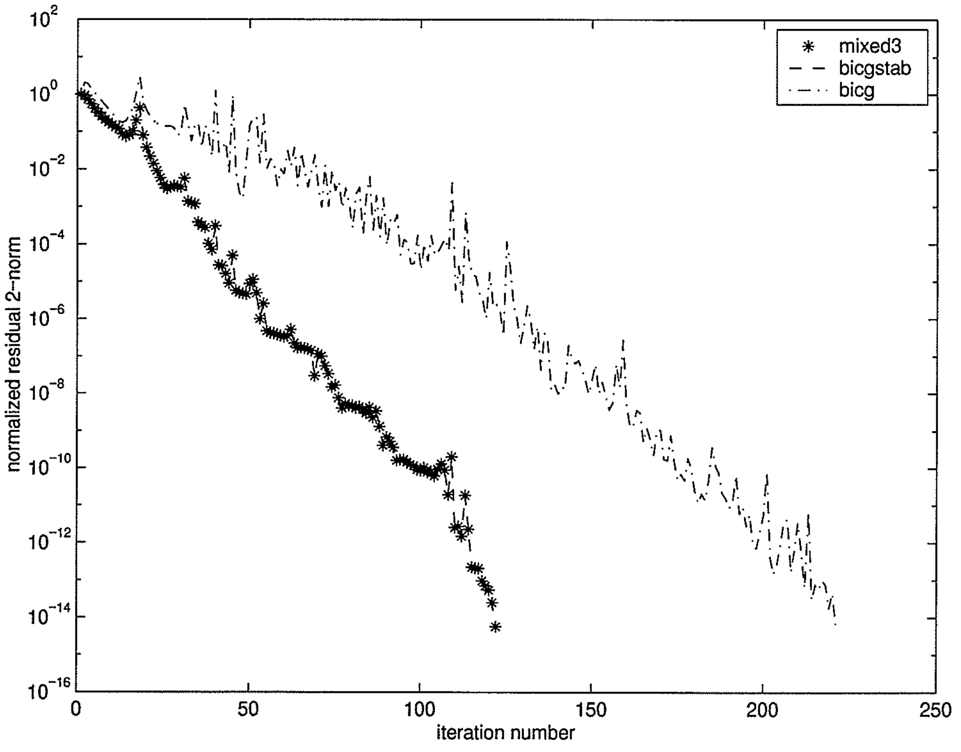


Figure B.6: **Convergence History for Example 4** $\beta = 100, \gamma = -100$



Bibliography

[1] W.E. Arnoldi, *The principle of minimised iteration in the solution of the matrix,* Quart. Appl. Math. 9 (1951), pp. 17-29.

[2] R. Askey, *Orthogonal Polynomials and Special Functions,* SIAM Regional Conference Series in Applied Mathematica 21, 1975.

[3] R. Barrett, M. Berry, T. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, V. Pozo, Romime C., and H. van der Vorst, *Templates for the solution of linear systems: Building blocks for iterative methods*, SIAM, Philadelphia, PA, 1994.

[4] C. Brezinski and H. Sadok, *Avoiding Breakdown in the CGS Algorithm,* Numerical Algorithms 1 (1991), 199-206.

[5] C. Brezinski, *Biorthogonality and Its Applications to Numerical Analysis,* Marcel Dekker, New York, 1992.

[6] T.F. Chan and Q. Ye, *A mixed-product Krylov Subspace method for solving non-symmetric linear systems,* Asian J. Math. 1 (1997): 422-434.

[7] T.F. Chan, L. de Pillis and H. van der Vorst, *Transpose-free formulations of Lanczos-type methods for nonsymmetric linear systems,* Numerical Algorithms 17 (1998), pp. 51-66.

[8] J. Demmel, *Numerical Linear Algebra,* SIAM, Philadelphia, 1996.

[9] I. S. Duff, R. G. Grimes, and J. G. Lewis, *Sparse Matrix Test Problems,* ACM Trans. Math. Softw., 15 (1989), pp.1-14.

[10] R. Fletcher, *Conjugate gradient methods for indefinite systems,* In G.A. Watson, editor, *Proceedings of the Dundee Biennial Conference on Numerical Analysis 1974,* pp. 74-89. Springer-Verlag, New York, 1975.

[11] D.R. Fokkema, G.L.G. Sleijpen and H.A. van der Vorst, *Generalized conjugate gradient squared,* J. Comp. and Appl. Math. 71: 125-146, 1996.

[12] R.W. Freund, *A transpose-free quasi-minimal residual algorithm for non-hermitian linear systems* SIAM J. Sci. Stat. Comput. 14: 470-482 (1993).

[13] M. Gutknecht, *Lanczos-type solvers for nonsymmetric linear systems of equations,* Acta Numerica 6: 271-397 (1997).

[14] A. Greenbaum, *Iterative Methods for Solving Linear Systems,* SIAM, Philadelphia, 1997.

[15] M.R. Hestenes and E.L. Steifel, *Methods of conjugate gradients for solving linear systems,* Journal of Research of the National Bureau of Standards, Section B, 49: 409-436 (1952).

[16] K.C. Jea and D.M. Young, *Generalized conjugate gradient acceleration of nonsymmetrizable iterative methods,* Linear Algebra and its Applications, 34, pp. 159-194 (1980).

[17] C. Lanczos, *Solution of Systems of Linear Equations by Minimized Iterations,* J. Res. Natl. Bur. Stand. 49, pp. 33-53 (1952).

[18] A.A. Nikishin and A.Yu. Yeremin, *Variable block CG algorithms for solving large sparse symmetric positive definite linear systems on parallel computers, I: General iterative scheme,* SIAM J. Matrix Anal. Appl. 16, pp. 1135-1153 (1995).

[19] D.P. O'Leary, *The block conjugate gradient algorithm and related methods,* Linear Algebra Appl. 29, pp. 293-322 (1980).

[20] A. Ruhe, *Implementation aspects of band Lanczos algorithms for computation of eigenvalues of large sparse symmetric matrices,* Mathematics of Computations, 33, pp. 680-687 (1979).

[21] Y. Saad, *Iterative Methods for Sparse Linear Systems* PWS Publishing, Boston, MA, 1996.

[22] Y. Saad and M.H. Schultz, *GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems* SIAM J. Sci. Stat. Comput. 7, pp. 856-869 (1986).

[23] G. Sleijpen and D. Fokkema, *BICGSTAB(l) for linear equations involving unsymmetric matrices with complex spectrum,* Electronic Trans. Numer. Anal. 1, pp. 11-32 (1993).

[24] G. Sleijpen and H. van der Vorst, *Reliable updated residuals in hybrid Bi-CG methods,* Computing 56:144-163 (1996).

[25] G. Sleijpen and H. van der Vorst, *Maintaining convergence properties of BiCGStab methods in finite precision arithmetic* Numerical Algorithms 10 (1995), pp. 203-223.

[26] G. Sleijpen, H. van der Vorst and D.R. Fokkema, *BiCGStab(l) and other hybrid Bi-CG methods* Numerical Algorithms 7(1994), pp. 75-109.

[27] P. Sonneveld, *CGS, A Fast Lanczos-type Solver for nonsymmetric Linear Systems* SIAM J. Sci. Stat. Comput. 10, p. 36-52 (1989).

[28] C. H. Tong and Q. Ye, *Analysis of the Finite Precision Bi-Conjugate Gradient algorithm for Nonsymmetric Linear Systems,* Stanford SCCM Report 95-11, Stanford University, Stanford, CA, October 1995.

[29] H. van der Vorst, *BiCGStab, A fast and smoothly converging variant of BiCG for the solution of nonsymmetric linear systems* SIAM J. Sci. Stat. Comput. 13, 631-644 (1992).

[30] Q. Ye, *Mixed product BiCG Krylov subspace methods for solving nonsymmetric linear systems,* Draft, 1999.

[31] S. Zhang, *GPBi-CG: Generalized product-type methods based on BiCG for solving nonsymmetric linear systems* SIAM J. Sci. Stat. Comput. 13, 631-644 (1992).